算术电路的等价性验证

翁延玲 严晓浪 葛海通 郑飞君

（浙江大学 超大规模集成电路设计研究所，浙江 杭州 310027）

wengyl@vlsi.zju.edu.cn

摘要：本文提出了一种利用综合引擎重现算术电路的优化过程的算法，提高了等价性验证效率。ZDFV 的综合引擎首先识别乘法器的编码方式，提取部分积的加法树实现方案；然后根据这些信息生成与实现电路结构相似且逻辑正确的网表。本算法可直接结合到现有的 RTL 和门级网表的验证流程中，从而提高算术电路的验证能力。

关键词：综合、等价性验证、算术电路

1. 引言

RTL（寄存器传输级）电路(下文称参考电路)与门级网表电路(下文称实现电路)之间的等价验证框架包括 RTL 电路的综合引擎和门级网表的比较引擎。等价性验证大多数的研究重点都集中于比较引擎[1-3]，其中以布尔逻辑分析引擎进展尤为突出[4-7]，这些研究都假定参考电路的综合为直接的翻译过程。事实上，参考电路和实现电路的结构相似性制约着比较引擎的验证效果，若综合引擎能识别并重现 RTL 到门级网表的优化变换，就能使生成的参考电路的门级网表具有好的相似性。

数字电路中的算术电路一般都以加法器作为基本实现单元。基于时序和面积的要求，加法器可能采用不同的方案实现——CSA，CLA 或 wallance tree 等。加法器的基本实现逻辑是“异或”和“与”，大规模的加法器会引入大量“异或”运算，而“异或”逻辑的验证是等价性验证的瓶颈。本文以乘法电路为例，利用等价性验证工具 ZDFV 的综合引擎从实现电路中提取算术逻辑的实现方案，并加以验证。结果不仅证明其有效性，也说明了在综合阶段所作决策对最后生成的门级网表结构具有决定性影响。

2. 算法模型和定义

定义 1　DAG 是单向无环图，$G(V = PI \cup PO \cup IN, E)$。$PI$、$PO$、$IN$ 分别为原始输入、原始输出及内部节点的节点集，边集 E 描述节点间互连关系。

定义 2　BASE(F[i])为输出 F[i]的依赖输入变量集合。

定义 3　DIS(v_o, v_i)为节点 v_o到输入 v_i的距离，　大小为 v_o到 v_i的最短路径的长度。

定义 4　加法图 G(V,E)是有向无环图。V = S∪C∪I。I 为原始输入，C 为进位输出，S 为和输出。S 有且只有两个源点，而 C 无源点。

加法电路可由半加器组成。图 1（a）为半加器表示符号，（b）是一种全加器的实现逻辑，（c）

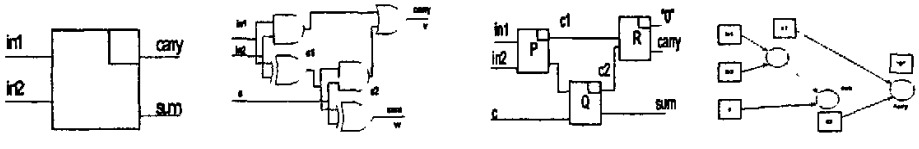

图 1 (a) 半加器符号　(b) 全加器逻辑　(c) 全加器符号　(d) 全加器的加法图

为其半加器符号图，（d）为其加法图。

2.1 距离计算

选定 out[i]构建电路的 DAG，如图 2 算法计算 *DIS*(*out*[*i*], BASE(out[i]))。*Gate* 为距离计算的起点，由 *gate* 回溯电路，计算其到相关的 *INT* 中元素的距离，记录最小值及对应路径。为抵消等价逻辑的不同实现对距离的影响，定义各类门的距离权值如下：

$$W(g)=\begin{cases}0 \text{ if } type \in \{inv,\ buf,\ PO\}\\ 1 \text{ if } type \in \{and, or, nand, nor\}\\ 2 \text{ if } type \in \{xor,\ nxor\}\end{cases}$$

各分支距离计算在遇到 *BASE*(*gate*)的元素时终止，距离计算在遍历 *BASE*(*gate*)中的元素后结束。

```
distance_recursion(gate)
{
  If(gate->type()==PI) return;
  else for_each_input(gate, gi) {
    if(DIS(out[i], gate) + W(gate) < DIS
(out[i], gi) {
   DIS (out[i], gi) = DIS(out[i], gate) +
W(gate);
    }
   distance_recursion(gi);
```

图 2 距离计算

2.3 加法图的提取[3]

提取加法图的算法如图 3 所示可分为三步：通过布尔逻辑推理得到原始电路中的 XOR 关系。本文基于 AIG 提取得到大部分 XOR 关系，由 SAT 和 BDD 辅助得到剩余 XOR 运算关系；根据已得 XOR 关系添加对应半加器，标记已匹配上的节点；由已得加法图继续匹配未被匹配半加器的区域，直至匹配完成。

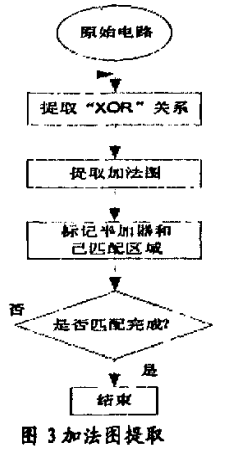

图 3 加法图提取

3. 电路特征提取

乘法器可分为部分积（partial product）产生电路与部分积的求和电路。其电路特征可由两部分组成：编码方式和加法树构架。

3.1 编码方式

图 4 为 4X4 的并行乘法器，A 是被乘数，B 是乘数。由图可得 out[1] = a1b0 + a0b1，out[2] = a2b0 +a1b1 + a0b2。BASE(out[2]) - BASE(out[1]) = {a2, b2}。 …… (1)

图 5 是布什编码（booth encoding）乘法。其第 k 位输出的计算方法如下：

Sum[k] = $\Sigma PP_{i(k-i)}$，k=0，2，4…… 且 k<n；

$PP_{ij} = (A_i+K_i)' = (W_i(a_j \cdot b_{i-1})' + W_i'(a_j \cdot b_{i-1})' + K_i)'$，$PP_{ij}$是第 i 行第 j 列的部分积；

$W_i = b_ib_{i-1} + b_i'b_{i-1}'$； $K_i = b_{i-1}b_ib_{i-1} + b_{i-1}'b_i'b_{i-1}'$；

sum[1] = pp01， BASE(sum[1])={b1, b0, a1, a0}；

sum[2] = pp02 + pp20，

BASE(sum[2])={b3, b2, b1, b0, a2, a1, a0}；

BASE(sum[2])-BASE(sum[1])={b3、b2、a2}。 …… (2)

对比（1）式和（2）式，布什编码的乘法器的 BASE 集合元素增量具有不对称性。因此，只需要提取低 3 位输出节点对应的 BASE 集合，计算其元素增量就可得该乘法器是否采用了布什编码。对于某

```
                    a3    a2    a1    a0
                    b2    b1    b0
              a3b0  a2b0  a1b0  a0b0
        a3b1  a2b1  a1b1  a0b1
  a3b2  a2b2  a1b2  a0b2
out[6] out[5] out[4] out[3] out[2] out[1] out[0]
```

图 4 并行乘法器

```
                          a3    a2    a1    a0
                          b3    b2    b1    b0
                         pp03  pp02  pp01  pp0
            pp13  pp12   pp11  pp100
sum6  sum5  sum4  sum3   sum2  sum1  sum
```

图 5 布什编码乘法器

些经过优化的布什电路，低几位的电路因重新优化可能不再具有该性质，那么只需要再继续搜索高位，亦可判断。

3.2 加法树构架提取

乘法器的加法树输出在组织结构上具有相似性，因此只需针对部分低位输出提取加法树。由半加器的位置和操作数的性质即可识别加法树的构架。

wallance tree 结构的加法树可由 CSA（斜进位加法器，carry-save adder）实现，最后一次输出终值的加法采用了带进位链的高速加法器。远离输出的 CSA 的操作数大多为部分积（加法图的 I），而靠近输出部分的 CSA 的操作数都基本是前级加法器的进位与和位（加法图的 S 和 C）。在阵列乘法器中其全加器的操作数除最后一次终值输出之外，操作数都包括了 I，S 和 C。从而可区分出乘法器的不同加法构架。本文实验中，提取了低六位输出的加法树就可区分出不同的加法树构架。

4. 验证框架

算术电路是等价性验证领域的瓶颈，商业工具为单独验证算术模块提供了黑匣子（blackbox）技术，用户通过其可设置算术电路的实现方案。当用户对算术模块的实现方案不了解时，本算法可在黑匣子技术基础上，直接结合到现有的 RTL 和门级网表的验证流程中。

在提取编码方式和加法树构架时，估算 m*n（m>n）的并行乘法器的算法复杂度：乘法器涉及到的加法器的数目为 O(mn)，mn 个部分积的位。搜索的输出为 3~6 位，涉及到的加法器个数为常数，并且这样的搜索是一次性的，因此该复杂度在实际的电路设计中可忽略。根据这些特征信息，综合引擎可生成与实现电路结构相似的网表，从而提高等价性验证的效率。

5. 实验结果

本算法在 ZDFV 的逻辑综合引擎上实现，分析实现电路，提取特征信息并生成与实现电路的具有相同实现方案的网表。测试平台是 Sun 公司的 Ultra-Sparc60 工作站，其配置为双 CPU，2G 内存。乘法器的实现有 CSA 阵列乘法器，Wallace tree 乘法器 。

表 1 是提取电路特征时间，消耗集中于加法图的提取，但该时间并不正比于电路规模。表 2 是 FORMALITY 的验证时间，实现电路是以对应实现方案实现的乘法电路，参考电路分别是阵列结构乘法器和本引擎相应实现的综合结果。当乘法器的位数达到 14 位时的网表文件，FORMALITY 运行时间过长，所以给出的电路规模只到 12 位。本算法还实验了各种类型的表达式和乘法结构，有符号的、无符号的及布什算法，结果类似。

表 1 提取电路特征时间

电路规模/bit	实现方案	运行时间/s
10	CSA	1.03
	WALL	1.20
12	CSA	1.05
	WALL	1.30

表 2 验证时间

电路规模/bit	实现方案	阵列结构	相似结构
10	CSA	16.57	15.03
	WALL	22.56	19.54
12	CSA	200.27	189.09
	WALL	156.92	112.44

由实验结果可得，参考电路以相似的结构实现时能加速验证，随着 ALU 规模增大，改善更明显。实验也再次证明了电路结构的相似性越大，所需验证时间将越短。

6. 结束语

在数字电路设计中，综合工具会根据面积时延等要求，对算术电路选择不同的实现方案，不同的实现架构之间的验证增加了等价性验证的复杂性。本算法尝试在综合阶段选择合适的实现架构，从而得到与实现电路具有较高相似性的网表。本文研究工作紧密结合工业界的电路实例，充分利用逻辑综合引擎在验证系统中的特殊位置，提出了针对算术电路的有效算法，明显提高了验证算法的效率。

参考文献

[1] A.Kiihlmann and F. Krohm. Equivalence checking using cuts and heaps[C]//Proceedings of the 42nd Conference on Design Automation. Anaheim, USA: ACM, 1997: 263-268.

[2] D.Brand Verification of large synthesized designs[C]//International Conference on Computer-Aided Design. San Jose, USA: ACM, 1993: 534-537.

[3] Dominik Stoffel and Wolfgang Kunz, Equivalence checking of arithmetic circuits on the arithmetic bit level [J], IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2004, 23(5): 586 – 597.

[4] Feng Lu, Li-C. Wang, Kwang-Ting Cheng, Ric C.-Y. Huang, A circuit SAT solver with signal correlation guided learning[C]// Design Automation and Test in Europe Conference and Exhibition. Munich, Germany: ACM, 2003: 892-897.

[5] F. Somenzi, CUDD: CU decision diagram package release 2.3.1[OL]. http:// vlsi.Colorado. edu/~fabio/CUDD/cuddIntro.html, 2001-2-16.

[6] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang and S. Malik Chaff: engineering an efficient SAT solver[C]// Design Automation Conference. Las Vegas, USA: ACM, 2001: 530-535.

[7] R. E. Bryant. Graph-based algorithms for Boolean function manipulation [J]. IEEE Transactions on Computers, 1986, C-35(8): 677-691.

Equivalence checking of arithmetic circuits

WENG Yan-ling, GE Hai-tong, YAN Xiao-lang, ZHENG Fei-jun

(Institute of VLSI Design, Zhejiang University, Hangzhou 310027, China)

Abstract: A new approach is proposed to improve the overall equivalence checking performance. The synthesis engine was made to replay the optimizations of arithmetic circuit and generate a gate netlist of great similarity. The synthesis engine recognizes different coding methods, and extracts the adder tree structure. With the extracted information, the RTL synthesis engine of ZDFV could generate a gate netlist that is logically correct and structurally similar to the implementation and thus speed up the equivalence checking process. The approach can be easily incorporated into existing RTL to gate equivalence checking frameworks and increase the robustness of equivalence checking for arithmetic circuits.

Key words: synthesis; equivalence checking; arithmetic circuit