文章编号: 0427-7104(2006)01-102-05

基于切割法的时序电路等价验证

黄 伟,唐璞山
(复旦大学 专用集成电路与系统国家重点实验室,上海 200433)

摘 要: 在Van Eijk时序电路等价验证算法中引入切割法,提出一种改进算法.由切割法引发的错反问题同时得到解决,合理的切割可以使时序电路等价验证只需较少时间.改进算法用SAT解答器作为计算引擎.实验结果表明,改进算法的运行速度约为原先算法的2倍.

关键词: 半导体技术; 计算机辅助设计; 形式验证; SAT解答器
中图分类号: TN 402 **文献标识码**: A

时序电路等价验证是电子设计自动化领域中的难题.验证2个时序电路等价与否就是检验在所有可达状态,以及所有的输入向量激励下,2个时序电路对应的输出是否恒等.传统的解决方法是符号法(Symbolic Approaches)[1],它基于二元判定图(Binary Decision Diagram)计算电路的所有可达状态.该方法消耗大量内存,使得它只能验证规模较小的时序电路.

增量法(Incremental Approaches)是在组合电路等价验证中提出的,它基于寻找2个待验证电路结构上的相似性(Structural Similarity).实际工程应用中,有些电路优化只会对电路作出较小的改动,2个待验证电路在结构上有很大的相似性.这种基于电路结构相似性的验证方法也已经应用到时序电路等价验证领域[2].

在文献[2]中,Van Eijk提出了一种时序电路等价验证的算法.其核心思想就是寻找在所有可达状态下,产品机(2个待验证时序电路对应输入端相连接,对应输出端接入异或门而成的电路)内部的哪些节点总是和其他点具有相等值,这些点称为等价点,它们构成一个等价关系.算法首先计算0时刻满足初始状态 s_0 的等价关系 T_0,再根据 $i(i\geqslant 0)$时刻的等价关系 T_i 计算得到 $+1$ 时刻"改进"的等价关系 T_{i+1}.等价关系经过一系列的改进会到达一个固定点,即等价关系不能再被改进.文献[2]证明,最终的等价关系满足产品机的所有可达状态.

Van Eijk算法大多数时间耗费在比较节点布尔函数是否等价上.它的做法是把所有可能等价的节点配对逐次比较,每次比较都对配对的2个节点建立关于源输入和状态输入变量的布尔函数.可以看到,这种策略不是优化的.当电路的规模较大时,可以在已经被证明等价的节点上建立切割线,对配对的2个节点建立关于此切割线上节点变量的布尔函数,再进行比较,这样会使问题的规模变小.实验结果表明本文的改进算法能大大提高原先算法的速度.

1 时序电路等价验证算法

时序电路的模型是确定性米利型有限状态机(Deterministic Mealy-type Finite-state Machine),并且已知电路的初始状态.用六元组 $\boldsymbol{M}=(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{S},s_0,\Delta,\Lambda)$来表示,其中 $\boldsymbol{X}$ 代表输入向量空间; $\boldsymbol{Y}$ 代表输出向量空间; $\boldsymbol{S}$ 代表状态空间; s_0 代表状态机的初始状态; $\Delta:\boldsymbol{S}\times\boldsymbol{X}\rightarrow\boldsymbol{S}$ 代表状态传输方程; $\Lambda:\boldsymbol{S}\times\boldsymbol{X}\rightarrow\boldsymbol{Y}$ 代表输出方程.

定义1 产品机(Product Machine) 2个待验证时序电路 $\boldsymbol{M}_1=(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{S}_1,s_{1.0},\Delta_1,\Lambda_1)$和 $\boldsymbol{M}_2=(\boldsymbol{X},$

收稿日期: 2004-12-21
基金项目: 国家自然科学基金资助项目(90207002); 国家"八六三"计划资助项目(2002AA1Z1460)
作者简介: 黄 伟(1979—),男,硕士研究生;通讯联系人唐璞山教授.

$\boldsymbol{Y},\boldsymbol{S}_2,s_{2.0},\Delta_2,\Lambda_2)$,把对应输入端相连接,把每 2 个对应输出端分别连接二输入异或门输入,组合成为一个有限状态机 $\boldsymbol{M}=(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{S},s_{0.0},\Delta,\Lambda)$;其中 $\boldsymbol{S}=\boldsymbol{S}_1\times\boldsymbol{S}_2,s_{0.0}=s_{1.0}\times s_{2.0},\Delta(\boldsymbol{S}_1,\boldsymbol{S}_2,\boldsymbol{X})=(\Delta_1(\boldsymbol{S}_1,\boldsymbol{X}),\Delta_2(\boldsymbol{S}_2,\boldsymbol{X})),\Lambda(\boldsymbol{S}_1,\boldsymbol{X})=(\Lambda_1(\boldsymbol{S}_1,\boldsymbol{X})\oplus\Lambda_2(\boldsymbol{S}_2,\boldsymbol{X}))$.

假设 2 个时序电路 C_1 和 C_2,其有限状态机分别为 $\boldsymbol{M}_1$ 和 $\boldsymbol{M}_2$,产品机为 $\boldsymbol{M}$.由产品机的构成可以看出,$\boldsymbol{M}_1$ 和 $\boldsymbol{M}_2$ 等价就要求在产品机 $\boldsymbol{M}$ 的所有可达状态下,对所有输入向量激励,$\boldsymbol{M}_1$ 和 $\boldsymbol{M}_2$ 的对应输出都相等,即产品机的所有输出恒为 0.

定义 2　产品机时间帧模型　它是由 2 个产品机构成,前一个产品机的状态输出与后一个产品机的对应状态输入相连接.称前一产品机为"当前时间帧",另一产品机为"下一时间帧".在此模型中,产品机的每个内部节点 v 对应 2 个布尔函数,当前状态函数(current-state function)f_v 和下个状态函数(next-state function)δ_v,其中 $f_v:\boldsymbol{S}\times\boldsymbol{X}\rightarrow B$ 描述的是在当前时间帧中 v 点关于当前状态和当前源输入(primary inputs)的函数关系;$\delta_v:\boldsymbol{S}\times\boldsymbol{X}\times\boldsymbol{X}\rightarrow B$ 描述的是在下一时间帧中 v 点关于当前状态,当前源输入和下一源输入的函数关系.

定义 3　信号集合　信号集合 F 包含产品机内每个节点的当前状态函数 $f(s,x)$.另外,布尔常量 0 和 1 也被加入信号集合.

定义 4　状态兼容条件(State Compatibility Condition)在上述信号集合 F 基础上给定一个等价关系 T,状态兼容条件表示为如下函数,

$$Q_T(s,x)=\prod_{f_m,f_n\in F\wedge f_mTf_n} f_m(s,x)=f_n(s,x); \tag{1}$$

其中 f_mTf_n 表示节点 m 和 n 对应的当前状态函数 f_m 和 f_n 满足等价关系 T,即 f_m 和 f_n 在同一个 T 的某个等价类中.Q_T 表示状态 s 和某一输入 x 是否遵从等价关系 T,也就是在状态 s 下,对输入向量 x 的激励,信号集合中所有遵从等价关系 T 的 2 个信号具有相同的布尔取值.

Van Eijk 算法首先计算产品机初始状态 s_0 下的等价关系 T_0,

$$f_mT_0f_n\Leftrightarrow\forall x\in\boldsymbol{X}:f_m(s_0,x)=f_n(s_0,x); \tag{2}$$

T_0 描述了在初始状态 s_0 下,对于所有的输入向量激励,产品机中的哪些电路节点总是和其他点具有相等布尔取值.

在得到一个初始的等价关系 T_0 之后,根据下式进入改进等价关系的迭代之中,

$$f_mT_{i+1}f_n\Leftrightarrow f_mT_if_n\wedge(\forall s\in\boldsymbol{S},x_i,x_{i+1}\in\boldsymbol{X}:Q_{T_i}(s,x_i)=1\Rightarrow\delta_m(s,x_i,x_{i+1})=\delta_n(s,x_i,x_{i+1})); \tag{3}$$

上式描述的是根据 T_i 计算得到 T_{i+1}的数学表达式.它是基于产品机时间帧模型的运算.它表明节点 m 和 n 对应的当前状态函数满足等价关系 T_{i+1}必须具备如下 2 个条件:首先,节点 m 和 n 满足等价关系 T_i;其次,在任意当前状态和源输入以及下一时间帧源输入激励下,T_i 的状态兼容条件为真时,必然有节点 m 和 n 的下一状态函数相等.

上述迭代的终止条件是 $T_i=T_{i+1}$,即 T_i 不能再被改进.

文献[2]已经证明,最终的等价关系满足产品机的所有可达状态.检查原来待比较的 2 个时序电路每一组对应输出的当前状态函数是否在最终等价关系的同一等价类中.如果是,则表明电路等价.如果不是,则可以通过加入附加方程,增加产品机内部节点,重新计算等价关系.

2　本文提出的改进算法

2.1　电路验证中的切割思想

切割法是在组合电路验证中被提出的.在电路验证中,比较 2 个内部节点的函数关系是经常的操作.最直接的做法是对待比较的 2 个节点建立关于源输入的布尔函数,再比较它们.然而这种策略并非最优.如在图 1 所示的电路中,比较节点 m,n 是否等价(即对所有的源输入激励,具有相同的布尔取值).a,b,c 和 d 为电路的源输入端,建立 m 点关于源输入的布尔函数会涉及到其支持点集对应的所有逻辑门,即区域 acm 中的所有逻辑门(此处的三角区域代表一种大概的区域分化,表明逻辑门的覆盖范围).支持点集

的定义如下.

定义5 支持点集(Support Points) 电路内部某节点的支持点集包括该节点的扇入节点,以及这些扇入节点的支持点集.

该定义是一个递归定义.它表明电路节点根据其在电路中所处深度(离源输入节点越远,深度越深;反之越浅)的不同,其支持点集所包含的节点数目也会不同:深度越深,其支持点集越大;反之越小.

图1中 n 点涉及区域 bdn 中的所有逻辑门.如果在电路内部作出如虚线所示的切割线,对节点 m 和 n 建立关于此切割线上节点对应变量的布尔函数,分别涉及到区域 egm 和区域 fhn 的所有逻辑门.显然,建立关于切割线上节点变量的布尔函数会涉及较少逻辑门,布尔函数的规模会变小,节点的比较会变得简单.

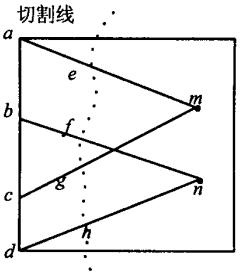

图1 电路中的切割技术

Fig.1 Cut points technique in a circuit

切割线并非随意建立.一般首先分析已经比较完毕,且已经被证明等价的节点配对,再挑选一批与待比较的2个节点距离相当,且在其支持点集上的节点配对,建立贯穿它们的切割线.切割线上等价的节点配对会被当作同一个节点来建立待比较节点的布尔函数.然而在切割线上完成节点布尔函数的比较时会不可避免地遇到错反问题[4].

定义6 错反问题(False Negative Problem) 配对节点对应的2个布尔函数事实上是相等的,但是由于选择内部切割线作为伪输入激励,节点之间的逻辑约束被忽略,导致布尔函数不等价.

比较图2(a)和图2(b)所示的电路.明显地,$s_1=s_2$,$t_1=t_2$,当在 s_1,t_1,s_2 和 t_2 处作切割时,电路可合并为图2(c).在图2(c)中,激励矢量 $s_1=1$,$t_1=1$ 使得 y_1,y_2 不等价.而事实上,这样的一个激励矢量不可能由源输入 i_1,i_2 和 i_3 的某个激励产生得到.

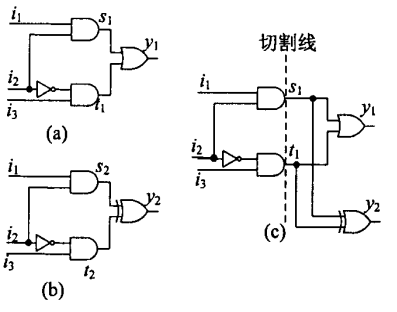

图2 错反问题例子

Fig.2 An example of false negative

错反问题是利用切割技术必须要解决的问题,解决错反问题的方法已比较成熟.该方法把切割线往源输入端回溯,建立一个新的切割线,争取包含更多的逻辑约束.2个节点的不等价要回溯到源输入端才会被确认,这种无休止的回溯往往会带来较多的时间消耗,是要避免的.因此通常会引入一个阈值来控制切割线的回溯,当超过此阈值时,回溯就不会再逐次进行下去,而把最终切割线直接建立在源输入上,即不引入内部切割线.

2.2 切割法在Van Eijk算法中的实现

分析Van Eijk算法可知,算法的关键步骤是计算等价关系的改进,它对应(3)式.(3)式是在任意 $s\in \boldsymbol{S}$,任意 $x_i,x_{i+1}\in \boldsymbol{X}$ 条件下,等价关系 T_i 的状态兼容条件为真时,完成"下一时间帧"中节点配对对应的下一状态函数的比较.在此可以引入切割法加速这种比较.引入切割法后,计算 T_{i+1} 的算法流程如图3所示.

正如前文所述的那样,提出的改进算法是在电路中建立切割线,把对配对节点函数的比较建立在其支持点集中节点比较的基础上.这样,算法必须先比较离输入较近的节点,再比较离输入较远的节点.当取到一个离源输入最近,且没有比较过的节点配对 np 时,算法会根据此配对在其支持点集寻找切割线 cf,在此切割线上完成配对点函数的比较.如果2个函数不等价并且切割线 cf 没有超过一定的阈值,则回溯切割线重新比较;如果2个函数相等,则返回并寻找新的未比较的节点配对;如果2个函数不相等且切割线已超过一定的阈值,则按照(3)式比较它们,并把比较结果保存在数据结构 cr 中.当所有的节点配对都比较完毕,根据 T_i 和 cr,形成新的等价关系 T_{i+1}.

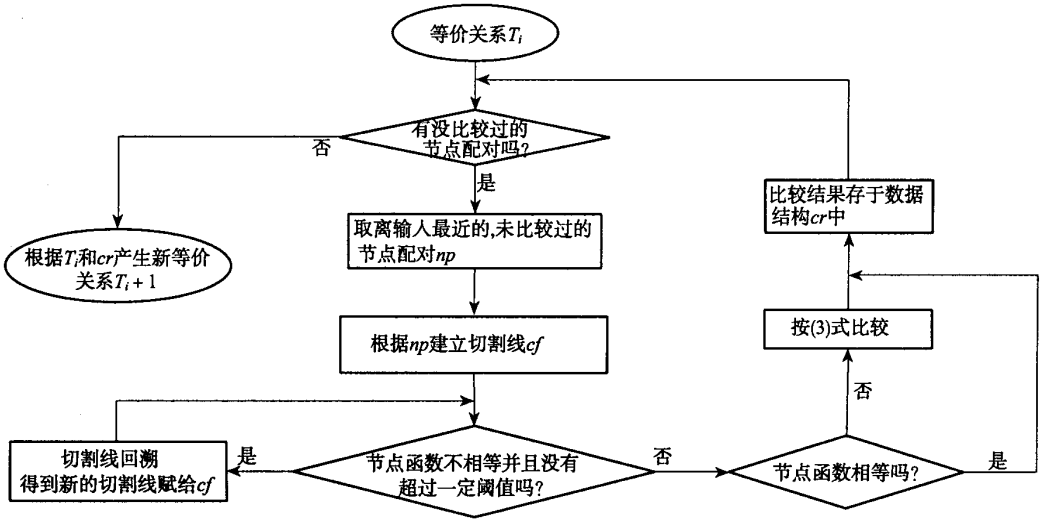

图 3　T_{i+1}计算流程

Fig. 3　Flow of calculating T_{i+1}

3　实验结果和结论

这部分给出本算法用 ISCAS'89 时序电路例子来做电路验证的结果. 基于上述的改进算法,在 SIS (Sentovich E M, Singh K J, Lavagno L, *et al*. SIS: A System for Sequential Circuit Synthesi . Technical report, University of California at Berkeley, 1992, Memorandum No. UCB/ERL M92/41.)环境下采用 C 语言实现了一个原形工具 FDSEC. 用 Zchaff SAT 解答器(http://www.princeton.edu/~chaff/zchaff.html.)作为此算法的引擎. 程序在主频为 450 MHz 的 Blade 1000 Sun 工作站上运行,物理内存 2 GB. 表 1 列出了实验数据,实验是对 SIS 的 fx 命令优化前后的时序电路进行等价验证. 表格的第 1 列是 ISCAS'89 时序电路例子名称;第 2 列给出计算最终等价关系的迭代次数;第 3 列给出用 SAT 解答器引擎,不用切割法时,Van Eijk 算法的运行时间;第 4 列为本文算法(即使用 SAT 解答器作为引擎,并使用切割法的算法)的运行时间;最后一列给出第 3,4 列比较,切割技术给 Van Eijk 算法带来的速度提高倍数.

表 1　关于 ISCAS'89 基准电路的试验结果

Tab. 1　Experimental results on ISCAS'89 benchmarks

电路	迭代次数	$t_{不切割}$/s	$t_{切割}$/s	提高倍数	电路	迭代次数	$t_{不切割}$/s	$t_{切割}$/s	提高倍数
S298	3	0.31	0.28	1.11	S967	4	5.63	3.02	1.86
S344	4	0.86	0.58	1.48	S1196	2	8.52	6.60	1.29
S349	2	0.40	0.21	1.90	S1238	2	14.18	12.48	1.34
S382	8	1.32	1.12	1.18	S1269	6	35.28	14.80	2.38
S400	9	1.64	1.38	1.19	S1423	13	103.88	49.66	2.09
S420	29	8.38	6.96	1.20	S1494	5	29.50	11.58	2.55
S499	1	0.38	0.24	1.58	S1512	15	42.09	29.26	1.44
S510	14	6.45	4.09	1.58	S3271	2	43.04	17.51	2.46
S526	11	3.11	2.80	1.11	S3330	3	72.03	65.13	1.11
S641	3	0.94	0.81	1.16	S4863	2	90.91	34.14	2.66
S838	69	104.36	73.55	1.42					

3.1　切割线回溯的控制

大量试验表明,为了提高验证效率,基于产品机时间帧模型对配对的节点作比较时,切割线只在下一

时间帧中建立,而不回溯到当前时间帧中;另外,当遇到错反问题时,切割线最多只回溯5次,即5次为回溯的阈值.

3.2 切割法和非切割法的比较

从表1最后一列中可以看出,把切割法运用到Van Eijk算法中可以平均提高其运行速度约2倍.除了s635之外,切割法给其他所有电路验证都带来速度的提高,特别是对s991的验证要比原先运行速度提高达5.12倍.足以见得本文提出的改进算法要明显优于原先的算法.

本文把切割技术运用到时序电路等价验证中,并在SIS环境下用C语言开发了一个原形工具FD-SEC.实验结果表明,改进算法的运行速度约为原先算法的2倍.由于切割法是基于2个电路相似性的方法,当待验证的2个时序电路结构非常相似(如经过组合优化前后的时序电路)时,本文提出的改进算法会大大加速电路验证的速度;但是当电路结构相似处较少(如经过多种优化的前后时序电路)时,切割法有时不但不能提高验证的速度,切割线的频繁回溯反而会成为运算的负担.所以,未来的研究方向就是把切割法和"电路转化"(一种把待验证的2个电路转换得更相似的方法)结合起来,提高对结构具有较少相似性的2个时序电路进行等价验证的速度.

参考文献:

[1] Burch J, Clark E, McMillan K, *et al*. Symbolic model check for sequential circuit verification [J]. *IEEE. Trans On Computer-Aided Design*, 1994, **13**(4): 401-424.

[2] Van Eijk C A J. Sequential equivalence checking based on structural similarities [J]. *IEEE Trans on Computer-Aided Design of Integrated Circuits and Systems*, 2000, **19**(7): 814-819.

[3] Huang S Y, Cheng K T, Chen K C. AQUILA: an equivalence verifier for large sequential circuits [EB/OL]. http://ieeexplore.ieee.org/iel3/4762/13192/00600302.pdf, 2000-08-04/2004-12-15.

[4] Kuehlmann A, Krohm F. Equivalence checking using cuts and heaps [EB/OL]. http://ieeexplore.ieee.org/iel3/4655/13048/00597155.pdf, 2004-02-26/2004-12-15.

Sequential Equivalence Check Using Cuts

HUANG Wei, TANG Pu-shan

(*ASIC & System State Key Laboratory, Fudan University, Shanghai* 200433, *China*)

Abstract: An improved algorithm which combines cut points technique with Van Eijk's sequential equivalence checking algorithm is presented. The false negative problem caused by the cut technique is solved, and reasonable cut results in less time for sequential equivalence checking. The calculating engine of the proposed algorithm is SAT-Solver. Experimental results show that the proposed algorithm can double the original speed.

Keywords: semiconductor technology; computer-aided design; formal verification; SAT-solver