

电路并发性质在 GSTE 中的研究和实现

郑德生, 杨国武, 徐忠林

(电子科技大学 计算机科学与工程学院, 四川 成都 611731)

摘 要: 由于 GSTE 的性质描述断言图的时序特性, 使其不能够简洁易懂地描述电路的并发性质, 因此将 meet 运算符引入 GSTE, 重新定义 GSTE 断言图及其验证算法, 使之能更简洁的描述和验证数字电路的并发性质, 实验表明引入 meet 运算符后的算法能够降低验证的复杂度.

关键词: GSTE; 形式化验证; 并发性质; 模型检验; 断言图

中图分类号: TN 407

文献标识码: A

文章编号: 1000-7180(2013)06-0043-05

The Implementation and Research of Circuit's Concurrent Properties in GSTE

ZHENG De-sheng, YANG Guo-wu, XU Zhong-lin

(School of Computer Science and Engineering, University of Electronic
Science & Technology of China, Chengdu 611731, China)

Abstract: This paper extends the GSTE with a new a new operator "meet". We modify the assertion graph and the algorithm of GSTE to make it can succinctly describe and verify the concurrent properties of digital circuits. The experiment result express that the new method can reduce the complexity of verification.

Key words: GSTE; formal verification; concurrent property; model checking; assertion graph

1 引言

数字集成电路是典型的并发系统, 如何实现对并发行为的有效验证就成为保证数字电路功能正确性的关键因素^[1-2].

GSTE^[3-5]是对 STE (符号轨迹赋值) 的进一步拓展, 它解决了 STE 存在的缺陷, 并允许用户根据需要在表达能力和时间效率、完备性和空间效率之间取得一个良好的平衡. 最重要的是它具有传统模型检验一样的表达能力. 由于它处理状态爆炸的独特方法, GSTE 在某些电路的验证已经超过了传统模型检验. 在 Intel, 它已被广泛应用于含有数以万计状态的超大规模集成电路设计的验证中^[6]. 在 GSTE 中, 电路的性质是用一种称为断言图的描述语言来描述的. 断言图是一个带有初始顶点的有向

图, 它的每一条边上有两个标签条件, 分别称为前序词和后序词. 断言图是一种基于标签所标识的动作的性质描述语言.

既然 GSTE 的验证规范描述的是基于动作的时序系统, 那么如何基于动作描述并发系统将是解决数字电路中并发行为验证的关键. 而进程代数^[7]是一种优良的、基于动作的且支持并发系统描述的数学语言. 进程代数提供丰富的操作算子, 它可以通过组合化的方式来支持并发系统的描述, 即可以将一个大的系统分解为若干小的系统以求简化设计过程, 也可以根据需要将小的系统组合成较大的系统以便观察系统的整体行为. 进程代数能够同时描述刻画时序系统和并发系统. 因此, 将进程代数引入到 GSTE 的性质描述语言, 并修改相应的 GSTE 算法, 将为 GSTE 验证电路的并发行为提供一个全新的解决思路.

2 断言语言

本中介绍 GSTE 中断言语言的相关定义. 域是元素的非空有穷的集合, 通常用 D 表示, 域中的元素称为符号. 给定域 D , 由域中的符号组成的任何有穷序列称为符号串或迹, 如 001110 是域 $D = \{0, 1\}$ 上的迹. 在电路验证中, 域 D 表示电路的所有状态的集合, 迹就是任意一个简单状态的无穷序列.

定义 1 (断言字母表) 给定域 D , D 的断言字母表 Σ 为一个二元组 $\Sigma = P(D) \times P(D)$, 其中 $P(D)$ 为 D 的幂集, 即 $P(D)$ 为域 D 的所有子集, 包括全集和空集.

定义 2 (断言字母) 给定断言字母表 Σ , 断言字母表 Σ 的任意一个元素被称为断言字母.

定义 3 (断言字) 给定断言字母表 Σ , Σ^* 的任意一个元素称为 Σ 的断言字. 其中, Σ^* 表示 Σ 上的所有的有穷长的断言字母串的集合.

定义 4 (断言语言) 给定断言字母表 Σ , $P(\Sigma^*)$ 的任意一个元素称为断言语言.

定义 5 (字满足) 迹 τ 满足 $P(D)$ 上的字 π , 记作 $\tau \models \pi$, 当且仅当 $\forall 1 \leq i \leq |\pi|, \tau[i] \in \pi[i]$.

定义 6 (断言字满足) 迹 τ 满足 Σ 上的断言字 w , 当且仅当 $\tau \models \text{ant}(w) \Rightarrow \tau \models \text{cons}(w)$.

定义 7 (断言字迹语言) 断言字 w 的迹语言, 记为 $\Omega(w)$, 是满足断言字 w 的所有迹的集合. 即 $\Omega(w) = \{\tau \in D^\omega \mid \tau \models w\}$.

给定断言字母 $\sigma = (A, B)$, 则 A 和 B 分别称为断言字母的前序词和后序词, 分别记作: $\text{ant}(\sigma) = A$ 和 $\text{cons}(\sigma) = B$. 前序词和后序词函数可以相应的拓展到 Σ^* 中的断言字. 例如, 给定断言字 w , 对 $\forall 1 \leq i < |w|$, 有 $\text{ant}(w)[i] = \text{ant}(w[i])$ 和 $\text{cons}(w)[i] = \text{cons}(w[i])$.

从形式上来, 断言语言是一个可操作的规范公式集合. 语言中的每一个断言字同时提供了模拟系统的激励以及期望得到响应. 有时需要非常多的甚至是无穷的断言字才能刻画系统的所有可能行为. 传统的 GSTE 定义的断言图是用带标签的有向图来表示这些断言字的一种方式. 其实, 传统的 GSTE 定义的断言图是一种比较特殊的正规自动机, 这里所描述的断言语言都能利用这种自动机产生.

3 并发规范

上面的断言语言和传统的 GSTE 一样, 并不能很好的描述电路的并发性. 为了更加简洁地描述

电路的并发性, 我们引入 meet 操作符 $\Pi: \Sigma \times \Sigma \rightarrow \Sigma$ 拓展传统的 GSTE 的断言语言, 如下, 定义两个断言字母的 meet 运算.

定义 8 给定断言字母 σ_1 和 σ_2 , 有:

$$\text{ant}(\sigma_1 \Pi \sigma_2) = \text{ant}(\sigma_1) \cap \text{ant}(\sigma_2)$$

$$\text{cons}(\sigma_1 \Pi \sigma_2) = \text{cons}(\sigma_1) \cap \text{cons}(\sigma_2).$$

对断言字母 meet 运算, 可以相应的拓展到两个长度相等的断言字, 其定义如下:

定义 9 给定断言字 $w_1, w_2 \in \Sigma^*$, 有:

$$w_1 \Pi w_2 =$$

$$\begin{cases} \epsilon & \text{若 } w_1 = w_2 = \epsilon \\ (w_1 \Pi w_2) \cdot (\sigma_1 \Pi \sigma_2) & \text{若 } w_1 = w'_1 \cdot \sigma_1 \text{ 且 } w_2 = w'_2 \cdot \sigma_2 \end{cases}$$

式中, $\cdot$ 为语言的串联操作符. 很容易看出 meet 操作符满足结合律、交换律和幂等律. 最后, meet 运算操作符可以同理拓展到两个断言语言.

定义 10 给定断言语言 $L_1, L_2 \in P(\Sigma^*)$, 有

$$L_1 \Pi L_2 = \{w_1 \Pi w_2 \mid w_1 \in L_1, w_2 \in L_2, |w_1| = |w_2|\}$$

目前, 在模型检测中已有不少用组合规范的形式来验证电路的并发性^[8-9]. 本节定义描述系统并发性质的规范语言, 我们称其为并发规范 (concurrent specification). 并发规范为一组断言语言集合上的代数方程的集合. 具体如下:

定义 11 (并发规范)

$$CS = \{L_0\} \cup \{L_1, \dots, L_{h-1}\} \cup \{L_h, \dots, L_{l-1}\} \cup \{L_l, \dots, L_{n-1}\}, \text{ 每一个代数方程如下:}$$

(1) (初始化) $L_0 = \epsilon \cup L_0 \cdot \sigma_0$, 其中: $\sigma_0 = (D, D)$;

(2) (前缀) $L_i = L_j \cdot \sigma_j$, 对于每一个 $1 \leq i < h$, 其中 $0 \leq j < n$ 且 $\sigma_j \in \Sigma$;

(3) (求和) $L_i = L_{i_1} \cup \dots \cup L_{i_k}$, 对于每一个 $h \leq i < l$, 其中 $1 \leq i_j < h$ 且 $1 \leq j \leq k$;

(4) (meet) $L_i = L_{i_1} \Pi \dots \Pi L_{i_k}$, 对于每一个 $l \leq i < n$, 其中 $q \leq i_j < l$ 且 $1 \leq j \leq k$.

在上面的定义中, ϵ 表示只包含空字 $\{\epsilon\}$ 的语言. $\cdot$ 表示将一个字母连接到语言的一个断言字的后面.

并发规范定义的风格与 Milner 的通信系统演算 CCS 理论^[10] 类似, 但是有三点不同: (1) 并发规范有一个特殊的初始化方程; (2) 并发规范的 meet 运算符替代了 CCS 的并行组合运算符“ $|$ ”; (3) 断言语言的迹语义. 并发规范是 GSTE 断言图的拓展. 实际上, 如果没有 meet 组合操作, 并发规范和

GSTE 断言图相对应,其中:(1)初始化语言对应于断言图的初始顶点加上一个自循环;(2)前缀语言对应于断言图的边。(3)求和语言对应于断言图的顶点。

下面以投票机系统的例子来说明上面的一些概念。如图 1 所示是一个投票机系统模型电路图。该投票机系统(Voting Machine, VM)有三个投票口组成。每个投票口通过状态位 $av[i]$ ($1 \leq i \leq 3$) 来表明该投票站是否可用,通过数据输入端口 $vin[i]$ ($1 \leq i \leq 3$) 接受单独的投票。一旦所有投票站都接收到它们的投票,投票机系统进行将对每个投票口进行计算,然输出计算结果。投票机系统的 $reset$ 接口是一个重置端口,它可以将投票机系统设置为初始状态,当 $reset$ 被置位时,所有投票口 $av[i]$ ($1 \leq i \leq 3$) 的重新设置为有效,表明相应的 $vin[i]$ ($1 \leq i \leq 3$) 可用接收投票。

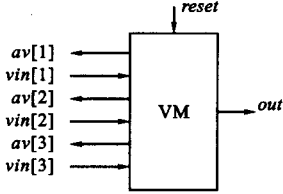


图 1 投票机系统模型图

注意,如图 1 所示的投票机系统模型并没有说明系统是如何实现的。这并不影响对系统性质的刻画,因为基于组合的方法可以不考虑具体的实现细节,从而比较简洁的描述系统的性质。图 1 所示的投票机系统模型的功能规范包括可以有下面一系列并发规范来描述。为简明起见,有些代数公式被简写为语言和正规表达式的混合形式。

(1)(Ready)当投票机系统被 $reset$ 或者 $polled$ 后,投票口 i ($1 \leq i \leq 3$) 处于准备状态。

$Ready[i] = ((true, true) * \cdot (reset, av[i]) \cup Poll \cdot (reset \vee \neg vin[i], av[i])) \cdot (reset \vee \neg vin[i], av[i])^*$

(2)(Voting)投票口 i ($1 \leq i \leq 3$) 正在接收投票。

$Voting[i] = Ready[i] \cdot (\neg reset \wedge vin[i], av[i])$

(3)(Voted)投票口 i ($1 \leq i \leq 3$) 已经接收了投票。

$Voted[i] = Voting[i] \cup (Voted[i] \parallel Wait) \cdot (\neg reset, \neg av[i])$

(4)(Wait)至少有一个投票口处于等待状态。

$Wait = \bigcup_{i=1}^3 Ready[i]$

(5)(Poll)三个投票口处于 Voted 状态,其中一个投票口正接收一个投票。

$Poll = (\prod_{i=1}^3 Voted[i]) \parallel (\bigcup_{i=1}^3 Voting[i])$

(6)(Output)投票机系统输出 polling 结果。

$Output = Wait \cdot (true, \neg vout) \cup Poll \cdot (true, vout)$

上面的性质规范并不是简单独立性质的交集。它是相互依赖的性质组成的语义网络。这些方程的集合能够用代数形式来定义。

4 并发规范模型检验算法

虽然可以利用对语言 meet 极限构建正规自动机的方式来模型检验用并发规范方式编写的性质,但是这个构建过程很可能导致规范大小的指数级增长。为了避免这个问题,本节修改后设计的算法能够深入规范语言的语法结构并建立从规范的语言元素到电路的状态集合的模拟关系。

定义 12 (模型)给定域 D , D 上的模型是一个三元组 $M = (S, R, L)$, 其中:

(1) S 是一个有穷状态集合;

(2) $R \subseteq S \times S$ 是 S 上的状态转移关系,对于每一个状态 $s \in S$ 都存在一个状态 $s' \in S$ 满足 $(s, s') \in R$;

(3) $L: S \rightarrow D$ 是一个标签函数,其给每一个状态 $s \in S$ 标签了一个值 $d \in D$ 。

由 M 可以得到一个单调的后像转移函数 $post_M: P(S) \rightarrow P(S): post_M(S) = \{s' \mid \exists s \in S, (s, s') \in R\}$, 其中 $S \in P(S)$ 。

模型 M 是没有初始状态的一个 Kripke 结构。为和 STE 一致,我们假设模型 M 的每一个状态都是初始状态。我们拓展 L 到状态集合并定义其反函数 L^- 。对于所有的 $S \in P(S)$ 和 $D \in P(D)$ 有: $L(S) = \{L(s) \mid s \in S\}$ 且 $L^-(D) = \{s \in S \mid L(s) \in D\}$ 。

给定模型 M , 对于任意一个从自然数到状态集合的函数 $\gamma: N \rightarrow S$ 满足: 对 $\forall i \geq 0, (\gamma(i), \gamma(i+1)) \in R$, 则称该函数为模型的一次运行。运行 γ 产生的迹, 记为 $L(\gamma)$, 是对 γ 上 L 的应用是可拓展的, 即, 对 $\forall i \geq 0, L(\gamma)(i) = L(\gamma(i))$ 。一个模型的迹语言, 记为 $\Omega(M)$, 是模型 M 的运行产生的所有迹的集合, 即, $\Omega(M) = \{L(\gamma) \mid \gamma \text{ 是模型的一次运行}\}$ 。

给定模型 M 和断言语言 L , 如果 $\Omega(M) \subseteq \Omega(L)$, 则称模型 M 满足断言语言 L , 记为 $M \models L$ 。给定并发规范 CS , 如果 M 满足 CS 中的所有的方程: 对 $L_i \in CS, M \models L_i$, 则 M 满足并发规范 CS , 记作 $M \models CS$ 。

模型检验最核心的思想就是对并发规范里的每

一个语言计算模型 M 上的模拟关系. 这里的模拟关系是并发规范内语言 $L_i (0 \leq i < n)$ 到状态集合 $T_i \in P(S)$ 的映射: 对于每一个状态 $s \in S, s \in T_i$, 如果有字和模型 M 上的运行满足 (1) $L(\gamma) = \text{ant}(w)$, 和 (2) $s = \gamma(|w|)$.

模拟关系的重要性体现在下面的定理.

定理 1 如果每一个前缀语言 $L_i = L_j \cdot \sigma_j$ 满足 $T_i \subseteq \text{cons}(\sigma_j)$, 则 $M \models \text{CS}$.

现在递归的显示如何计算基于这种结构的贵贵的模拟关系. 令 S_n 表示 n 个状态集合元组 $(S_0, S_1, \dots, S_{n-1})$. 定义偏序关系 $<: S_n < S_n$ 形成一个有限的 CS, 当且仅当 $\forall 0 \leq i < n, S_i \subseteq S_i'$. $(P(S) \times \dots \times P(S), <)$, 底元素为 $\varphi_n = (\varphi, \dots, \varphi)$.

算法 1: 并发 GSTE 验证算法

```

cGSTE(CS, M)
{
  //初始化模拟关系
   $T_0 = S$ ;
  for( $i = 1; i < n; i++$ )
   $T_i = \Phi$ ;
   $\text{active} = \{L_i \mid L_i = L_0 \cdot \sigma_j\}$ ;
  //模拟计算
  while( $\text{active} \neq \Phi$ )
  {
     $L_i = \text{pickOne}(\text{active})$ ;
    case  $L_i = L_0 \cdot \sigma_j; \text{new} = L^-(\text{ant}(\sigma_j))$ ;
       $L_i = L_j \cdot \sigma_j; \text{new} = \text{post}(T_j) \cap L^-(\text{ant}(\sigma_j))$ ;
       $L_i = L_{i_1} \cup \dots \cup L_{i_k}; \text{new} = \bigcup_{j=1}^k T_{i_j}$ ;
      else;  $\text{new} = \bigcap_{j=1}^k T_{i_j}$ ;
    endcase
    if( $\text{new} \neq T_i$ )
    add to  $\text{active}$  every  $L_k$  having  $L_i$  in its definition;
     $T_i = \text{new}$ ;
  } //endwhile
  //检验验证结果
  if( $T_i \not\subseteq \text{cons}(\sigma_j)$  for some  $L_i = L_j \cdot \sigma_j$ )
  return(false);
  return(true);
}

```

下面定义模型 M 的 n 个状态集合元组的更新函数: $Y(S_n) = (Y_0(S_n), Y_1(S_n), \dots, Y_{n-1}(S_n))$ 其中:

(1) 初始化 $L_0 = \varepsilon \cup L_0 \cdot \sigma_0; Y(S_n) = S$,

(2) 前缀 $L_i = L_j \cdot \sigma_j, 1 \leq i < h$

$Y_i(S_n) = \begin{cases} L^-(\text{ant}(\sigma_j)) & \text{如果 } L_j = L_0 \\ \text{post}(S_j) \cap L^-(\text{ant}(\sigma_j)) & \text{其他} \end{cases}$

(3) 求和 $L_i = L_{i_1} \cup \dots \cup L_{i_k}, h \leq i < l, Y_i(S_n) = \bigcup_{j=1}^k S_{i_j}$,

(4) $\text{meet} L_i = L_{i_1} \prod \dots \prod L_{i_k}, l \leq i < n, Y_i(S_n) = \prod_{j=1}^k S_{i_j}$.

更新函数 Y 是单调的, 序列 $(Y^0(\varphi_n), Y^1(\varphi_n), Y^2(\varphi_n), \dots)$ 是一个具有最小不动点的上升序列. 即, $\exists M \geq 0, \forall k \geq M, Y^k(\varphi_n) = Y^M(\varphi_n)$. $Y^M(\varphi_n)$ 是并发规范 CS 在模型 M 上的一个模拟关系.

5 验证实验与结果

图 2 所示为一个三输入端的电路. 该电路有三个输入端分别为 in1 、 in2 和 in3 , 一个输出端 out , 该电路的功能为当每个输入至少 1 次被输入为 1 时, 输出 out 输出为 1.

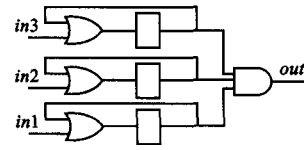


图 2 三个输入端的电路图

利用新的实现的验证算法来验证该电路的性质, 利用组合的方法可以先对每个缓存器单元进行独立的模拟. 图 3 所示为该三个输入端电路要验证的性质的并发断言图描述. 在图 3 中, 子断言图 A 和 B 和 C 分别有一条名为 Done 的边, 它们分别刻画了三个输入端 in1 、 in2 和 in3 至少输入了一次 1. 最后一个子断言图 Main 根据边上的组合同步条件 $(A \text{ ON Done}) \& (B \text{ ON Done}) \& (C \text{ ON Done})$ 将 A、B 和 C 子断言图的模拟结果组合来进行判断, 然后验证输出结果是否符合规范.

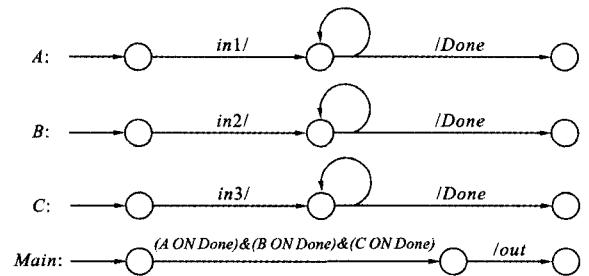


图 3 电路性质性质的并发断言图描述

对于该电路具有的此并发性质, 假如用 GSTE 的一个断言图来描述该电路的性质, 断言图如图 4 所示. 该图的上半部分的路径有 $3! = 6$ 条, 这 6 条表示 3 个输入端以任意的顺序输入各自的缓存器, 其中任意 2 个输入端都不会在同一时间输入. 最中间的一条路径指三个输入刚好同一时间输入各自的锁存器. 下半部分的路径表示 3 个输入端每次有 2 个输入端在同一时间输出, 共 6 条. 所以, 从初始顶点出发的分支路径有 13 条. 比图 3 多了 9 条路径.

从上面的例子可以看出,在描述具有并发性质的时候,如果用顺序断言图来描述,其复杂度和其输入或者模块的个数出现指数级的增长.而利用并发断言图,来描述验证并发性质则可以很好地解决这个问题.并发断言图的复杂度只会和输入或者模块的个数成线性增长.

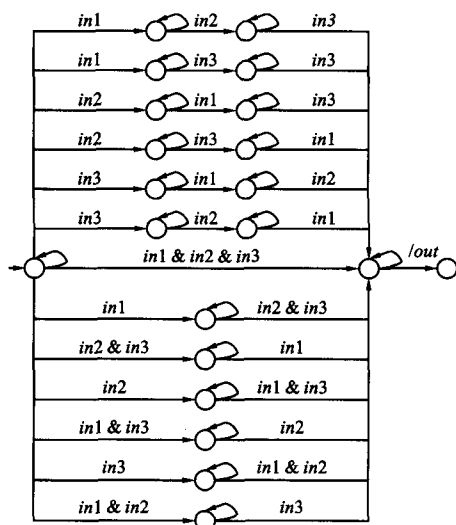


图4 三个输入端电路的断言图

综上所述,与传统的 GSTE 算法比较,新的并发 GSTE 的描述语言和验证算法能够更加简洁的描述和验证电路的并发性质,在验证并发性质时能够有效地减小断言图的复杂度,从而缩短其验证时间.

6 结束语

本文借鉴进程代数的表示方法提出了一个基于广义符号轨迹赋值的组合方法来克服先前断言图不能简洁描述电路并发性质的限制,提出了一个规范语言,它能用组合的方式简洁地描述系统的并发行为.这种组合是逻辑的,不依赖于对系统实现细节的深入理解.这个语言是对广义符号轨迹赋值规范语言的拓展,它引入了一个新的 meet 运算符,用类似于进程代数的方式表达.针对新的断言图规范,对经典的广义符号轨迹赋值的算法进行了修改,该算法能直接深入规范的语法结构并建立从规范的语言元素到电路状态集合的模拟关系.该算法能够高效而实用的直接验证并发断言图.

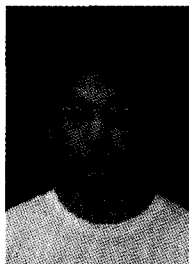
参考文献:

- [1] William K Lam. Hardware design verification: simulation and formal method-based approaches[M]. Boston: Prentice Hall PTR, 2005: 10-20.
- [2] Clarke E, Grumberg O, Jha S, et al. Counterexample

— guided abstraction refinement for symbolic model checking[J]. Journal of the ACM (JACM), 2003, 50(5): 752 - 794.

- [3] 郑德生, 杨国武, 雷航. 一种推广化符号轨迹赋值中伪报错的改进算法[J]. 微电子学与计算机, 2008, 25(5): 207-210.
- [4] Yang J, Seger X. Introduction to generalized symbolic trajectory evaluation, very large scale integration (VLSI) systems[J]. IEEE Transactions, 2003, 11(3): 345 - 353.
- [5] Yang J, Seger C. Generalized symbolic trajectory evaluation[C]// STE Symposium, A Satellite Workshop of CAV2000. USA: Chicago, 2000.
- [6] Schubert T. High-level formal verification of next generation microprocessors[C]// 40th ACM/IEEE Design Automation Conference. USA: California, 2003.
- [7] Hennessy M. Algebraic theory of processes[M]. Boston: MIT Press, 1988.
- [8] Henzinger T, Qadeer S, Rajamani K. You assume, we guarantee: methodology and case studies[J]. Computer Aided Verification, 1998 (LNCS 1427): 440 - 451.
- [9] Henzinger T, Qadeer S, Rajamani K, et al. An assume-guarantee rule for checking simulation[J]. ACM Trans. on Programming Languages and Systems, 2002(24): 51 - 64.
- [10] Hoare C. Communicating sequential processes[M]. Boston: Prentice Hall, 1985.

作者简介:



郑德生 (1983-), 博士研究生. 研究方向为形式化验证、模型检测.



杨国武 教授, 博士生导师. 研究方向为形式化验证、量子计算.



徐忠林 硕士研究生. 研究方向为形式化验证及其在硬件电路的应用.