第 38 卷 第 3 期 计 算 机 工 程 2012 年 2 月
Vol.38 No.3 Computer Engineering February 2012

·工程应用技术与实现· 文章编号：1000—3428(2012)03—0228—06 文献标识码：A 中图分类号：TP301.6

优化的可重构多常数乘法器生成算法

叶晓敏，王侃文，陈佳临，周学功，王伶俐

（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：针对线性变换中单个定点数输入与多组定点常数相乘的问题，以加/减法器、移位器和多路选择器为基本单元，提出一种可重构多常数乘法器(RMCM)生成算法。该算法分别运用局部优化和全局优化 2 种策略，通过计算多常数乘法器(MCM)之间的拓扑相似度，对 MCM 的有向无环图(DAG)进行合并，选取其中估算面积最小的 DAG 作为输出。实验结果表明，利用该算法生成的乘法器在面积上具有优势，可用于实现视频多标准中多组不同系数的线性变换。

关键词：多常数乘法器；可重构多常数乘法器；有向无环图；定点算术；线性变换

Optimized Generation Algorithm for Reconfigurable Multiple-constant Multiplier

YE Xiao-min, WANG Kan-wen, CHEN Jia-lin, ZHOU Xue-gong, WANG Ling-li

(State Key Laboratory of ASIC & System, Fudan University, Shanghai 201203, China)

【Abstract】Aiming at the problem of the Multiplier of a fixed-point input by multiple sets of fixed-point constants in linear transformation, this paper proposes Reconfigurable Multiple-constant Multiplier(RMCM) algorithm making use of adders/subtractors, shifters and multiplexers. It generates RMCM in two strategies, which are local optimization and global optimization, and selects the best Directed Acyclic Graph(DAG) with the smallest areas as the output. Experimental result shows that the Multiplier generated by the proposed algorithm has less area costs and can be applied in linear transformation with several different sets of coefficients.

【Key words】Multiple-constant Multiplier(MCM); Reconfigurable Multiple-constant Multiplier(RMCM); Directed Acyclic Graph(DAG); fixed-point arithmetic; linear transformation

DOI: 10.3969/j.issn.1000-3428.2012.03.076

1 概述

在数字信号处理中，存在大量的常数乘操作，可通过加法、减法和移位实现一个定点输入与一个或多个定点常数相乘的常数乘法器[1-3]。针对线性变换中单个定点数输入与多组定点常数相乘的问题，本文提出一种优化的可重构多常数乘法器(Reconfigurable Multiple-constant Multiplier, RMCM)生成算法。RMCM 的输入 x 是一个定点数，通过控制输入端 i，选择与 N 组、每组 M 个元素的给定定点常数集合$\{c_{11}, c_{12},\cdots,c_{1M}\},\{c_{21},c_{22},\cdots,c_{2M}\},\cdots,\{c_{N1},c_{N2},\cdots,c_{NM}\}$中的一组相乘，输出为$\{c_{i1}x,c_{i2}x,\cdots,c_{iM}x\}$，控制输入端 i 的位宽为 $\text{lb}N$。与使用 N 组常数的查找表和 M 个通用乘法器的方法相比，本文用加法器、移位器和多路选择器实现可重构多常数乘法器，运用 2 种策略分别针对局部和全局进行优化，在生成过程中计算多常数乘法器(Multiple Constant Multiplier, MCM)之间的拓扑相似度，并对其进行合并，从而使其具有很小的综合后面积。

2 相关研究

假设作为乘数的常数都是整数，一个位宽为 w 的常数 c 的二进制表示为：

$$c = b_{w-1}b_{w-2}\cdots b_1b_0 = \sum_{i=0}^{w-1} b_i 2^i, b_i \in \{0,1\} \quad (1)$$

现有基于加法和移位的常数乘法器可分为 3 类：单常数乘法器(Single-constant Multiplier, SCM)，多常数乘法器，可重构单常数乘法器(Reconfigurable Single-constant Multiplier, RSCM)，分别如图 1~图 3 所示。本文提出的可重构多常数乘法器如图 4 所示。

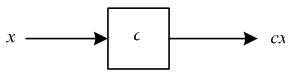


图 1 单常数乘法器

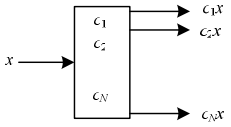


图 2 多常数乘法器

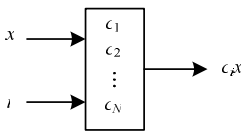


图 3 可重构单常数乘法器

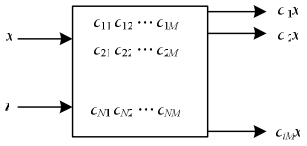


图 4 可重构多常数乘法器

2.1 单常数乘法器

一个变量 x 和一个定点整常数 c 相乘的乘法 $y=cx$ 可以分解为加法、减法及移位，如图 1 所示。找到一个分解、使加法运算数目最少的问题称作单常数乘法器问题。SCM 问题是一个 NP-hard 问题[1]。

从式(1)可以看出，输入 x 和常数 c 的积 cx 可以直接映射

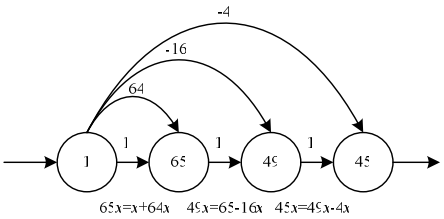
基金项目：复旦大学专用集成电路与系统国家重点实验室自主课题基金资助重点项目(09ZD005)

作者简介：叶晓敏(1986－)，男，硕士研究生，主研方向：多核处理器，FPGA 软件设计；王侃文、陈佳临，博士研究生；周学功，助理研究员；王伶俐，教授

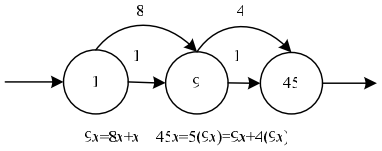
收稿日期：2011-07-30 **E-mail：**yexm0310@hotmail.com

为一系列的移位和加法。如果 c 的二进制表示有 k 个非 0 位，则就加法数目而言，乘法代价为 $k-1$。由于移位可以通过硬件上的直接连接实现，本文假设移位的代价可忽略。并且由于加法和减法 2 种运算在复杂度上几乎一样，这 2 种运算可用相似的硬件结构实现，因此用“加法”代替加法及减法。

针对 SCM 问题，文献[2]提出了一种 CSD 重编码方法，相对于二进制分解平均节省 33%的代价。但由于只考虑了一种图拓扑结构，CSD 不能得到最优分解结果[4]。如图 5 所示，45 是通过 CSD 重编码无法获得最优分解的最小自然数，其中，最优分解使用了一种与 CSD 分解不同的图拓扑结构。图 5 是常数乘法器的有向无环图(Directed Acyclic Graph, DAG)表示。图中节点代表加法，边代表加法间的数据流。每个节点的入度为 2，代表每个加法有 2 个操作数。每条边上标记 2 的 s 次幂 $k=2^s$，代表该边上对操作数移 s 位。每个节点都标注了中间常数 f，这些中间常数称为基数[5]，x 是 DAG 的输入，则产生的输出为 fx。



(a)CSD 分解



(b)最优 SCM 分解

图 5　以 45 为例的 SCM 有向无环图

2.2　多常数乘法器

SCM 问题的一种延伸是一个变量 x 与 N 个常数 $c_1,c_2,\cdots,c_N$ 在一个乘法器中并行相乘的问题，如图 2 所示。由于常数分解的中间结果可以被共享，因此多常数乘法分解后的运算数目，要比几个单常数分解后的运算数目之和少。找到一个与多个常数并行相乘的分解，使其加法运算数目最少的问题称作 MCM 问题。MCM 通常被应用在线性变换中。针对 MCM 问题，文献[4]提出了一种基于图的算法，这是目前最好的方法。DAG 中的每个节点都可以用一个 A 运算表示：

$$A_p(u,v)=|(u\ll l_1)+(-1)^s(v\ll l_2)|\gg r=|2^{l_1}u+(-1)^s2^{l_2}v|2^{-r} \tag{2}$$

其中，u 和 v 是节点的整数输入；$l_1, l_2\geqslant 0$ 为整数；$r\geqslant 0$ 为整数；$s\in\{0, 1\}$为符号位，控制节点为加法或减法；<<表示左移；>>表示右移；$p=(l_1, l_2, r, s)$是参数集合。为了保留输出的所有有效位，$2r$ 必须整除。用图表示一个 A 运算如图 6 所示。

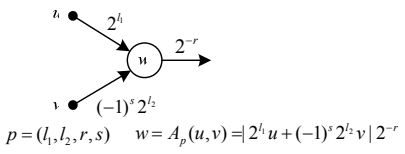


图 6　A 运算的图表示

文献[4]将 DAG 约束为奇基数图，即所有 A 运算的结果都为奇数。使用奇基数图的算法可以通过合适的右移使目标常数变为奇数。

文献[4]的算法通过从 A 运算得到的结果中选取及向 DAG 中添加下一个节点，其具有可终止性和最优性。本文基于该算法生成 MCM DAG。

2.3　可重构单常数乘法器

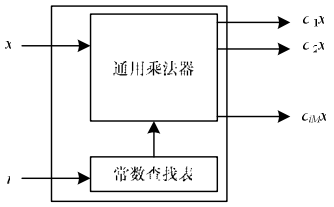
SCM 问题的另一种延伸是一个变量 x 与 N 个常数 $c_1, c_2,\cdots,c_N$ 相乘，由 lb$N$ 位的控制端 i 选择输出 c_ix 的问题，如图 3 所示。乘法器中包含了多路选择器，使用控制逻辑切换状态，以获得与不同常数相乘的输出。找到最少加法运算数目及多路选择器数目以实现这种乘法，称作可重构单常数乘法器问题。

文献[6]提出了一种针对 RSCM 的生成方法，利用 SCM DAG 之间的拓扑相似度，通过在 DAG 中插入多路选择器，合并多个 SCM DAG，达到共享加法器、节省面积的目的。

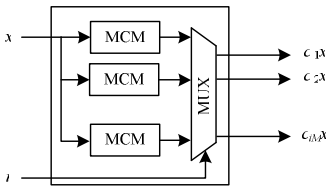
3　可重构多常数乘法器

本文提出的可重构多常数乘法器问题可描述为：对于给定 N 组、每组 M 个元素的定点常数集合$\{c_{11},c_{12},\cdots,c_{1M}\}$, $\{c_{21}, c_{22},\cdots, c_{2M}\},\cdots,\{c_{N1},c_{N2},\cdots,c_{NM}\}$，输出一个只包含加法器、移位器和多路选择器的 DAG，在 DAG 中可以通过 lb$N$ 位的控制端 i 选择某一组特定常数$\{c_{i1},c_{i2},\cdots,c_{iM}\}$，使输入 x 与之相乘，输出对应的一组乘积$\{c_{i1}x,c_{i2}x,\cdots,c_{iM}x\}$，用于实现多组不同系数的线性变换。目前尚未见研究这一问题的论文。

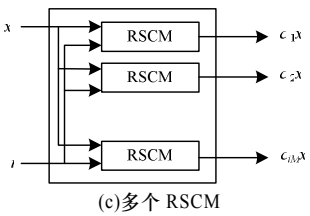
该电路的直接实现如图 7(a)所示。预设常数集合存在于一个查找表中，对于给定的配置，控制输入端从查找表中选择一组常数作为通用乘法器的输入。但这种方法的面积代价过大[4]。另一种实现方法是多个 MCM 的输出加上多路选择器，由控制输入端进行选择，如图 7(b)所示。这种方法也不是最优的，因为每种配置下或多或少会有加法器未被使用到。还有一种实现方法是使用多个 RSCM 结果拼接成 RMCM，如图 7(c)所示。这种方法没有考虑到同组常数之间的相关性和 DAG 相似度，因此，也不是最优的。本文提出的优化 RMCM 算法综合考虑了共享可能的中间计算结果，并利用其中 DAG 的相似性，因此，在面积上具有优势。与其他多常数乘法器的实现相比，面积平均节省 10%以上。



(a)通用乘法器与查找表



(b)多个 MCM



(c)多个 RSCM

图 7　3 种可重构多常数乘法器的实现方式

3.1 RMCM 生成算法

本文提出一种优化的 RMCM 算法，首先对 N 组给定常数集合进行 MCM DAG 生成，在得到 N 组给定常数集合的 MCM DAG 之后对这些 DAG 进行合并，选择其中估算面积最小的一个作为 RMCM DAG 输出。算法的高层次描述如下：

算法 1 可重构多常数乘法器生成算法

输入 N 组、每组 M 个元素的给定定点常数集合$\{c_{11}, c_{12},\cdots,c_{1M}\},\{c_{21},c_{22},\cdots,c_{2M}\},\cdots,\{c_{N1},c_{N2},\cdots,c_{NM}\}$

输出 估算面积最小的、通过 lb$N$ 位的控制端 i 选择与某组特定常数$\{c_{i1},c_{i2},\cdots,c_{iM}\}$乘积的合并 DAG

```
RMCM(constant_set[N])
1:      {strategy one}
2:      for i = 1,2,…,N do
3:          mcm_dag[i] = MCM(constant_set[i])
4:      end for
5:      rmcm_dag_1 = FuseMCMDAGs(mcm_dag[N])
6:      cost_1 = EstimateArea(rmcm_dag_1)
7:      {strategy two}
8:      mcm_dag_all = MCM(constant_set[N])
9:      mcm_dag[N] = SplitDAGs(mcm_dag_all)
10:     rmcm_dag_2 = FuseMCMDAGs(mcm_dag[N])
11:     cost_2 = EstimateArea(rmcm_dag_2)
12:     return (cost_1 > cost_2) ? rmcm_dag_1 : rmcm_dag_2
```

从算法描述中可以看到，算法使用了以下 2 种策略生成 RMCM DAG，每种策略中都用到了文献[4]的 MCM DAG 生成算法：

第 1 种策略是进行局部优化：(1)对每组常数集合分别进行 MCM DAG 生成，得到 N 个 MCM DAG，这样可以使每组常数集合的 MCM DAG 内部达到最优，没有考虑不同 MCM DAG 之间的相关性。(2)调用函数 FuseMCMDAGs 对这些 MCM DAG 进行合并，生成 RMCM DAG。

第 2 种策略是进行全局优化：(1)对 N 组常数集合中的所有常数进行 MCM DAG 生成，这样可使各组常数集合之间的 MCM DAG 相关性达到最大。(2)在函数 SplitDAGs 中将这整个 DAG 拆分为 N 个 MCM DAG。(3)调用 FuseMCMDAGs 进行合并。FuseMCMDAGs 是合并 MCM DAG 的函数，该函数反复调用下一节中提到的 FusePairMCMDAGs 对 N 个 MCM DAG 进行合并。

对合并后的 DAG 分别调用面积估算函数 EstimateArea 进行面积代价的估算，比较估算出的面积代价，选择 2 种策略中估算面积较小的一个 RMCM DAG 作为最终的输出。

图 8~图 12 是运用 2 种策略分别对 2 组常数{45,19}、{36,22}进行 RMCM 运算的结果。

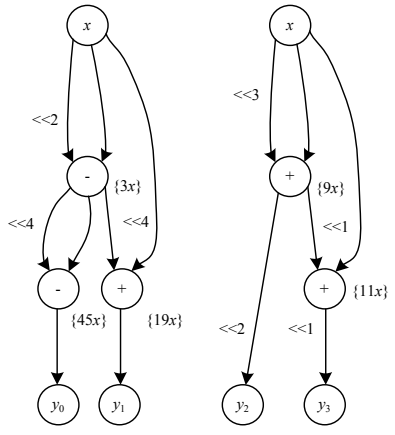


图 8 策略 1 第(1)步生成的独立 MCM DAG

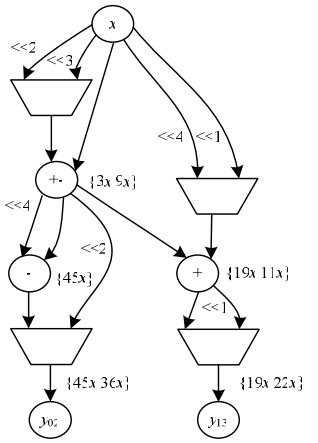


图 9 策略 1 第(2)步合并后的 DAG

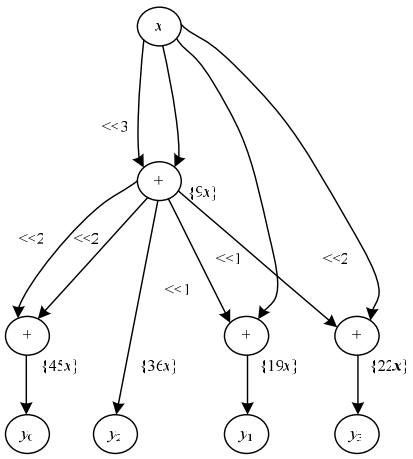


图 10 策略 2 第(1)步生成的 DAG

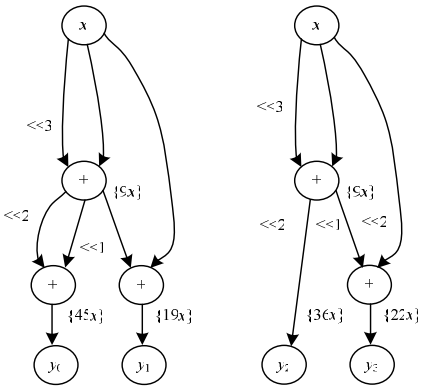


图 11 策略 2 第(2)步拆分后的 DAG

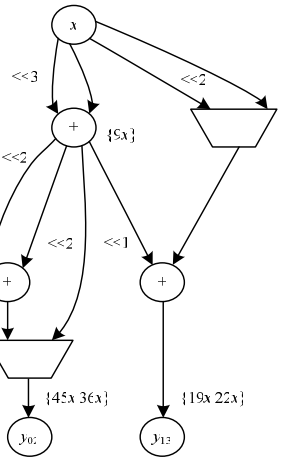


图 12 策略 2 第(3)步合并后的 DAG

3.2 MCM DAG 合并算法

本文对文献[6]提出的合并方法进行了改进，通过对各组常数的 MCM DAG 进行合并，生成 RMCM DAG。合并算法的关键性质是合并后的 DAG 加法器数目与所有输入的 DAG 中最大的加法器数目相等，不随 N 的增加而变化。

为了合并 N 个 DAG，先介绍合并 2 个 DAG 的基准算法(算法 2)。这个算法将在合并 N 个 DAG 的过程中作为子程序单元被迭代调用。把 2 个待合并 DAG 标记为 DAG_L 和 DAG_R，分别对应于与 2 组常数$\{c_{L1},c_{L2},\cdots,c_{LM}\},\{c_{R1},c_{R2},\cdots,c_{RM}\}$的乘法，分别有 n 和 m 个 A 运算节点，假设 $n\geqslant m$。节点集合标注如下：

$Nodes_L=\{Node_{L,0}, Node_{L,1},\cdots,Node_{L,n-1}\}$

$Nodes_R=\{Node_{R,0}, Node_{R,1},\cdots, Node_{R,m-1}\}$

合并算法尝试找到并利用 2 个 DAG 之间拓扑的相似性。相似区域被合并，允许 DAG_L 和 DAG_R 中的 A 运算使用同一个加法器实例，硬件开销很小；对于不相似的区域，加法器仍然可以被 DAG_L 和 DAG_R 中的 A 运算以不同配置使用，但必须插入多路选择器以连接正确的输入到共享的加法器或加法操作数的不同移位。

算法 2 合并 2 个 MCM DAG

输入 DAG_L 和 DAG_R(分别有 n 和 m 个 A 运算节点，且 $n\geqslant m$)

输出 估算面积最小的与$\{c_{L1},c_{L2},\cdots,c_{LM}\}$或$\{c_{R1},c_{R2},\cdots,c_{RM}\}$相乘的合并 DAG

```
FusePaiRMCMDAGs(DAG_L, DAG_R)
1.    best_aRea = ∞
2.    Rmcm_dag = niL
3.    foR aLL assignmentsφof DAG_R to DAG_L do
4.        dag = FusePaiR(DAG_L,DAG_R,φ)
5.        aRea = EstimateARea(dag)
6.        if aRea < best_aRea then
7.            best_aRea = aRea
8.            Rmcm_dag = dag
9.        end if
10.   end for
11.   RetuRn Rmcm_dag
```

算法首先枚举所有的 $Nodes_R$ 向 $Nodes_L$ 的分配。分配 DAG_R 中的每个节点 $Node_{R,i}$ 到 DAG_L 中唯一的节点 $Node_{L,j}$，每种分配都是一个一一映射φ: $Nodes_R\rightarrow Nodes_L$，$Node_{R,i}$ 和 $Node_{L,j}$ 共享合并 DAG 中的同一个加法器。当一种节点分配确定之后，函数 FusePaiR 从输入节点开始创建合并 DAG。考虑节点 $Node_{R,i}$和 $Node_{L,j}$的合并，每个节点有 2 个操作数(入边)，其中一个没有移位。如果 2 个节点的运算均为加法或均为减法，则合并后的节点的运算仍为加法或减法；如果 2 个节点的运算一个为加法一个为减法，则合并后的节点是加减法节点，通过控制端选择运算类型。FusePaiR 通过插入多路选择器来合并节点的入边，如果当前节点为加法运算，则根据加法交换律，可尝试交换节点的 2 条入边来减小面积，否则，交换律失效，FusePaiR 不会尝试交换边来减小面积。算法通过面积估算函数 EstimateARea 估算合并 DAG 的面积，选择估算面积最小的合并 DAG 返回。

将 DAG_L 推广到已合并 DAG 的情况，则可使用 FusePaiRMCMDags 进行 N 个 DAG 的合并。若 DAG_L 已是一个合并 DAG，则其中的节点之前可能已经有多路选择器。这种情况下的 A 运算节点分配和之前一样，入边的合并也类似，在一个有 v 选 1 多路选择器输入的合并节点情况下，一个新的多路选择器将被添加，成为 v+1 选 1 多路选择器。函数 FuseMCMDAGs 反复调用 FusePaiRMCMDags，可实现 N 个 DAG 的合并。

当多个 DAG 以不同的先后顺序进行合并时，合并顺序错误可能会导致选择非最优解。文献[6]给出了相关例子。为了避免出现这种问题，对这些 DAG 的合并顺序进行排列组合，从中选择最优的 DAG。

使用 $A_{\max}$ 标记待合并的 N 个 MCM DAG 中最大加法器数目，使用 $A(N)$标记合并后 DAG 的加法器数目，$A(N)=A_{\max}$。这是由于合并算法不添加任何加法器，只添加多路选择器，因此本文合并算法生成的 RMCM DAG 中加法器的最大数目等于输入 MCM 电路的最大加法器数目。

3.3 面积估算

面积估算函数 EstimateArea 遍历给定 DAG 并且计算每个多路选择器、加法器、减法器和加法/减法器需要的位宽 k，用 ak 估算面积，其中，a 是根据 ASIC 技术和工艺映射库决定的常数。本文将其映射到 0.13 μm ASIC 技术，使用 Synopsys Design Compiler 和商用 0.13 μm 标准单元库，对面积进行优化。常数 a 的估计值如下：

(1)对于 v 选 1 多路选择器，$a_{\text{mux}}=6v$；

(2)对于加法器，$a_{\text{add}}=33$；

(3)对于减法器，$a_{\text{sub}}=36$；

(4)对于加法/减法器，$a_{\text{addsub}}=46$。

整个 DAG 的面积代价 $Area$ 通过所有位宽与上述位宽常数 a 相乘并累加得到：

$$Area=\sum_{i=1}^{n_{\text{mux}}}a_{\text{mux}}\cdot bw_{\text{mux}_i}+\sum_{i=1}^{n_{\text{add}}}a_{\text{add}}\cdot bw_{\text{add}_i}+\sum_{i=1}^{n_{\text{sub}}}a_{\text{sub}}\cdot bw_{\text{sub}_i}+\sum_{i=1}^{n_{\text{addsub}}}a_{\text{addsub}}\cdot bw_{\text{addsub}_i}$$

在计算合并 DAG 的代价中，当前节点会选择以下 3 种结构进行合并，并得到不同的合并代价[6]：

(1)v 选 1 多路选择器

$bw_{\text{mux}}=\text{lb}(f_{\text{mux}})+n-s_{\text{mux}}$

其中，f_{mux} 是 v 个输入中最大输入值；s_{mux} 是最小移位数目。文献[6]的 SCM 合并算法不支持前一个节点的输出有右移的情况，s_{mux} 总是为正；而本文提出的合并算法支持右移，因此，在前一个节点的输出有右移的情况下，s_{mux} 为负，即右移会使 bw_{mux} 增大，从而增大估算面积。

(2)加法器

$bw_{\text{add}}=\max(\text{lb}(f_{\text{lmux}})+s_{\text{lmux}},\text{lb}(f_{\text{rmux}})+s_{\text{rmux}})+n-ax(s_{\text{lmux}},s_{\text{rmux}})$

其中，lmux 和 rmux 分别表示左和右前驱多路选择器；f 和 s 的计算方法与多路选择器中一样。

(3)减法器或加/减法器

$bw_{\text{add}}=\max(\text{lb}(f_{\text{lmux}})+s_{\text{lmux}},\text{lb}(f_{\text{rmux}})+s_{\text{rmux}})+n$

因为减法不能像加法一样硬连接这些最低位到 0，所以没有 $\max(s_{\text{lmux}},s_{\text{rmux}})$。

通过实验比较使用 EstimateArea 得到的估算面积和综合后面积，平均误差在所有情况下低于 10%。

4 实验结果及比较分析

根据上述 RMCM 生成算法，本文采用 C++编程语言实现了 RMCM 可重构多常数乘法器设计工具，该工具读取给定常数集合，输出可重构多常数乘法器的 DAG，以及与之对应的 Verilog 网表文件。利用生成的 Verilog 网表文件，在 Synopsys Design Complier 中进行综合，版本号为 D-2010.03-SP2，使用 SMIC 0.13 μm 标准单元库。

4.1 随机实验

假设定点输入 x 的位宽是 16，给定的常数集合的常数位宽是 16，选取 N=2,3,4 及 M=2,3,4，运行本文提出的 RMCM 生成算法，对于每对 N 和 M 运行 100 个随机产生的例子，得到实验结果。

表 1 是 2 种策略的估算面积与综合后面积的平均值，可以发现，策略 1 的误差为 6%~10%，策略 2 的误差为 1%~5%，策略 1 的误差较策略 2 大。这是由于策略 1 相比策略 2 而言，不同 DAG 中存在基数相同节点的可能性更小，因此，需要插入更多的多路选择器，而多路选择器的面积估算是造成误差增大的主要原因。总体来说，本文采取的面积估算函数计算结果与综合后的面积结果相比，误差在 10%以内。

表 1 2 种策略的估算面积与综合后面积对比

N	M	策略 1 的平均估算面积/μm²	策略 2 的平均估算面积/μm²	策略 1 的平均综合后面积/μm²	策略 2 的平均综合后面积/μm²	策略 1 的误差/(%)	策略 2 的误差/(%)
2	2	7 826.92	7 434.64	7 244.32	7 225.17	9.32	5.07
2	3	10 185.77	9 512.25	9 984.24	10 141.41	9.99	4.75
2	4	11 435.49	12 258.86	12 548.23	12 902.26	9.92	4.46
3	2	9 318.14	9 249.27	9 176.72	9 894.52	7.55	3.58
3	3	12 227.24	13 504.72	12 950.79	14 083.62	8.02	2.07
3	4	16 958.68	16 096.50	16 616.38	17 734.20	8.62	2.33
4	2	10 624.50	11 430.95	10 961.03	12 307.47	6.52	2.14
4	3	14 938.13	15 878.61	15 557.20	17 386.33	7.08	1.51
4	4	18 896.44	20 358.58	19 988.53	22 234.58	6.65	1.31

选取 2 种策略综合后面积较小的结果作为最终结果，从表 2 中可以发现，选取策略 1 的百分比大于选取策略 2 的百分比，并且随给定常数数目的增加而增加。这是因为在常数数目增大的情况下，策略 2 对全局进行优化，但在拆分后的 MCM DAG 中会有冗余出现，造成不必要的加法器增加，而图 8 给出的例子是因为常数数目少，即使拆分也与最优 MCM DAG 使用的加法器数目相同，所以在这种情况下策略 2 体现出了优势。

表 2 RMCM、MCM 和 RSCM 方法生成结果综合后面积对比

N	M	选取策略 1 的比例/(%)	选取后的综合后面积平均值/μm²	使用 MCM 构建 RMCM 的综合后面积平均值/μm²	使用 RSCM 构建 RMCM 的综合后面积平均值/μm²	相对 MCM 方法的面积节省率/(%)	相对 RSCM 方法的面积节省率/(%)
2	2	44	6 960.94	8 738.03	8 224.16	20.16	14.96
2	3	60	9 752.19	12 084.10	12 373.46	18.90	21.01
2	4	59	12 272.54	15 199.69	16 269.41	19.21	24.40
3	2	78	9 063.11	12 936.92	10 538.39	29.82	13.83
3	3	83	12 872.74	18 364.32	15 917.89	29.86	18.99
3	4	76	16 388.57	23 182.39	20 956.88	29.28	21.67
4	2	88	10 893.81	18 043.48	12 118.28	39.53	9.93
4	3	94	15 514.86	24 698.32	18 253.69	36.26	14.92
4	4	91	19 904.48	31 568.84	24 243.31	36.82	18.11

由表 2 可以看出，选取 2 种策略综合后面积较小的结果的面积平均值比 2 种策略分别综合后的面积平均值小，这是因为这里的平均值是对每个例子选取较小的面积之后进行平均计算的结果。使用本文提出的 RMCM 结构相比图 7(b)由 MCM 构建 RMCM 的方法，在面积上节省了 18%以上，常数数目越多，节省的面积越大；相比图 7(c)由 RSCM 构建 RMCM 的方法，在面积上节省 10%~20%。因此，本文提出的 RMCM 生成算法在面积上优势明显。

4.2 应用实例

对数字信号处理中常用的离散余弦变换用 RMCM 算法进行实现，并与不用 RMCM 算法的结果进行比较。

离散余弦变换的主要运算是矩阵运算[7]，一个 4 点反变换的矩阵如下所示：

$$\boldsymbol{C}_4=\begin{bmatrix} b & c & d & e \\ c & -e & -b & -d \\ d & -b & e & c \\ e & -d & c & -b \end{bmatrix}$$

则 4 点反变换过程可表示为：

$$\boldsymbol{Y}_4=\boldsymbol{C}_4\boldsymbol{X}_4$$

其中，$\boldsymbol{Y}_4=[y_0\ y_1\ y_2\ y_3]^{\mathrm{T}}$；$\boldsymbol{X}_4=[x_0\ x_1\ x_2\ x_3]^{\mathrm{T}}$。

由此可以看出，在矩阵相乘时需要用到多常数乘法的运算，而为了支持多个标准的多常数乘法，需要这些多常数在不同时刻可配置，这正是本文提出的 RMCM 生成算法适用的范围。

文献[8]实现了一个支持视频多标准线性反变换的结构，可以支持 MPEG-2/4 和 VC-1 这 2 种标准的矩阵相乘。MPEG-2/4 的 b、c、d、e 值分别为 502、426、284、100；VC-1 的 b、c、d、e 值分别为 16、15、9、4。该结构是通过观察数字得出的结果，因此，还有更大的优化空间。将其与 RMCM 算法生成的结构进行比较，结果如图 13、图 14 所示。在综合时将时钟周期约束设为 10 ns，两者性能的比较结果如表 3 所示。从结果可以看出，由于使用了更少的加法器和更少的多路选择器，因此 RMCM 算法与文献[8]的结果相比减小了 20%的面积。并且由于 RMCM 是自动化的设计方法，可以很快地应用于更多标准的变换系数上。

表 3 RMCM 与文献[8]结构的性能比较

结构	加法器数目	多路选择器数目	面积/门	时序/ns
文献[8]的结构	7	8	1 427	9.93
RMCM 结构	6	7	1 146	9.86

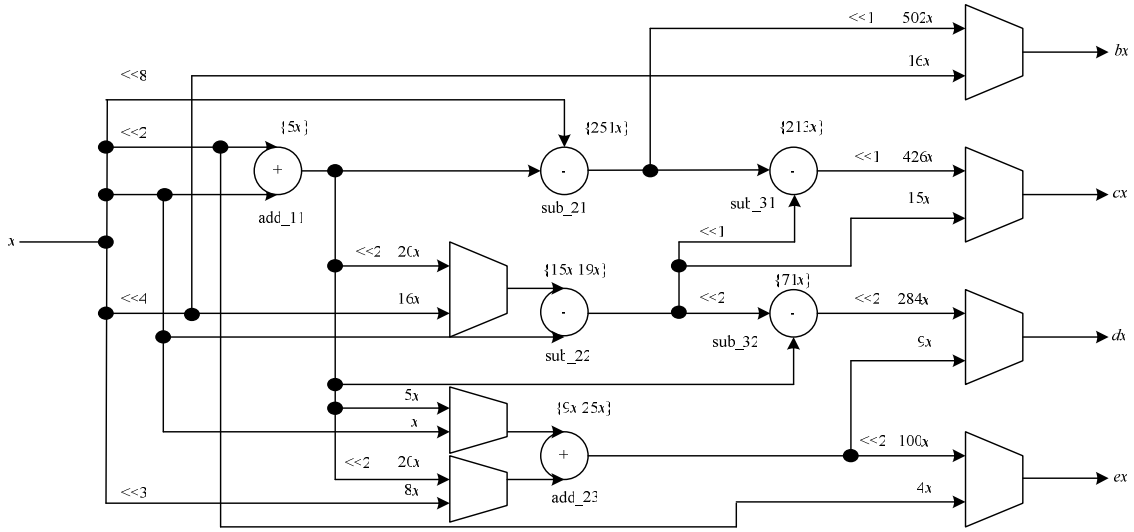


图 13　文献[8]的 4 点 DCT 反变换结构

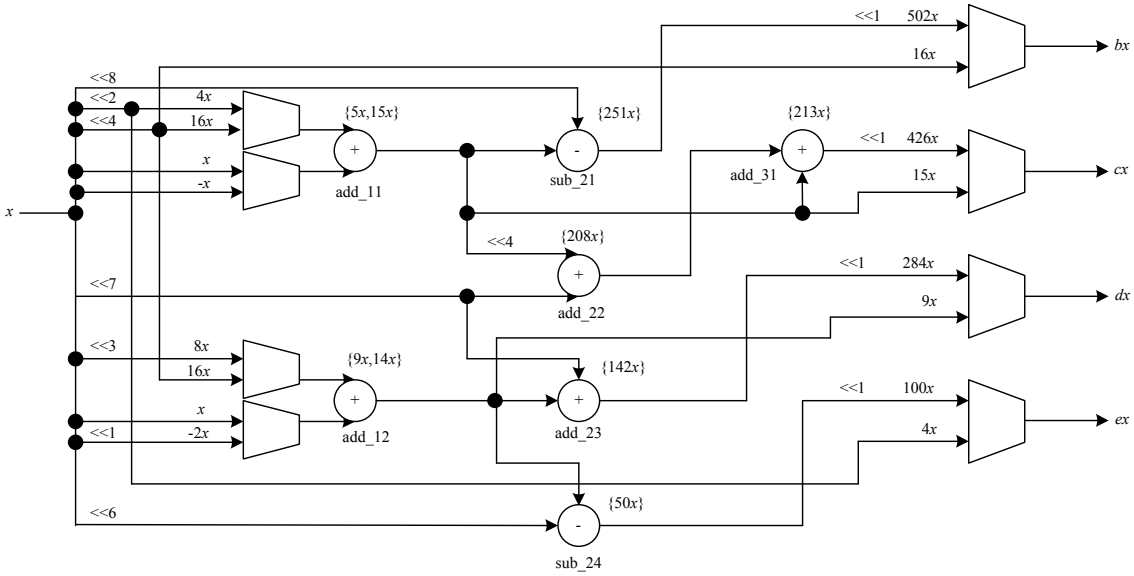


图 14　利用本文 RMCM 生成算法实现的 4 点 DCT 反变换结构

5　结束语

本文提出了一种针对可重构多常数乘法器问题的解决方法，通过 RMCM 生成算法，对输入常数集合先后进行 MCM DAG 生成和 MCM DAG 合并，最后输出面积最优的 RMCM DAG。通过输出的 Verilog 网表进行 Design Complier 综合，实验结果表明，本文提出的 RMCM 生成算法与原有的 MCM 和 RSCM 相比，面积节省得多。并且该算法可较好地用于解决视频多标准线性变换的问题，具有广泛的应用前景。

参考文献

[1] Cappello P, Steiglitz K. Some Complexity Issues in Digital Signal Processing[J]. IEEE Transactions on Acoustics, Speech, and Signal Processing, 1984, 32(5): 1037-1041.

[2] Koren I. Computer Arithmetic Algorithms[M]. 2nd ed. Natick, USA: A. K. Peters, 2001.

[3] Gustafsson O, Dempster A G, Wanhammar L. Extended Results for Minimum-adder Constant Integer Multipliers[C]//Proc. of IEEE International Symposium on Circuits and Systems. Scottsdale, Arizona, USA: IEEE Press, 2002.

[4] Voronenko Y, Puschel M. Multiplierless Multiple Constant Multiplier[J]. ACM Transactions on Algorithms, 2007, 3(2): 11-49.

[5] Dempster A G, MacLeod M D. Constant Integer Multiplier Using Minimum Adders[J]. IEE Proceedings of Circuits, Devices and Systems, 1994, 141(5): 407-413.

[6] Tummeltshammer P, Hoe J C, Puschel M. Time-multiplexed Multiple-constant Multiplier[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2007, 26(9): 1551-1563.

[7] 张国荣，印　鉴. 基于离散余弦变换矩阵的隐私数据保护方法[J]. 计算机工程, 2009, 35(2): 157-158.

[8] Qi Honggang, Huang Qingming, Gao Wen. A Low-cost Very Large Scale Integration Architecture for Multistandard Inverse Transform[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2010, 57(7): 551-555.

编辑　张　帆