结合二叉判决图和布尔可满足性的等价性验证算法

严晓浪, 郑飞君, 葛海通, 杨 军
(浙江大学超大规模集成电路设计研究所, 浙江杭州 310027)

摘 要: 本文提出了一种结合二叉判决图 BDD 和布尔可满足性 SAT 的新颖组合电路等价性验证技术. 算法是在与/非图 AIG 中进行推理, 并交替使用 BDD 扩展和基于电路 SAT 解算器简化电路. 如尚未解决, 将用基于合取范式 SAT 解算器进行推理. 与已有算法相比主要有如下改进: 在 AIG 中结合多种引擎进行简化, 不存在误判可能; 充分利用了基于电路解算器和基于合取范式解算器各自优点, 减小了 SAT 推理的搜索空间. 实验结果表明了本算法的有效性.

关键词: 等价性验证; 与/非图; 孤立节点; 二叉判决图; 可满足性解算器
中图分类号: TN4 文献标识码: A 文章编号: 0372-2112 (2004) 08-1232-03

Combining Binary Decision Diagrams and Boolean Satisfiability for Equivalence Checking

YAN Xiao-lang, ZHENG Fei-jun, GE Hai-tong, Yang Jun
(Institute of VLSI Design, Zhejiang University, Hangzhou, Zhejiang 310027, China)

Abstract: A new combinational equivalence checking approach integrating Binary Decision Diagrams and Boolean Satisfiability is proposed. The algorithm works on the representation called And/Inverter Graph of the circuit. The BDD propogation and Circuit-based SAT Solver are applied in an intertwined manner to reduce the space of the miter circuit. If failed, CNF-based SAT Solver is used to solve the problem. The efficiency of the proposed approach is shown through its application on the LGsynth91 benchmark circuits.

Key words: equivalence checking; AIG; isolated vertex; BDD; SAT solver

1 引言

组合电路的等价性验证问题 CEC(Combinational equivalence checking) 就是检验两个组合电路设计的功能是否等价[1]. 很多技术已被提出用于解决这个问题并被用于验证较大规模设计的正确性. 这些技术大致可归结为两大类: 功能性验证和结构性验证. 功能性方法通过构建两个待验证电路的有序二叉判决图 ROBDD(Ordered binary decision diagrams) 并检验两个 ROBDD 是否同构[2]. 结构性方法通过构建两个待验证电路的联接电路(Miter circuit), 使用 ATPG 等技术来证明联接电路的输出是否是 stuck-at-0 情况[3].

当前主流等价性验证方法大多数将结构性和功能性技术结合到同一框架里, 同时结合多种引擎进行验证. 常用引擎包括随机模拟、ATPG 技术、OBDD 等. 随着近年来提出的高效可满足性解算器(SAT Solver) 诸如 zChaff[4] 具有良好性能, 可满足性解算器也已成为重要验证引擎之一[5]. 通常而言, 两个待验证电路间存在结构相似性, 也就是说电路间存在大量等价节点, 如何识别这些等价节点并用于减小验证电路规模是一个研究热点. 目前常用简化方法是引入断点子集[6,7], 断点子集一般是指原始输入和等价节点的集合, 但由于断点子集选择不当易引起误判问题(False negative)[8], 结合误判消除常常反而需消耗更多的时间.

本文提出一种结合 BDD 扩展和 SAT 推理的等价性验证算法, 使用的引擎包括与/非图 AIG(And/Inverter graph)[9]、OBDD、基于电路和基于 CNF 的可满足性解算器. 本文余下部分组织如下: 首先介绍一些基本定义和定理, 第三部分介绍本文算法; 第四部分给出实验结果; 最后进行小结.

2 基本定义和定理

定义 1 AIG 是有向的非循环图 G(V= PI∪PO∪IN, E). 这里 PI、PO、IN 分别是指对应于原始输入、原始输出及内部节点的节点集, 边集 E 描述了节点间互连关系. 每个非输入节点 n∈V均对应与一个二输入与运算, 并在边上附带非运算.

图 1 是 z= AND(a, b) 对应 AIG 表征, 其中节点 a, b∈PI, z∈PO, 边(b, z) 加黑点表示非运算.

定义 2 FI_v 表示 AIG 中节点 v 的所有输入, 传输输入 TFI_v(Transitive fanins) 定义如下:

$$TFI_v = FI_v \cup \left(\bigcup_{u \in FI_v} TFI_u\right)$$

定义 3 对 AIG(V, E), 孤立节点 IV(Isolated vertex) 是指满足以下条件的节点 v:

对 ∀vc∈TFI_v 和 ∀vd | TFI_v, (vc, vd)∉E

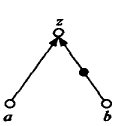

图 1 AIG 表征

定理、在等价性验证中, AIG 中孤立节点(不包括常数节点)可用新变量节点代替, 且无误判可能.

众所周知, 导致误判的原因是断点子集内至少两个断点是相关的, 也就是说当所有断点间不相交时, 误判问题就不会出现. 因而从孤立节点的定义, 将孤立节点用一新变量替代可简化验证任务, 且不会导致误判.

收稿日期: 2003-09-22; 修回日期: 2003-12-12
基金项目: 国家自然科学基金(No. 90207002)

推 理 对多输出 AIG, 如某节点在某个输出内是孤立节点(且非常数), 则在此输出中该节点可用新变量节点代替, 且无误判可能.

3 验证算法

本文提出等价性验证算法流程见图 2. 首先以 AIG 结构来构建联接电路, 在构建过程中使用哈希表来识别结构等价节点加以优化. 如果结构简化后联接电路输出不能得到常数, 则接下去将在 AIG 中交替地使用(1) BDD 扩展; (2) SAT 推理这两种技术来简化 AIG. 如果在最大限定条件内验证完成则结束, 反之则将 AIG 转化为 CNF 格式, 使用当前主流解算器 zChaff 进行推理直至结束.

311 AIG 结构简化

算法首先以 AIG 形式构建联接电路, 联接电路即将两个待验证电路输入共享, 对应输出用异或门连接. 构建 AIG 的伪代码见图 3. p_1, p_2 是待新建节点的两个输入节点, 在构建节点 p 前, 首先对特殊情况进行预处理, 如 p_1, p_2 为常数节点或互相等价等. 接下去则进行哈希查找, 判别待构建节点是否已存在. 如存在则直接合并同构节点, 反之则新建节点 p, 如果新建节点是非常数孤立节点, 则直接用新变量节点替代. 每个新建节点都将进入到一个哈希表中, 此哈希表使用节点 p 的输入 p_1, p_2 属性作为关键字.

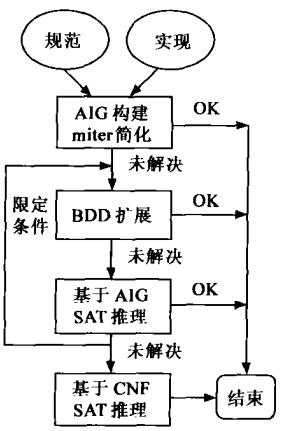

图 2 算法流程

```
Algorithm create_vertex(p1, p2){
  special case preprocess;
  if hash lookup(p1, p2) does not find vertex p{
    creat vertex p;
    if p is the isolated vertex
      replace p with a new variable vertex;
  add p to the hash table;
  }
  return p;
}
```

图 3 AIG 新节点构建伪代码

图 4 中是将电路转化为 AIG 结构一个简单例子, (a)中是两个待验证电路 y 和 z, 而(b)中是其对应联接电路 M 的 AIG 表征, 在构建节点 5 时, 因通过哈希表识别到节点 1 与节点 2 同构, 这样就直接将 1 和 2 合并, 又节点 1 是孤立节点, 则直接用新变量节点替代. 同理在构建过程中还可合并节点 5 和 3, 4 和 6.

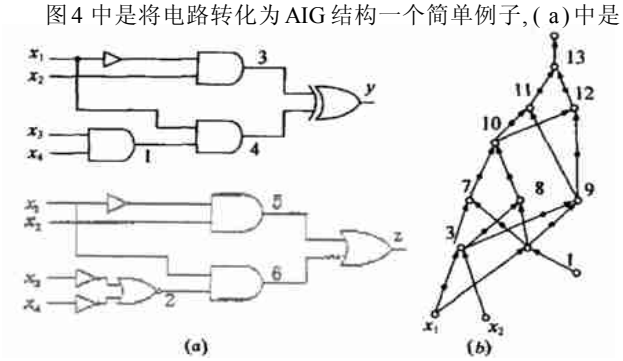

图 4 与/非图结构转化

312 二叉判决图扩展

在联接电路用 AIG 表示以后, 使用 BDD 扩展技术来识别内部顶点功能等价节点并加以进一步简化.

本文中 BDD 扩展伪代码见图 5. BDD 扩展由一个排序堆(Sorted heap)控制, 排序根据是对应节点的 BDD 大小. 首先是初始化堆, 即对每个原始输入, 构建其 BDD 并加入到堆中. 然后从堆中迭代地移除最小的 BDD, 并处理此 BDD 对应 AIG 中节点的那些扇出构建 BDD, 如果新的 BDD 能在限定大小内(算法中限定了最大构建 BDD 节点数) 被构建, 则加入到堆中. 在构建 BDD 过程中, 功能等价节点将被发现并得以合并, 并为孤立节点则直接用新变量节点代替. 如果未能解决问题, 以上操作将继续直至堆为空.

值得指出的是, 本算法中通过在 AIG 中识别孤立节点加以简化, 如前所述不存在误判可能. 而使用 BDD 扩展如未能解决问题, 将继续用 SAT 解算器进行推理.

```
Algorithm bdd_propogation(p){
  initilize the heap H;
  while heap H is not empty {
    extract the vertex Vwith the smallest
    BDD from H;
    for all fanout Vcof V{
      build BDD for vertex Vc with limited size;
      if Vc is the vertex p
        return equal or unequal;
      if find equivalent vertex Vd of Vc{
        merge Vc and Vd;
        if Vdis the isolated vertex;
          replace it with a new variable vertex;
      }
      put Vc to the heap H;
    }
    return undecided;
  }
```

图 5 BDD 扩展伪代码

313 基于电路的解算器和基于 CNF 的解算器

在用 SAT 进行推理时, 本算法中使用了两种类型可满足性解算器(SAT Solver), 即基于合取范式 CNF 解算器和基于电路解算器. 基于电路解算器易于同 BDD 扩展等其它技术结合进行交替验证推理, 同时能利用电路特有信息隐含进而减小搜索空间. 而基于 CNF 的解算器则在处理冲突分析(conflict analysis) 和子句学习(learned clauses) 中有优势.

基于电路解算器也是基于 DPLL 算法[11], 它试图发现一系列顶点赋值使得目标节点取值为逻辑/ 10, 如果赋值空间中不存在此类赋值则证明不可满足性. 解算器由初始化和推理证明两步组成: 初始化首先将目标顶点赋值为/ 10, 然后进行

隐含(implication)操作; 推理证明则处理顶点赋值和回溯情况.

在AIG结构中,将基于电路的解算器和BDD扩展交互进行推理. BDD引擎是从原始输入向原始输出构建BDD进行扩展,基于电路SAT引擎是从原始输出向原始输入方向进行推理. 每进行一次BDD扩展和SAT推理,构建BDD的上限大小和SAT解算器回溯值将递增. 同时在BDD扩展过程中得到简化结果将减小SAT搜索空间,而在SAT推理中也可得到常数节点简化AIG结构.

如果使用BDD扩展或基于电路SAT推理技术在限定条件(即规定构建BDD上限大小和SAT回溯最大值)验证成功则结束,反之则接下去将使用基于CNF解算器行推理. 首先我们将AIG结构转化为CNF公式,表1中给出AIG中四种基本类型与CNF的对应关系,并在下面给出证明. 这样,验证问题就转化为判断在目标顶点赋/10情况下电路对应CNF是否可满足. 如为UNSAT则证明两个待验证电路等价,反之则不等价. 值得指出的是,在用基于CNF解算器如zChaff进行推理时,之前用基于电路解算器进行推理结果将以冲突门(conflict gate)形式继承下来用来减少搜索空间.

表1　AIG中基本顶点对应CNF公式

门函数	CNF表征
z= AND(x, y)	(x+ y+ z)(x+ z)(y+ z)
z= AND(x, y)	(x+ y+ z)(x+ z)(y+ z)
z= AND(x, y)	(x+ y+ z)(x+ z)(y+ z)
z= AND(x, y)	(x+ y+ z)(x+ z)(y+ z)

证明

因 p= qZ(p2> q)(q- > p),

又有 py qZp+ q,

综上所得 p= qZ(p+ q)(p+ q);

同理得 z= AND(x, y)Z(z+ xy)(z+ $\overline{xy}$),

经转化即:

z= AND(x, y)Z(x+ y+ z)(x+ z)(y+ z)

其余几种情况类似可证.

4　实验结果

在基于CUDD和zChaff这两个软件包我们实现了以上算法. 所有实验结果在512 M内存的SUN ULTRA 10工作站上运行得到.

该算法使用实验电路是LGsynth91国际标准测试电路,实验是验证原始电路和综合后电路的等价性,在实验中限定构建BDD最大结点数5000,基于电路的SAT最大回溯次数为5000. 实验数据如表2所示,第一栏是被验证电路的名字,第二栏是电路的输出数,第三栏给出本文算法的验证CPU时间(秒). 从实验结果看,本文算法大多数情况中优于仅使用BDD扩展和SAT推理的技术,尤其是电路C3540, C5315和C7552等,部分输出用BDD扩展可较快验证,而部分输出需用SAT推理可很快得以验证.

表2　实验结果

电路名	输出数	验证时间
apex7	37	0.39
alu4	8	0.10
i8	81	0.98
rot	107	1.01
pair	137	1.42
des	245	3.12
dalu	16	0.28
C432	9	1.26
C880	25	0.31
C1908	25	1.60
C2670	140	1.10
C3540	22	9.83
C5315	123	1.86
C7552	108	2.10

5　小结

本文提出一种结合BDD扩展和SAT推理的CEC算法. 算法在AIG结构中交替使用BDD扩展和基于电路的SAT解算器推理以简化验证问题. 如在限定条件下验证尚未成功,将在已有搜索空间下用基于CNF的SAT进行推理直至结束. 在对LGsynth91部分测试电路的实验结果说明本算法能处理一大类CEC问题.

参考文献:

[1] Thomas Kropf. Introduction to Formal Hardware Verification[M]. Berlin, Germany: Springer Press, 1999.

[2] R E Bryant. Graph2based algorithms for boolean function manipulation[J]. IEEE Trans Computers, 1986, C235: 6772691.

[3] D Brand. Verification of large synthesized designs[A]. ICCAD[C]. San Jose, USA, 1993. 5342537.

[4] M Moskewicz, C Madigan, Y Zhao, L Zhang, S Malik. Chaff: Engineering an efficient SAT solver[A]. DAC[C]. Las Vegas, USA, 2001. 5302535.

[5] E Goldberg, M R Prasad, R K Brayton. Using SAT for combinational equivalence checking[A]. DATE[C]. Munich Germany, 2001. 1142121.

[6] Y Matsunaga. An efficient equivalence checker for combinational circuits[A]. DAC[C]. Las Vegas, USA, 1996. 6292634.

[7] Andreas Kuehlmann, Florian Krohm. Equivalence checking using cuts and heaps[A]. DAC[C]. Anaheim, USA, 1997. 2632268.

[8] C L Berman, L H Trevillyan. Functional comparison of logic designs for VLSI circuits[A]. ICCAD[C]. Santa, Clara, USA, 1989. 4562459.

[9] Andreas Kuehlmann, Viresh Paruthi, Florian Krohm, Malay K. Ganai. Robust boolean reasoning for equivalence checking and functional property verification[J]. IEEE Trans CAD, 2002, C221: 1377121394.

[10] Malay K Ganai, Lintao Zhang, Pranav Ashar, Aarti Gupta, Sharad Malik. Combining strengths of circuit2based and CNF2based algorithms for a high2performance SAT[A]. DAC[C]. New Orleans, USA, 2002. 7472750

[11] M Davis, G Logemann, D Loveland. A machine program for theorem proving[J]. Communications of the ACM, 1962, 5: 3942397.

作者简介:

严晓浪　男,1947年出生于杭州,浙江大学教授、博士生导师,主要从事电子设计自动化、系统级芯片设计等方面的科研和教育工作.

郑飞君　男,1980年出生于绍兴,浙江大学博士研究生,主要从事集成电路测试、形式化验证研究.