

模型检测中状态爆炸问题研究综述

侯刚 周宽久 勇嘉伟 任龙涛 王小龙
(大连理工大学软件学院 大连 116620)

摘要 模型检测已成为保证软件系统正确性和可靠性的重要手段,但随着软件功能日益强大,其规模和复杂度也越来越大,在模型检测过程中容易产生状态爆炸问题。如何解决模型检测中的状态爆炸,已成为工业界和理论界无法回避的重要课题。系统地综述模型检测领域解决状态爆炸问题的关键技术和主要方法,并提出该领域的最新研究进展与方向。

关键词 软件系统,模型检测,状态空间爆炸,形式化验证
中图分类号 TP301 **文献标识码** A

Survey of State Explosion Problem in Model Checking

HOU Gang ZHOU Kuan-jiu YONG Jia-wei REN Long-tao WANG Xiao-long
(School of Software, Dalian University of Technology, Dalian 116620, China)

Abstract Model checking has become an important approach to ensure the correctness and reliability software systems. However, with the increasingly powerful software functionality, system scale and complexity are also growing. Then it is easy to produce state explosion problem in model checking process. How to solve the state explosion in model checking has become an important issue that can not be avoided by the industry and theorists. In this paper, we overviewed the key technology and main methods of solving state explosion problem, and proposed the latest research progress and direction in this field of model checking.

Keywords Software systems, Model checking, State explosion, Formal verification

1 引言

模型检测(Model Checking)是一种应用于验证有限状态系统满足规范的形式化方法,主要针对具有逻辑性质的有限状态系统,目前已广泛应用于软件的可信性验证。该方法最早由 Edmund M. Clarke、Allen Emerson^[1]与 Joseph Sifakis^[2]等人分别独立提出。其思想是先建立待测系统的有限状态模型,然后用算法穷尽检测模型中的状态,判断其是否满足待测属性。若不满足,则根据反馈信息判断具体系统中是否确实存在违反此属性的执行路径(即反例路径)。模型检测工具在算法支持下可以自动执行,并能在系统不满足性质时提供反例路径,因此备受工业界关注。

模型检测是基于对状态空间的穷举搜索,对于并发系统,其状态的数目往往随并发分量的增加而呈指数增长。因此,当系统的并发分量较多时,直接对其状态空间进行搜索实际上是不可能的,这就是所谓的状态爆炸问题。由于软件涉及无穷数据域上的运算,因此状态爆炸问题十分突出,已成为模型检测方法应用于软件可信评价及验证的一个具有挑战性同时又无法回避的难题。

对于这一难题,学者首先想到通过减少和压缩状态空间的方法来减少计算量,如符号模型检测、对称模型检测、程序

切片、偏序规约、抽象等。本文将按照模型检测技术的发展历程综述上述技术和方法(如图1所示),并对该领域的最新研究热点与现状进行分析与讨论。

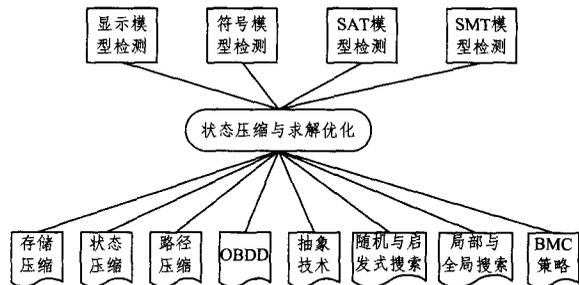


图1 主要的状态压缩技术与求解策略

2 显示模型检测

在模型检测领域早期采用显示状态表示(Explicit State Expression)的方法,即通过列表或者表格等方式显示存储所有的状态空间,这些状态表的大小与系统模型的状态数成正比。而模型的状态数与并发系统的大小成指数关系。因此,随着所要检测的系统规模增大,所要搜索的状态空间呈指数增长,算法验证所需要的时间/空间复杂度将超过实际硬件所能承受的程度。因此,状态空间爆炸极大地限制了显示模型

本文受国家自然科学基金项目(91018003,61272174)资助。

侯刚(1982-),男,博士生,讲师,主要研究方向为可信软件、形式化方法, E-mail: hg_dut@163.com;周宽久(1966-),男,博士,教授,主要研究方向为可信软件、形式化方法、系统工程。

检测应用。为了解决这一问题,过去的 20 年内,在显示模型检测方面有超过 100 篇以上的文章来讨论如何解决状态爆炸问题。总起来看,大体上可以分为 4 种基本思想:状态空间压缩、存储空间压缩、并行和分布式计算、随机化和启发式搜索算法。

2.1 状态空间削减

当我们观察一些简单模型及其状态空间时,会发现状态中存在一定的冗余,冗余的存在说明部分状态空间是等价的。因此,应该考虑删减冗余状态,减小搜索空间。在这一过程中,应该指明哪些状态需要删减,并且证明删减后的状态空间等价于删减前完整的状态空间。总起来说,主要有 3 种方法体现了这种思想。

2.1.1 基于状态的削减

在这方面比较有代表性的研究是对称性削减(Symmetry Reduction),该方法由 Clarke^[3]和 Emerson^[4]等人提出,此后也有大量研究文章发表^[5-11],它是一种通过避免对系统中对称状态重复搜索而有效减少验证规模的方法。对称性经常表现在包含多个相同进程的并发系统中,例如互斥协议(MUTEX)。由于交换相同进程不会影响系统的整体行为,因此可以采用自同构置换群刻画进程之间的对称性,并基于此,将系统的状态空间划分为若干个等价类。进一步,在所有的等价类上定义迁移关系得到商结构(Quotient Structure)。根据对称性,商结构与原始系统之间呈互模拟关系,如图 2 所示。在商结构上对时序逻辑所表达的系统属性进行验证时,可以得到与原系统上相同的结果。由于根据置换群所划分的等价类数目远小于系统原有状态,因此采用对称化简可以有效地减少系统状态空间,提高模型检测效率。

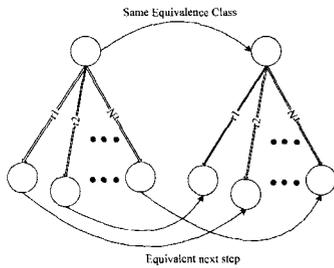


图 2 等价类上的互模拟关系

此外,针对验证系统的状态削减,有效方法还包括活变量削减(Live Variable Reduction)^[12,13]、影响锥削减及切片(Cone of Influence Reductions, and Slicing)^[14,15]等方法。活变量削减^[12]是指采用有效的活性变量信息构成等值,可以挖掘变量或者队列上未使用的死数据,使用这种削减方法最大好处是可以在不生成初始值的情况下生成指数模型。切片方法则是通过切片技术将与验证性质无关的代码去除,减小相关验证模型的大小。针对切片技术,J. Hatcliff^[14]提出了关于模型验证中有限自动机转换的切片方法,并通过实验证明了其切片技术的可行性;B. Dwyer^[15]提出了一种适用于 Java 程序检测的切片工具,实验证明该工具能够有效削减被测程序的状态空间。

2.1.2 基于路径的削减

在这方面比较有代表性的研究是偏序规约(Partial Order Reduction),该方法用以解决并发异步模型交替执行导致的状态空间爆炸,由 Overman 在 1981 年首先提出^[16],此后,也

涌现出大量研究成果^[17-23]。一个系统可以由多个进程组成,并发执行使得不同进程动作可以有許多不同次序,这使得验证成本大幅提高。并发系统的一次执行本质上可看作各事件不断插入的线性序列。这样,对 n 个事件可产生 $n!$ 种线性序列。通过对各事件分析可发现,有些事件可能与执行次序无关,将这些事件预先插入到序列中的固定位置,则可避免重复验证一批本质上相同的路径。偏序规约在缓解状态空间爆炸问题上,侧重于减少验证路径来节省内存空间。但其需要通过判断来分析哪些事件可预分配固定位置,这使时间成本增加。在衡量时间和空间成本中,偏序规约更倾向于以最小空间成本缓解状态爆炸问题。

在基于路径的状态压缩方面,有效方法还包括转换融合(Transition Merging)^[24,25]以及同步可达性分析(Simultaneous Reachability Analysis)^[26]等手段。转换融合是将一系列系统转换压缩为一个跳变(Jump Transition),从而形成一个简化版本的待测模型。R. Kuishan^[10]在其研究中给出了相应的压缩算法,并通过实验证明其压缩算法保留了 LTL 每一步随机的性质。同步可达性分析建立在对多个指定协议的验证中采用多信道同时处理多次可执行的转换,通过每个协议提供的全局状态来执行变换。

2.1.3 组合方法

组合方法是一种通过检测局部状态空间来验证整体状态空间的方法^[27-30]。对于大型系统验证,组合方法利用“分而治之”策略,根据系统模块结构,先分别验证系统各个局部模块性质,再由局部模块性质推断整个系统性质。如果系统满足每一局部性质,并且局部性质的合取蕴涵了整个规范,那么完整的系统也必定满足这个规范。总而言之,组合推理是借助分治策略,从本地属性推导出全局属性,从而减小状态空间。

组合验证主要有两种策略^[31-33]:简单组合策略和假设/保证(Assume/Guarantee)组合策略(A/G 组合策略)。其基本思想的形式化表述如下:定义“||”操作表示系统组合,假设系统由两个子系统 P 和 Q 组成,则完整系统可用 $P || Q$ 来表示。再假设子系统 P 的子规约(性质)为 α ,子系统 Q 的子规约为 β (其中 α, β 为 LTL、 \forall CTL 或者 \forall CTL* 公式),则简单组合验证策略的形式如式(1)所示。

$$\frac{P | \alpha, Q | \beta, \alpha, \beta = \varphi}{P || Q = \varphi} \quad (1)$$

简单组合策略要求模块在任意环境下都满足该局部性质,而没有考虑模块之间的相互影响,因此,在验证时较为简单,但是许多实际问题无法通过这种策略获得解决。

在实际使用组合验证时,往往要考虑到环境因素,因此,提出了一种考虑环境因素的(A/G)组合验证策略。该策略将要验证的子系统期望环境(所满足的性质)表达为时序逻辑公式,如设 P 的环境满足时序逻辑公式 α ,而且在环境满足公式 α 的条件下,可以验证 P 满足性质 β ,如果可以另外验证 Q 满足公式 α ,则可以得出结论: $P || Q = \beta$ 。该推理过程如式(2)所示。

$$\frac{\langle \text{true} \rangle Q(\alpha), \langle \alpha \rangle P(\beta)}{\langle \text{true} \rangle P || Q(\beta)} \quad (2)$$

组合方法的难点在于如何正确分解系统和规范,同时不破坏原系统的整体性和正确性。该方法主要缺点在于设计者需要手工定义子系统划分和属性划分,这降低了模型检验的自动化优势。为解决这一问题,JM. Cobleigh 于 2003 年提出

自动化的组合验证方法^[34],该方法能够自动分析出失效的环境假设,并加以改进,重新进行组合验证。目前,组合验证已成功应用于大型系统的验证中。KenMcMillan 成功将组合验证方法应用于处理机的 Cache 一致性协议^[35]以及微处理器乱序执行单元^[36];2008 年,Gurov 利用组合验证方法对基于控制流的顺序程序安全属性进行验证^[37],并取得了很好的效果;Ahrendt 在最近发表的文章中利用组合验证方法建立了面向对象建模语言 Creol 的验证系统^[38]。

2.2 存储空间压缩

对存储空间的需求是模型检测主要瓶颈,因此,可以采用以时间换空间的方法节约存储空间。存储空间消耗主要体现在搜索算法需要存储已经访问过的状态空间,因此,对存储空间的压缩主要针对这一问题。现有的研究中,主流方法包括字节向量压缩(Byte Vector Compression)^[39-44]、缓存和选择性存储(Caching and Selective Storing)^[45-47]以及扫线法(Sweep Line Method)^[48-50]等。在这方面 G. J. Holzmann 等人做了大量研究工作,提出了包括实现无损压缩的 Collapse Compression 算法^[51,52]、Minimized Automaton Representation 算法^[53]以及实现有损压缩的 Bitstate Hashing 技术^[54,55]、Hash Compact 方法^[56,57]等众多压缩算法与技术。例如 Minimized Automaton Representation 算法不是将系统可达状态存储在传统的哈希查找表中,而是将它们存储在一个最小化的确定性有限状态自动机中,并通过建立和维护这个最小自动机来匹配状态。这个自动机可以表示成一个有限图,在搜索过程中每遇到一个状态时都要访问该图,如果发现新状态就立即更新该图。使用该压缩方法可以节省大量的内存,相对于标准的搜索方法,它有时能使验证器所需的内存空间以指数级的方式减少,但同时时间的开销将明显增大。

存储空间压缩方法适合于对计算存储空间要求苛刻的环境,通过不断压缩计算空间,以时间上的累积来换取空间利用率的提高。当然使用者也需要考虑时间复杂度,要保证所花费的验证时间在可接受的限度内。

2.3 并行和分布式计算

此类方法主要利用网络工作站进行分布式计算或者通过多核处理器进行并行计算来提高状态空间的搜索效率。在分布式环境下,状态空间被分割成组并存储在不同的服务器上,服务器之间通过交换消息传递将要访问的状态信息^[58-60],实现分布式搜索。但是,经典的搜索算法由于基于深度优先搜索(Depth-first Search),难以在分布式的环境下使用,也有大量文献研究适用于分布式环境下的搜索算法^[61-66]。例如在文献[61]中,J. Barnat 提出了在 SPIN 中的分布式 LTL 模型检测算法,分析了在分布式 SPIN 中嵌套深度优先搜索的性能。

此外,随着多核处理器的广泛应用,也有相关学者研究在多核环境下,通过并行化的搜索策略,提高模型验证的效率^[67,68]。2007 年,J. Barnat 在其研究^[67]中提出了采用多核技术进行 LTL 模型验证的思想,通过设计共享内存的算法,来构建一种基于共享内存的并行 LTL 模型检测器。

借助日益发展的多核及分布式技术来加速模型验证,在时间上极大地提高了验证效率,是一种缓解空间状态爆炸的有效途径。可以预见的是,结合分布式计算及多核并行技术的模型验证将是未来发展的重要趋势。

2.4 随机化技术和启发式算法

当实际验证问题规模过大时,即使采用多种策略也难以解决状态空间爆炸,此时,可以考虑使用随机化技术或者启发式算法来验证部分状态空间。在这样的求解模式下,模型正确性验证无法保证,但是可以有效发现模型错误。

2.4.1 启发式搜索

状态空间爆炸问题已被证明为 NP 完全问题。求解 NP 完全问题时,启发式搜索算法利用问题自身给出的特性信息,将搜索定位在最关注的子问题方向上。在可接受的计算时间、占用空间等开销的前提下,给出待解决问题的一个可行解。这组可行解并不能保证最优性,但却对整个问题的深入解决有着深刻的意义。在启发式搜索过程中,关键步骤是如何确定下一个考察结点,不同的确定方法就形成了不同的搜索策略。首先考虑选择重要性高的结点时,可用于指导搜索过程并与具体问题求解有关的控制性信息称为启发性信息。充分利用与问题求解有关的启发性信息,估计出节点的重要性,以利于得到最优解。

针对模型检测中的启发式搜索,A. Groce 提出了用启发式搜索算法检测 Java 程序的思想^[69],针对不同种类的特定错误,实现了在程序上进行结构式启发搜索。实验证明这种结构式搜索效果比传统启发式搜索更有效,同时在启发搜索算法中引入概率因素,使问题考虑得更加全面。A. Kuehlmann 等人研究了随机行走模型,依据各自概率赋予每个状态不同优先级的启发式搜索算法^[70]。Zhou 研究了基于马尔科夫使用模型(Markov Usage Model)的启发式软件验证方法,通过计算函数状态间的迁移概率,指导对状态空间的搜索过程^[71]。启发式搜索算法为解决状态空间爆炸问题提供了新的思路,因此,加强启发式算法与其他算法思想结合将提升求解这一问题的可能性。

2.4.2 随机搜索与局部搜索

随机搜索技术在理想情况下,认为目标位置基本服从均匀分布,搜索轨迹将随机且均匀分布于目标区域。在随机采样过程中比较随机点函数值大小,然后逐步缩小搜索区域,收敛速度因为随机搜索轨迹的随意性和搜索力散布的均匀性而比较低,但随机搜索算法对于解决一类不带特殊性质的函数有着天然的优势。针对随机化搜索算法在状态爆炸问题中的应用,H. Sivaraj 提出了在分布式存储模型检测中使用基于启发式算法的随机搜索技术^[72],该技术首先显式枚举状态,再通过随机化处理简化搜索过程。此外,P. Haslum、RadekPelánek 等人也给出了针对模型检测的随机搜索算法^[73,74]。

局部搜索算法将搜索限制在局部范围,在每次迭代中,算法依据邻域函数在当前解的某个邻域内寻找更优解来改善已找到的最优解的质量。若在迭代中,当前解的邻域内没有更好的解,则此解为局部最优解。在解决状态空间爆炸问题上,M. D. Jones 实现了在 LTL 性质验证中引入并行局部搜索算法^[75];G. J. Holzmann 等人也在协议验证中使用了相应的局部搜索策略^[76-78]。

2.4.3 位状态哈希

与传统方法相比,位状态哈希技术通过可达性分析来增强验证质量,并能够兼容其它验证方法,因此,在模型检测领域得到了一定程度的应用^[79]。检测工具 SPIN 就采用了其相

似的思想, Peter. C. Dillinger 提出 SPIN 快速准确位状态验证方法^[80], 即通过对 SPIN 中哈希计算提供多个有效哈希值来加快遍历状态空间的速度。其后, 又在概率验证中引入了 Bloom 过滤器来加快验证速度, 并引入 3SPIN 的概念^[81]。2005 年, Peter. C. Dillinger 又将动态规划与 3SPIN 结合^[82], 通过位哈希技术以最小时间发现错误。

3 符号模型检测

符号模型检测的思想最早由 McMillan 提出^[83], 区别于显示模型检测用表的形式显示表示状态空间, 它使用布尔函数来表示模型状态集合和状态转换关系, 并使用二叉决策图在计算机中存储布尔函数, 从而在很大程度上缓解了状态空间爆炸问题, 增大了模型检测可以处理问题的规模。

二叉决策图(Binary Decision Diagrams, BDD)是 Lee 提出^[84]的一种表示和操作布尔函数的数据结构, 是一种有向无环图(DAG)。它首先以二值判定树(Binary Decision Tree, BDT)的形式被提出来的, 二叉决策图可以通过对二值判定树进行优化(如对一些计算进行短路、规约)得到。二叉决策图中每一个非终端结点用布尔变量 x, y, z, \dots 表示, 而终端结点用 1 或 0 标记。每一个非终端结点 v 由一个变量 $\text{var}(v)$ 标识, 它有两个后继结点: $\text{low}(v)$ 对应变量被赋值为 0 的情况, $\text{high}(v)$ 对应变量 v 被赋值为 1 的情况。每一个终端结点 v 被表示为 $\text{value}(v)$, $\text{value}(v)$ 取值为 0 或 1。从初始结点出发到终端结点, 我们可以确定该路径上对变量的赋值是否使该二值判定树表示的布尔公式为真。因此, 二叉决策图既能较为紧凑地表示布尔函数, 也可以方便地表达布尔运算。

为进一步压缩二叉决策图 Bryant 等提出了自底向上的压缩算法^[85], 该方法通过冗余删除、合并同构树等方法将二叉决策图简化为有序的二叉决策图(Ordered Binary Decision Tree, OBDD), 它是布尔函数的一种规范表示形式(如图 3 所示)。与真值表、DNF、CNF 相比, 通常情况下, OBDD 所用的存储空间小, 布尔运算效率高, 能够减少随着问题规模增长而导致的布尔公式指数级增长, 较好地解决了状态空间爆炸问题。

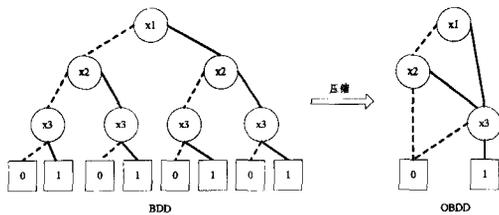


图 3 BDD 压缩为 OBDD

目前, 基于 OBDD 及其扩展形式的符号模型检验已成为系统验证的流行技术。Strehl 提出了适合于进程网络模型的符号验证 IDD 技术和算法^[86]。Moller 给出了赋时自动机和赋时 Petri 网等实时系统模型的符号 DDD 分析和验证方法^[87]。Campos 研究了基于符号模型检验的实时系统时间性能分析, 并给出了其在医疗监护系统、飞机控制器中的应用简例^[88]。Yoneda 实现了时间 Petri 网的符号模型检验方法^[89]。符号模型检验也在通讯协议、铁路互锁系统、大型软件系统规格、嵌入式系统设计、FPGA 设计等领域问题的验证中得到了成功应用^[90-92]。此外, 也有学者针对 OBDD 进行了相关的优

化研究^[93,94]。

4 可满足性理论

命题逻辑公式的 CNF 范式可满足性问题 SAT 是计算机科学重要的核心问题, 模型检测的许多问题都可转化为 CNF 范式的可满足性问题来求解, 例如 VLSI 设计验证等。SAT 问题被证明是 NP 完全问题, 同样存在状态空间爆炸。目前解决该问题的方法主要包括搜索算法优化、有界模型检测和相关硬件加速方法。

4.1 完备搜索与不完备搜索

SAT 搜索算法主要有完备算法^[95-98]和不完备算法^[99-105]两大类。完备算法基于穷举思想和回溯搜索机制, 系统地搜索 SAT 问题的解空间, 如果问题是可满足的, 则给出命题逻辑公式的所有解; 如果公式是不可满足的, 则给出完备性证明。完备性算法的优点是能够保证找到对应 SAT 问题的解或者证明公式不可满足, 但是计算效率低, 平均时间复杂度虽然是多项式级的, 但是最坏情况下时间复杂度却是指数级的, 不适用于求解大规模的 SAT 问题。不完备算法也称局部搜索算法, 其基本思想是给定一个全局赋值向量, 以此来减少不可满足子句的数目, 其优点是求解速度快, 可以用于验证不满足性, 但是不能保证验证 SAT 问题的可满足性。

在 SAT 模型验证领域, 针对状态空间爆炸问题, 算法方面的研究主要集中在对完备算法和不完备算法的改进, 以提高 SAT 的求解效率。在完备算法研究方面, 最具代表性的算法是 Davis 和 Putnam 等人提出的 DPLL^[95]算法, 以及在此基础上引入启发式搜索策略的二代 DPLL 算法: SATO^[96]、POSIT^[97]、GRASP^[98]等; 在此之后的改进又引入了冲突子句学习技术(冲突子句分析和学习、回跳、监视字等)来形成更高效的 SAT 算法, 如 MINISAT^[98]、CHAFF^[100]、BERKMIN^[101]。在不完备算法研究方面, 比较经典的局部搜索算法有 GSAT^[102]、WSAT^[103]、NSAT^[104]、TSAT^[105]、SDF^[106]、RANGER^[107]和 GUNSAT^[108]等, 其中常用的局部搜索算法是基于贪婪搜索的 GSAT 和基于噪声模型分析的 WSAT。在 GSAT 算法中, 变量的翻转很可能会陷入一个局部圈子, 无法跳出, 从而影响求解; 而 WSAT 算法是在 GSAT 算法基础上引入随机噪声参数, 改进的主要目的就是减少其陷入局部圈子的概率。这些算法都具有较高的求解效率, 在一定程度上缓解了状态空间爆炸问题。

4.2 有界模型检测

有界模型检测^[109,110]是针对基于 OBDD 技术模型检测的不足, 如状态空间爆炸、需要人工设定逻辑公式中的变量顺序才能有效压缩存储空间、检测变量少等问题而产生的一种新的模型检测技术。其优势是把 BMC 问题编码成 SAT 实例, 充分利用 SAT 工具进行求解, 使验证变量数提升一个数量级以上; 此外由于采用宽度优先搜索(Breadth-first Search), 所获得的反例是长度最短、最简明的反例, 有利于设计者理解问题, 找出原因。已有研究证明当验证边界上界 k 小于 60 时, BMC 要优于传统的模型检测^[111]。

有界模型检测的主要过程是: 先把要验证的系统或模型构造为有限状态自动机(Finite State Machine, FSM), 通过 FSM 状态间的转移来模拟系统或模型运行; 要验证的规范说明用线性时序逻辑 LTL(Linear-time Temporal Logic)进行说

明;设定验证边界上界 K ;FSM 状态间转移关系和 LTL 逻辑规范的否定形式 NNF(Negation Normal Form)通过“逻辑与”构成 BMC 转换公式;把 BMC 转换公式编码成 SAT 实例,通过 SAT 工具求解。若有解,则找到反例;反之,若不可满足,则表明要验证的系统或模型运行到 K 阶段时,是安全的、没有错误的。

近年来,有界模型检测已成为模型验证领域的研究热点。目前,主要在 3 个方面提高 BMC 性能:其一在于对 BMC 转换公式的优化^[112-115];其二是编码成 SAT 实例时,对其变量和子句的优化^[116,117];其三将 BMC 问题映射成 SAT 实例后,针对 SAT 子句特点,优化 SAT 工具,提高 SAT 求解效率^[118]。上述第 1 种方法是采用把 BMC 转换公式转换为逻辑等价且结构简单、易于实现的 BMC 公式,这样的公式在以后进行编码时,可直接产生变量和子句都较少且易解的 SAT 实例,从而提高 BMC 的效率。第 2 种方法也是尽可能生成少的变量和子句,但在优化时有可能破坏 BMC 问题特征,若再用第 3 种方法中针对 BMC 优化的 SAT 工具,可能无法取得较好的求解效果^[115]。

4.3 硬件加速方法

随着超大规模集成电路技术的发展,将可编程逻辑阵列(Field-Programmable Gate Array, FPGA)、图形处理器(Graphic Process Unit, GPU)等作为硬件加速器来求解 NP 完全问题已取得了较好的效果,其中包括基于硬件加速器的 SAT 问题求解。区别于并行计算解决状态空间爆炸问题,硬件加速方法在挖掘求解过程并行性的同时,更为重要的是通过底层硬件逻辑门电路直接计算布尔代数,这将极大提高 SAT 问题的求解效率。

在通过 FPGA 进行 SAT 加速求解方面,主要分为实例型(Instance Specify)算法和应用(Application Specify)型算法。在实例型算法中,硬件结构针对每一个实例进行编译配置然后计算,每一个不同实例对应不同硬件结构。1996 年, Suyama 等人^[119]首次提出了实例型硬件直接逻辑算法,该算法首先将 SAT 实例的 CNF 公式转换为 3-SAT,然后由 SFL Generator 转换为 HDL 语言,将其编译下载后在 FPGA 固化为硬件逻辑,形成 CNF 的 Clause 逻辑计算模块,通过在 $2N-1$ 的空间用穷举法或类似 DP 回溯算法搜索取值,对实例的可满足性进行求解;1999 年, P. Zhong 等人提出硬件状态机 DP 算法,该算法将 SAT Clause 编译为 FPGA 硬件 Deduction 模块和 Implication 模块,并采用流水线结构进行计算^[120];2000 年, Abromavici 等人提出硬件并行 PODEM 算法,该算法将 CNF 编译成为对应的 PODEM 电路形式,通过向前推导输出模型(Forward Model)和向后反推输入模型(Backward Model),进行多路并行的推导,得到 SAT 问题的解^[121];2002 年, Dandalis 等人提出了硬件混合算法,将算法在软件和硬件中进行功能划分,从而将算法中耗时最多的 Deduction 模块和 Implication 模块由 FPGA 硬件来实现^[122]。

在应用型算法中,硬件结构针对应用进行一次编译和配置,然后在同一台硬件机构上对应用中不同的实例进行计算。2002 年, Sousa 等人提出 FPGA 硬件应用型算法,该算法以 3SAT 为对象,采用寄存器阵列存储 CNF 实例将算法在软件和硬件中分工^[123,124];2004 年, Skliarova 等人提出了 FPGA-DP 混合算法,该算法以三态存储阵列作为联系变量和 Clause

阵列的矩阵模块,从而实现了将 SAT 实例存储到矩阵模块中的转换^[125];2008 年, J. D. Williams 等人也提出了自己的 FPGA-DP 混合算法,该算法将软硬件模块功能进行合理划分,并实现了 BCP 在硬件 FPGA 中的运算和相关接口模型^[126]。

FPGA 实例型算法编译配置过程本身就是一个 SAT 问题的求解过程,计算时间较长,而应用型硬件算法相对来说更具实用价值。此外,与不断增长的 SAT 应用需求相比, FPGA 硬件平台资源总显得不足,很多 SAT 软件算法中的先进技术硬件暂时还很难实现,因此,将 SAT 求解过程在软件和硬件上合理划分的混合型算法更具合理性。划分分为两种方式:按照计算复杂性划分,将计算量大的、可以并行计算的部分划分给硬件系统;按照存储量或者逻辑容量来划分,将 FPGA 不能容纳的部分由软件来管理和控制^[127]。

此外,在通过 GPU 进行 SAT 加速求解方面,罗忠文等人^[128,129]也提出了相应的求解方案,验证了 GPU 对小规模 3-SAT 问题求解的适用性。此类研究结论大多关注自身方法与软件方法之间的加速比,因此,无法有效评价其在验证规模上的效果。

5 可满足性模理论

SAT 的求解对象是命题逻辑公式,而命题逻辑表达能力相对较弱,无法直接表达许多特定应用领域问题,比如在 RTL 电路验证中,由于 SAT 抽象层次较低,采用位级信息描述问题将丢失大量逻辑信息,从而导致验证失败。为解决这一问题,引入了一阶逻辑,与之对应的可满足性判断问题称为 SMT 问题。一阶逻辑由于是在命题逻辑基础上补充量词和项,因此抽象层次更高,表达能力更强。但和 SAT 相似, SMT 同样存在状态空间爆炸问题,目前解决该问题的主要方法有抽象技术、持续验证技术以及有界模型检测(BMC)等。由于 SMT 中的有界模型检测^[130-133]思想与 SAT 中类似,因此,在此重点综述另外两项技术。

5.1 抽象技术

抽象技术^[134-137]是解决 SMT 状态空间爆炸问题的一个有效手段(不限于解决 SMT 状态空间爆炸,也可用于其它类型模型检测)。在验证系统性质时,不是直接对实际系统进行分析,而是剔除系统无关信息,仅保留有用信息,从而构建抽象系统,再对该抽象系统进行分析验证就会变得容易,从而在一定程度上缓解了状态空间爆炸。目前,常用的抽象方法包括数据抽象(Data Abstraction)、谓词抽象(Predicate Abstraction),以及基于反例的抽象提纯(Counter-Example Guided Abstraction Refinement, CEGAR)等。在使用时,这些方法并不是完全相互独立,在实际的验证系统中往往需要集成多种抽象方法,以取得较好的效果。

数据抽象^[134]是一种通用性很强的抽象技术,其基本思想是通过去掉系统的部分数据信息来构建状态数较小的系统模型。当系统变量的定义域为无穷域时,将导致系统拥有无穷状态,而通常所需验证的性质与系统变量的具体取值无关,因此可以为每一个变量定义一个有穷的抽象数据类型,来替代原数据类型,实现对状态空间的压缩。数据抽象可以直接从原始系统中构造抽象模型,其构造过程一般需要手工完成。用数据抽象构造的系统模型状态空间会得到压缩,但系统行为却比原系统多,因此这种抽象是过近似的。

谓词抽象是应用范围最广的软件系统抽象技术之一,由 Graf 等人于 1997 年提出^[138],此后,相关学者也做了大量研究^[139-145]。该技术实现了自动将无穷状态系统映射到有穷状态系统的方法,而且基于谓词抽象的软件模型检测有效地结合了定理证明和模型检测技术。在谓词抽象中,抽象状态对应在该状态上成立的谓词公式的集合,其构造过程是从抽象初始状态出发,通过定理证明器来判定后续状态中每一个谓词公式是否成立。在谓词抽象中,抽象迁移关系计算最为耗时。最坏情况下,构造抽象模型的时空开销随谓词数目呈指数增长。因此,使用尽可能少的谓词构造抽象模型,是设计高效模型验证算法的关键。

基于反例的抽象提纯是一种实用的抽象策略^[136,146,147],在实际构造抽象系统过程中,总是首先构造一个相对粗糙的初始抽象模型,然后采用逐步求精方法来产生一个适用的抽象模型。具体流程如图 4 所示,首先在抽象模型中进行验证,如果没有发现错误,则实际系统中也没有错误;如果发现反例,则需要在实际系统中测试该反例,若在实际系统中该反例并未引起错误,说明上述反例为伪反例,分析出现伪反例的原因,以避免该伪反例的信息加入到抽象模型中(对模型纯化操作),继续进行验证。采用这种方法既可以有效降低模型抽象难度,又可以得到适当精度。

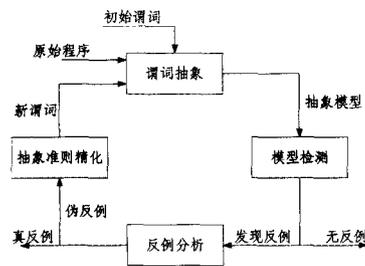


图 4 CEGAR 框架

5.2 持续性验证

近年来,尽管 SMT 求解器验证能力不断增强,但是针对大型软件验证问题仍显得力不从心。针对这一问题, Lucas Cordeiro 等人^[148]提出了持续性验证概念,目的在于通过软件配置管理系统(Software Configuration Management, SCM)获取信息,关注新的或者修改过的函数验证结果,从而快速检测系统设计错误。持续性验证首先调用 SCM 确定哪些功能或者方法被修改了,然后对这些被修改过的函数与修改之前的进行等价性对比,决定是否需要重新进行验证。因为证明两个函数版本等价性的时间消耗要比重新验证性质的消耗小,从而降低了系统验证时间。因此,持续性验证技术更适合大型软件状态空间的性质验证。

6 我们的工作与展望

作者所在课题组近年来开展了基于形式化方法的多核可信嵌入式软件开发与验证的相关研究。在软件模型方面,针对嵌入式软件采用状态机建模的特点,在传统 STM(State Transition Matrix, 状态迁移矩阵)建模方法的基础上^[149],研究并行 STM 建模方法,在保证前端建模易用性的同时,也保证模型具有严格的形式化语义。此外,针对并行 STM 模型验证问题,研究 SMT 与 BMC 策略相结合的验证方法,即将形式化的 STM 模型编码为一阶谓词逻辑公式,再通过 BMC

的求解策略逐渐展开状态空间,形成分步的、优化后的 BMC 转换公式,并通过 SMT 求解器进行逻辑验证。同时,在 BMC 转换公式中,引入 STM 自身的结构化信息和已经被验证的属性信息来压缩需要验证的状态空间。通过上述方法,有效抑制状态空间爆炸问题。

在程序验证方面,针对多核程序具有的原子性侵犯(Atomicity Violation)、数据竞争(Data Race)与死锁(Dead Lock)等问题,研究将程序控制流图(Control Flow Graph, CFG)与数据流图(Data Flow Graph, DFG)相结合的代码抽象方法,即通过自行开发的代码分析工具分析多核程序中的线程交织情况^[150-152],并构建程序控制流图和数据流图;然后使用形式化方法对多核程序进行形式化建模,形成多核源代码可达性树(Reachability Tree, RT),根据可达性树生成待验证 BMC 公式,最后通过 SMT 求解器进行可满足性验证。在解决状态爆炸问题时,通过寻找 CFG 中的最大强联通分量,有效地压缩程序中的循环路径,提高 SMT 求解器验证效率。

针对大规模的模型验证问题,课题组未来将研究基于 FPGA 的 SAT 加速求解技术,以通过 FPGA 并行计算能力提高模型验证速度,扩大验证问题规模。我们认为,大量的现实问题并不存在理论上的无穷状态空间,因此,状态爆炸问题可以通过硬件计算能力的提高,在可接受的时间要求内,得到有效的求解。通过 FPGA 实现 SAT 求解的具体过程如图 5 所示。首先,使用 SAT 转换算法将要求解的 CNF 公式转换成 3-SAT 公式;然后将 3-SAT 公式分割成小规模的 CNF 公式(称为 Bins),Bins 大小一定要能适应 FPGA 的计算结构,这种分割操作在 FPGA 之外的主机上实现;完成上述预处理后,将 CNF 实例(Bins 形式)通过 I/O 设备加载到 FPGA 板上的 DRAM 内存卡里;Leon 软件管理两方面调度:一是将实例从 DRAM 加载到 BRAM,二是将 Bins 从 BRAM 加载到 FPGA 构建的 SAT 求解器上;最后通过 SAT 求解器完成 CNF 公式并行计算,输出计算结果。

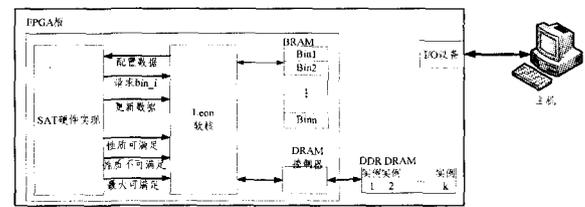


图 5 基于 FPGA 的 SAT 实现框架

结束语 近年来,软件系统可信性受到了越来越多的关注,模型检测技术已经得到了相当程度的重视。但是,随着软件系统规模的不断扩大和复杂性的不断提高,状态空间爆炸仍将是制约模型检测技术发展的主要瓶颈。本文总结了主流的状态空间爆炸解决方法,从实用性角度来看,并不存在单一的最优策略,而多种简单技术的融合往往会取得较好的效果。因此,研究工作不仅应关注新技术的开发,同时,也应注重结合自身模型的特点,研究现有技术的选择、整合以及高效调度。此外,针对大规模软件系统验证,我们认为比较有实用价值的研究方向包括:

(1)并行化算法及硬件加速:采用并行化技术求解,如基于 FPGA(或者 GPU)的模型验证是一个好的方向,尽管改变不了算法复杂性,但可以通过并行化算法及硬件加速,解决比以前更大规模的问题;

(2)采用层次验证与组合验证方法:人们在设计软件时往往采用模块化设计,将复杂问题分而治之,因此在软件验证时也应采用层次化验证与组合验证方法。

参 考 文 献

- [1] Clarke E, Emerson E. Design and synthesis of synchronization skeletons using branching time temporal logic [C] // Proceedings of Logic of Programs, 1981, 5000(131):52-71
- [2] Queille J, Sifakis J. Specification and verification of concurrent systems in CESAR[C] // Proceedings of the 5th Colloquium on International Symposium on Programming. LNCS 137, 1982: 337-351
- [3] Clarke E, Filkorn T, Jha S. Exploiting symmetry in temporal logic model checking [C] // Courcoubetis C, ed. Proceedings of the 5th Int'l Conf. on Computer-Aided Verification. LNCS 697, 1993:450-461
- [4] Emerson E, Sistla A. Symmetry and model checking [C] // Proceedings of the 5th Int'l Conf. on Computer-Aided Verification. LNCS 697, 1993:105-131
- [5] Norris I C, Dill D. Better verification through symmetry [J]. Formal Methods in System Design, 1996, 9(1/2):41-75
- [6] Sistla A, Godefroid P. Symmetry and reduced symmetry in model checking [C] // CAV. LNCS 2102, 2001:91-103
- [7] Iosif R. Symmetry reduction criteria for software model checking [C] // Proceedings of SPIN Workshop. LNCS 2318, 2002:22-41
- [8] Bosnacki D. A light-weight algorithm for model checking with symmetry reduction and weak fairness [C] // SPIN. LNCS 2648, 2003:89-103
- [9] Emerson E, Wahl T. Dynamic symmetry reduction [C] // Proceedings of Tools and Algorithms for the Construction and Analysis of Systems. LNCS 3440, 2005:382-396
- [10] Miller A, Donaldson A, Calder M. Symmetry in temporal logic model checking [J]. ACM Computing Surveys, 2006, 38(3):1-36
- [11] Wahl T. Adaptive symmetry reduction [C] // Proceedings of Computer Aided Verification (CAV'07). LNCS 4590, 2007:393-405
- [12] Fernandez J, Bozga M, Ghirvu L. State space reduction based on live variables analysis [J]. Journal of Science of Computer Programming (SCP), 2003, 47(2/3):203-220
- [13] Self J, Mercer E. On-the-fly dynamic dead variable analysis [C] // Proceedings of SPIN Workshop. LNCS 4595, 2007:113-130
- [14] Hatcliff J, Dwyer M, Zheng H. Slicing software for model construction [J]. Higher Order Symbol. Comput, 2000, 13(4):315-353
- [15] Dwyer M, Hatcliff J, Hoosier M, et al. Evaluating the effectiveness of slicing for model reduction of concurrent object oriented programs [C] // Proceedings of Tools and Algorithms for the Construction and Analysis of Systems. LNCS 3920, 2006:73-89
- [16] Overman W. Verification of Concurrent Systems Function and Timing [D]. UCLA, 1981
- [17] Peled D. Combining partial order reductions with on-the-fly model-checking [C] // Proceedings of Computer Aided Verification (CAV 1994). LNCS 818, 1994:377-390
- [18] Holzmann G, Peled D. An improvement in formal verification [C] // Proceedings of Formal Description Techniques VII. LNCS 234, 1994:197-211
- [19] Godefroid P. Partial-order methods for the verification of concurrent systems—an approach to the state-explosion problem [J]. LNCS, 1996, 1032:212-220
- [20] Penczek W, Sreter M, Gerth R, et al. Improving partial order reductions for universal branching time properties [J]. Fundamenta Informaticae, 2000, 43(1-4):245-267
- [21] Gueta G, Flanagan C, Yahav E, et al. Cartesian partial-order reduction [C] // Proceedings of SPIN Workshop. LNCS 4595, 2007:95-112
- [22] Gueta G, Flanagan C, Yahav E, et al. Cartesian Partial-Order Reduction [C] // Computer Science. LNCS 4595, 2007:112-115
- [23] Wang C, Yang Z, Kahlon V, et al. Peephole Partial Order Reduction [C] // Computer Science. 2008, 4963:382-396
- [24] Dong Y, Ramakrishnan C. An optimizing compiler for efficient model checking [C] // Proceedings of Formal Description Techniques for Distributed Systems and Communication Protocols and Protocol Specification. Testing and Verification, 1999:241-256
- [25] Kurshan R, Levin V, Yenigün H. Compressing transitions for model checking [C] // Proceedings of Computer Aided Verification (CAV 2002). LNCS 2404, 2002:569-581
- [26] Ozdemir K, Ural H. Protocol validation by simultaneous reachability analysis [J]. Computer Communications, 1997, 20:772-788
- [27] Pnueli. In Transition From Global to Modular Temporal Reasoning about Programs [M]. Logics and models of concurrent systems, 1985:123-144
- [28] Grumberg O, Long D. Model checking and modular verification [J]. ACM Transactions on Programming Languages and Systems, 1994, 16(3):843-871
- [29] Krimm J P, Mounier L. Compositional state space generation from Lotos programs [C] // Proceedings of Tools and Algorithms for the Construction and Analysis of Systems (TACAS 1997). LNCS 1217, 1997:239-258
- [30] 汤宪飞, 蒋昌俊, 丁志军, 等. 基于 Petri 网的语义 Web 服务自动组合方法 [J]. 软件学报, 2007, 12(1):2991-3000
- [31] 文艳军, 王戟, 齐治昌. 并发反应式系统的组合模型检验与组合精化检验 [J]. 软件学报, 2007, 18(6):1270-1281
- [32] Berezin S, Ser C, Clarke E M [C] // Compositional reasoning in model checking. LNCS 1536, 1998:81-102
- [33] Henzinger TA, Qadeer S, Rajamani S K, et al. An assume-guarantee rule for checking simulation [J]. ACM Trans. on Programming Languages and Systems, 2002, 24(1):51-64
- [34] Cobleigh J, Giannakopou D, Pasareanu C. Learning Assumptions for Compositional Verification [C] // Proceedings of 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS2003). LNCS, 2003:331-346
- [35] McMillan K. Parameterized Verification of the Flash Cache Coherence Protocol by Compositional Model Checking [C] // Correct Hardware Design and Verification Methods (ChARME 2001). Springer, 2001:2144
- [36] McMillan K. Verification of an Implementation of Tomasulo's Algorithm by Compositional Model Checking [C] // Proceedings of 10th Conference on Computer-aided Verification (CAV 98). LNCS 1427, 1998:100-121
- [37] Gurov D, Huisman M, Sprenger C. Compositional verification of

- sequential programs with procedures [J]. *Information and Computation*, 2008, 206: 840-868
- [38] Ahrendt W, Dylla M. A system for compositional verification of asynchronous objects [J]. *Science of Computer Programming*, 2012, 77(12): 1289-1309
- [39] Gregoire J. State space compression in spin with GETSs[C]// *Proceedings of Second SPIN Workshop*. Rutgers University, New Jersey, 1996
- [40] Visser W. Memory efficient state storage in SPIN [C]// *Proceedings of SPIN Workshop*. 1996: 21-35
- [41] Larsen K, Larsson F, Pettersson P, et al. Efficient verification of real-time systems: Compact data structure and state-space reduction [C]// *Proceedings of Real-Time Systems Symposium (RTSS'97)*. IEEE Computer Society Press, 1997: 14-24
- [42] Parreaux B. Difference compression in spin[C]// *Proceedings of Workshop on automata theoretic verification with the SPIN model checker (SPIN'98)*. IEEE Computer Society Press, 1998: 25-29
- [43] Geldenhuys J, e Villiers P. Runtime efficient state compaction in SPIN[C]// *Proceedings of SPIN Workshop*. LNCS 1680, 1999: 12-21
- [44] Geldenhuys J, Valmari A. A nearly memory-optimal data structure for sets and mappings[C]// *Proceedings of Model Checking Software (SPIN)*. LNCS 2648, 2003: 136-150
- [45] Godefroid P, Holzmann G, Pirotin D. State space caching revisited[C]// *Proceedings of Computer Aided Verification (CAV 1992)*. LNCS 663, 1992: 178-191
- [46] Geldenhuys J. *State caching reconsidered*[J]. *SPIN*, LNCS 2989, 2004: 23-38
- [47] Penna G, Intrigila B, Melatti I et al. Exploiting transition locality in automatic verification of finite state concurrent systems [J]. *Software Tools for Technology Transfer (STTT)*, 2004, 6(4): 320-341
- [48] Christensen S, Kristensen L, Mailund T. A sweep-line method for state space exploration [C]// *Proceedings of Tools and Algorithms for Construction and Analysis of Systems (TACAS 2001)*. LNCS 2031, 2001: 450-464
- [49] Mailund T, Westergaard W. Obtaining memory-efficient reachability graph representations using the sweep-line method [C]// *TACAS*. LNCS 2988, 2004: 177-191
- [50] Schmidt K. Automated generation of a progress measure for the sweep-line method [C]// *TACAS*. LNCS 2988, 2004: 192-204
- [51] Holzmann G, Godefroid P, Pirotin D. Coverage preserving reduction strategies for reachability analysis[C]// *Proceedings of Protocol Specification*. 1992
- [52] Holzmann G. State Compression in SPIN[C]// *Proceedings of the Third Spin Workshop*. 1997
- [53] Holzmann G, Puri A. A Minimized Automaton Representation of Reachable States[J]. *Software Tools for Technology Transfer*, 1999, 2(3): 270-278
- [54] Holzmann J. An Improved Reachability Analysis Technique [J]. *Software Practice and Experience*, 1998, 18(2): 137-161
- [55] Holzmann J. An Analysis of Bitstate Hashing [J]. *Formal methods in system design*, 1998, 13(3): 287-307
- [56] Wolper P, Leroy D. Reliable Hashing without Collision Detection[C]// *Proceedings of the 5th International Conference Aided Verification*. LNCS 697, 1993: 59-70
- [57] Stern U, Dill D. Improved Probabilistic Verification by Hash Compaction[C]// *Proceedings of IFIP WG 10. 5 Advanced Research Workshop on Correct Hardware Design and Verification Methods*. IFIP, LNCS 987, 1995: 206-224
- [58] Garavel H, Mateescu R, Smarandache I. Parallel state space construction for model-checking[C]// *Proceedings of SPIN Workshop*. LNCS 2057, 2001: 217-234
- [59] Lerda F, Sisto R. Distributed-memory model checking with SPIN[C]// *Proceedings of SPIN workshop*. LNCS 1680, 1999: 22-39
- [60] Lerda F, Visser W. Addressing dynamic issues of program model checking[C]// *Proceedings of SPIN Workshop*. LNCS 2057, 2001: 80-102
- [61] Barnat J, Brim L, Stern A J. Distributed LTL model-checking in SPIN[C]// *Proceedings of SPIN Workshop on Model Checking of Software*. LNCS 2057, 2001: 200-216
- [62] Barnat J, Brim L, Cern' a I. Property driven distribution of nested DFS [C]// *Proceedings of Workshop on Verification and Computational Logic*. DSSE Technical Report, 2002: 1-10
- [63] Barnat J, Brim L, Chaloupka J. Parallel breadth-first search LTL model checking[C]// *Proceedings of Automated Software Engineering (ASE'03)*. 2003: 106-115
- [64] Cern' a I, Pel' anek R. Distributed explicit fair cycle detection [C]// *Proceedings SPIN workshop*. LNCS 2648, 2003: 49-73
- [65] Barnat J, Brim L, Chaloupka J. From distributed memory cycle detection to parallel LTL model checking[J]. *ENTCS*, 2005, 133(1): 21-39
- [66] 孙伟, 马绍汉. 分布式博弈树搜索算法[J]. *计算机学报*, 1995, 18(1): 39-45
- [67] Barnat J, Brim L, Rockai P. Scalable multi-core ltl model-checking[C]// *Proceedings of SPIN Workshop*. LNCS 4595, 2007: 187-203
- [68] Holzmann G, Bosnacki D. The design of a multicore extension of the spin model checker[J]. *IEEE Transactions on Software Engineering*, 2007, 33(10): 659-674
- [69] Groce A, Visser W. Heuristics for model checking java programs [J]. *Software Tools for Technology Transfer (STTT)*, 2004, 6(4): 260-276
- [70] Kuehlmann A, McMillan K, Brayton R. Probabilistic state space search [C]// *Proceedings of Computer-Aided Design (CAD 1999)*. 1999: 574-579
- [71] Zhou Kuan-jiu, Wang Xiao-long, Hou Gang, et al. Software Reliability Test Based on Markov Usage Model[J]. *Journal of Software*, 2012, 7(9): 2061-2068
- [72] Sivaraj H, Gopalakrishnan G. Random Walk Based Heuristic Algorithms for Distributed Memory Model Checking[J]. *Electronic Notes in Theoretical Computer Science*, 2003, 89: 51-67
- [73] Haslum P. Model checking by random walk[C]// *Proceedings of ECSEL Workshop*. 1999
- [74] Pel' R anek, Han' zl T, Cern' a I, et al. Enhancing random walk state space exploration[C]// *Proceedings of Formal Methods for Industrial Critical Systems (FMICS'05)*. 2005: 98-105
- [75] Jones M, Sorber J. Parallel search for LTL violations [J]. *Software Tools for Technology Transfer (STTT)*, 2005, 7(1): 31-42
- [76] Holzmann G. Algorithms for automated protocol verification [J]. *AT&T Technical Journal*, 1990, 69(2): 32-44

- [77] Lin F, Chu P, Liu M. Protocol verification using reachability analysis: the state space explosion problem and relief strategies [J]. *Computer Communication Review*, 1987, 17(5): 126-134
- [78] Mihail M, Papadimitriou C. On the random walk method for protocol testing [C]//*Proceedings of Computer Aided Verification (CAV 1994)*. LNCS 818, 1994: 132-141
- [79] Holzmann G. An analysis of bitstate hashing [C]//*Proceedings of Protocol Specification*. 1995, 13: 301-314
- [80] Dillinger P, Manolios P. Fast and accurate bitstate verification for SPIN [C]//*Proceedings of SPIN workshop*. LNCS 2989, 2004: 57-75
- [81] Dillinger P, Manolios P. Bloom filters in probabilistic verification [C]//*Proceedings of Formal Methods in Computer-Aided Design (FMCAD)*. LNCS 3312, 2004: 367-381
- [82] Dillinger P, Manolios P. Enhanced Probabilistic Verification with 3Spin and 3Murphi [C]//*Model Checking Software Lecture Notes*. Computer Science 3639, 2005: 272-276
- [83] McMillan, Burch, Jerry R. Symbolic model checking for sequential circuit verification [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1994, 13(1): 401-424
- [84] Lee C Y. Representation of switching circuits by binary-decision programs [J]. *Bell Technical Journal*, 1959, 38(4): 985-999
- [85] Bryant R E. Graph-based algorithms for boolean function manipulation [J]. *IEEE Transactions on Computers*, 1986, 35(8): 677-691
- [86] Strehl K, Thiele L. Interval diagram for efficient symbolic verification of process networks [J]. *IEEE Trans on Computer Aided Design of Integrated Circuits and Systems*, 2000, 19(8): 939-956
- [87] Moller J, Lichtenberg J, Anderson H R, et al. Difference Decision Diagrams [C]//*Proc of Annular Conf of the European Association for Computer Science Logic*. 1999: 112-126
- [88] Campos S, Clarke E, Marrero W, et al. Timing analysis of industrial real-time systems [C]//*Proc of 1st workshop on industrial Strength Formal Specification Techniques*. Boca Raton, Florida, 1995: 97-106
- [89] Okawa Y, Yoneda T. Symbolic CTL model checking of time petri nets [J]. *Electronics and Communications in Japan*, 1997, 80(4): 11-20
- [90] Anderson R J, Beam P, Burns S, et al. Model checking large software specifications [C]//*Proceedings of the Fourth ACM SIGSOFT Symposium on the Foundations of Software Engineering* San Francisco CA, USA, 1996: 156-166
- [91] Corte L A, Eles P, Peng Z. Formal coverification of embedded systems using model checking [C]//*Proc of 26th EUROM ICRO Conference*. The Netherlanda, 2000: 106-113
- [92] Laij T, Pedram M, Vrudhulas B K. BDD based decomposition of logic functions with application to FPGA synthesis [C]//*Des Autom Conf*. 1993: 642-647
- [93] Sieling D. The Nonapproximability of OBDD minimization [J]. *Information and Computation*, 2002, 172(1): 103-138
- [94] 同华, 张伟, 赵海燕, 等. 基于二分决策图的特征模型验证方法 [J]. *软件学报*, 2010, 21(1): 84-97
- [95] Davis M, Logemann G, Loveland D. A Machine Program for Theorem Proving [J]. *Communications of the ACM*, 1962, 5(7): 394-397
- [96] Zhang Han-tao. SATO: an Efficient Propositional Prover [C]//*Proceedings of the 14th International Conference on Automated Deduction (CADE-97)*. London, UK, 1997: 272-275
- [97] Freeman J W. Improvements to Propositional Satisfiability Search Algorithms [D]. University of Pennsylvania. Philadelphia, PA, USA, 1995
- [98] Marques-Silva J P, Sakallah K A. GRASP: A new search algorithm for satisfiability [C]//*Proceedings of the ACM/IEEE International Conference on Computer-Aided Design*. Washington, DC, USA, 1996: 220-227
- [99] Een N, Sorensson N. An Extensible SAT-solver [C]//*Theory and Applications of Satisfiability Testing*. Santa Margherita Ligure, Italy, 2003: 502-518
- [100] Moskewicz M, Madigan C, Zhao, et al. Chaff: Engineering an Efficient SAT Solver [C]//*Proceedings of 38th Conference on Design Automation*. Las Vegas, NV, USA, 2001: 530-535
- [101] Goldberg E, Novikov Y. BerMin: A fast and robust SAT-solver [C]//*Proceedings of Design Automation and Test in Europe (DATE)*. Paris, France, 2002: 142-149
- [102] Selman B, Levesque H, Mitchell D. A new method for solving hard satisfiability problems [C]//*Proceedings of the 10th AAAI'92*. Menlo Park, CA: AAAI Press, 1992: 440-446
- [103] Selman B, Kautz H A, Cohen. Noise strategies for improving local search [C]//*Proceedings of the 12th AAAI'94*. Menlo Park, CA: AAAI Press, 1994: 337-343
- [104] McAllester D, Selman B, Kautz H. Evidence for invariants in local search [C]//*Proceedings of the 14th AAAI'97*. Menlo Park, CA: AAAI Press, 1997: 321-326
- [105] Mazure B, Sais L, Gregoire E. Tabu search for SAT [C]//*Proceedings of the 14th AAAI'97*. Menlo Park, CA: AAAI Press, 1997: 281-285
- [106] Schuurmans D, Southey F. Local search characteristics of incomplete SAT procedures [J]. *Artificial Intelligence*, 2001, 132(2): 121-150
- [107] Prestwich S, Lynce I. Local Search for Unsatisfiability [C]//*Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing*. Seattle, WA, USA, 2006: 283-296
- [108] Audemard G, Simon L. GUNSAT: A Greedy Local Search Algorithm for Unsatisfiability [C]//*Proceedings of the 20th International Joint Conference of Artificial Intelligence*. Hyderabad, India, 2007: 2256-2261
- [109] Biere A, Cimatti A, Clarke E M, et al. Symbolic model checking without BDDs [C]//*Proceedings of the 5th Int'l Conf. on Tools and Algorithms for the Constructions and Analysis of Systems (TACAS'99)*. LNCS 1579, Berlin: Springer-Verlag, 1999: 193-207
- [110] Cimatti A, Clarke E M, Giunchiglia E, et al. NuSMV2: An open source tool for symbolic model checking [C]//*Brinksma E, Larsen K G, eds. Proceedings of the 14th Int'l Conf. on Computer Aided Verification (CAV 2002)*. LNCS 2404, Berlin: Springer-Verlag, 2002: 359-364
- [111] Amla N, Kurshan R, McMillan K, Medel R. Experimental analysis of different techniques for bounded model checking [C]//*Hubert G, Hatcliff J, eds. Proceedings of the 9th Int'l Conf. on Tools and Algorithms for the Constructions and Analysis of*

- Systems(TACAS 2003). LNCS 2619, Berlin; Springer-Verlag, 2003; 34-48
- [112] Cimatti A, Pistore M, Roveri M, et al. Improving the encoding of LTL model checking into SAT[C] // Agostino C, ed. Proceedings of the 3rd Int'l Workshop on Verification, Model Checking, and Abstract Interpretation (VMCAI 2002). LNCS 2294, Berlin; Springer-Verlag, 2002; 196-207
- [113] Frisch A, Sheridan D, Walsh T. A fixpoint encoding for bounded model checking[C] // Aagaard MD, OLeary J W, eds. Proceedings of the 4th Int'l Conf. on Formal Methods in Computer-Aided Design (FMCAD 2002). LNCS 2517, Berlin; Springer-Verlag, 2002; 238-255
- [114] Latvala T, Biere A, Heljanko K, et al. Simple bounded LTL model checking[C] // Hu AJ, Martin AK, eds. Proceedings of the 5th Int'l Conf. on Formal Methods in Computer-Aided Design (FMCAD 2004). LNCS 3312, Berlin; Springer-Verlag, 2004; 186-200
- [115] 杨晋吉, 苏开乐, 骆翔宇, 等. 有界模型检测的优化[J]. 软件学报, 2009, 20(8): 2005-2014
- [116] Jackson P, Sheridan D. Clause form conversions for boolean circuits[C] // Hoos HH, Mitchell DG, eds. Proceedings of the 7th Int'l Conf. on Theory and Applications of Satisfiability Testing (SAT 2004). LNCS 3542, Berlin; Springer-Verlag, 2004; 183-198
- [117] Jackson P, Sheridan D. The optimality of a fast CNF conversion and its use with SAT[C] // Hoos HH, Mitchell DG, eds. Proceedings of the 7th Int'l Conf. on Theory and Applications of Satisfiability Testing (SAT 2004). LNCS 3709, Berlin; Springer-Verlag, 2005; 827-831
- [118] Strichman O. Accelerating bounded model checking of safety properties[J]. Formal Methods in System Design, 2004, 24(1): 5-24
- [119] Yokoo M, Suyama T, Sawada H. Solving Satisfiability Problems Using Field Programmable Gate Arrays; First Results[C] // Proceedings of the Second International Conference Principles and Practice of Constraint Programming. Cambridge, Massachusetts, USA, 1996; 497-509
- [120] Zhong P, Martonosi M, Ashar P, et al. Using Configurable Computing to Accelerate Boolean Satisfiability[J]. IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, 1999, 18(6): 861-868
- [121] Abramovici M, Saab D. Satisfiability on Reconfigurable Hardware[C] // Proceedings of Seventh International Workshop Field-Programmable Logic and Applications. London, UK, 1997; 448-456
- [122] Dandalis A, Prasanna V K. Run-Time Performance Optimization of an FPGA-Based Deduction Engine for SAT Solvers[J]. ACM Trans. Design Automation of Electronic Systems, 2002, 7(4): 547-562
- [123] Sousa J, Marques-Silva J P, Abramovici M. A Configurable/ Software Approach to SAT Solving[C] // Proceedings of the Ninth IEEE International Symposium Field-Programmable Custom Computing Machines, Rohnert Park, California, USA, 2001
- [124] Reis N A, Sousa J. On Implementing a Configurable[C] // Software SAT Solver Proceedings of 10th IEEE International Symposium Field-Programmable Custom Computing Machines. Napa, CA, USA, 2002; 282-283
- [125] Skliarova I, Ferrari A B. A Software/Reconfigurable Hardware SAT Solver[J]. IEEE Trans. Very Large Scale Integration(VLSI) Systems, 2004, 12(4): 408-419
- [126] Kestur, Davis S, Williams J D. BLAS Comparison on FPGA, CPU and GPU[C] // VLSI (ISVLSI). IEEE Computer Society Annual Symposium. Lixouri Kefalonia, Greece, 2010; 288-293
- [127] 周进, 赵希顺. 基于硬件可编程逻辑(FPGA)的 SAT 算法综述[J]. 电子世界, 2012, 6: 61-69
- [128] Luo Zhong-wen, Yang Zheng-ping, Liu Hongzhi, et al. GA Computation of 3-SAT problem on Graphic Process Unit[C] // Proceedings of the International Symposium on Intelligent Computation and its Application. Wuhan, Hubei, 2005; 27-31
- [129] Luo Zhong-wen, Liu Hong-zhi. Cellular genetic algorithms and local search for 3-SAT problem on graphic hardware[C] // Proceedings of 2006 IEEE Congress on Evolutionary Computation. Vancouver, Canada, 2006; 2988-2992
- [130] Armando A, Mantovani J, Platania L. Bounded model checking of software using SMT solvers instead of SAT solvers[J]. International Journal on Software Tools for Technology Transfer, 2009, 11(1): 69-83
- [131] Cordeiro L, Fischer B, Marques-Silva J. SMT-based bounded model checking for embedded ANSI-C software[C] // Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering. Auckland, New Zealand, 2009; 137-148
- [132] Ganai M, Gupta A. Accelerating high-level bounded model checking[C] // ICCAD. San Jose, CA, USA, 2006; 794-801
- [133] 徐亮. 改进的以 SMT 为基础的实时系统限界模型检测[J]. 软件学报, 2010, 21(7): 1491-1502
- [134] Clarke E M, Grumberg O, Long D E. Model checking and abstraction[J]. ACM Transactions on Programming Languages and System(TOPLAS), 1994, 16(5): 1512-1542
- [135] Dams D, Gerth R, Grumberg O. Abstract interpretation of reactive systems[J]. ACM Transactions on Programming Languages and System(TOPLAS), 1997, 19(2): 253-291
- [136] Clarke E M, Grumberg O, Jha S. Counterexample-Guided abstraction Refinement for symbolic Model checking[J]. Journal of the ACM, 2003, 50(5): 752-794
- [137] 袁志斌, 徐正权, 王能超. 软件模型检测中的抽象[J]. 计算机科学, 2006, 33(7): 276-279
- [138] Graf S, Saidi H. Construction of abstract state graphs with PVS[C] // Proceedings of 9th International Conference on Computer Aided Verification. Haifa, Israel, 1997; 72-83
- [139] Colon M, Uribe T. Generating finite-state abstractions of reactive systems using decision procedures[C] // Computer Aided Verification. Vancouver, Canada, 1998; 293-304
- [140] Ball T, Podelski A, Rajamani S K. Boolean and Cartesian Abstraction for Model Checking C Programs[C] // Proceedings of Tools and Algorithms for the Construction and Analysis of Systems. Genova, Italy, 2001; 268-283
- [141] Ball T, Majumdar R, Millstein T, et al. Automatic Predicate Abstraction of C Programs[C] // Proceedings of the Conference on Programming Language Design and Implementation. Snowbird, Utah, 2001; 203-213

馈回路,这就可以大大提高仲裁速度。PPE 仲裁器结构如图 8 所示。

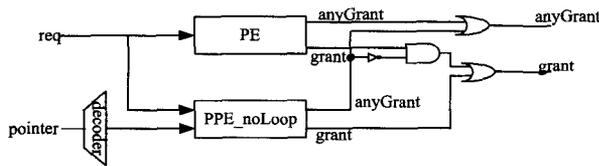


图 8 PPE 仲裁器结构图

PPE 仲裁器工作流程如下:如果在 req[pointer]到 req[N-1]之间没有任何输入请求,那么 PPE 的 grant 输出就等于 PE 的 grant 输出。如果在 req[pointer]到 req[N-1]之间有输入请求,那么 PPE 的 grant 输出就等于 PPE_noLoop 的 grant 输出。很明显 PPE 的输出 grant 就是一个 2 输入选择器的输出,PPE_noLoop 的 anyGrant 信号为选择信号,它用来指示是否有仲裁输出,它是通过将各个比特位或起来得到的。PPE 仲裁器由于是通过组合逻辑来实现的,因此延迟很小,可以满足高速 Round Robin 仲裁器设计的要求。

结束语 缓冲交叉开关相比于无缓冲的交叉开关具有仅需要较简单的调度算法、降低了输入与输出之间的同步要求、更好的性能等优点。Round Robin 调度算法仅需要简单的指针轮转操作,算法的复杂度仅为 $O(1)$,且实现简单,系统的延迟小,因此 Round Robin 调度算法受到了高度的重视。本文根据缓冲交叉开关交换机的特点,结合 Round Robin 算法的优势,设计了一种高速 Round Robin 仲裁器,以满足高速交换机设计的需求。

参考文献

- [1] Magill R B, Rohrs C E, Stevenson R L. Output-queued switch emulation by fabrics with limited memory[J]. IEEE Journal on Selected Areas Communications, 2003, 21(4): 606-615
- [2] Karol M, Hluchyj M, Morgan S. Input versus output queueing on a space division switch[J]. IEEE Transactions on Communications, 1987, 35(12): 1347-1356
- [3] McKeown N. Scheduling Algorithms for Input-Queued Switches [D]. Ph D dissertation, University of California at Berkeley, 1995
- [4] McKeown N, Mekkittikul A, Anantharam V, et al. Achieving 100% throughput in an input-queued switch (extended version) [J]. IEEE Transactions on Communications, 1999, 47(8): 1260-1267
- [5] Nabeshima M. Performance evaluation of a combined input- and crosspoint-queued switch[J]. IEICE Transactions on Communications, 2000, E83-B(3): 737-741
- [6] Rojas-Cessa R, Oki E, Jing Z, et al. CIXB-1: Combined input-once-cell-crosspoint buffered switch [C] // IEEE Workshop on High Performance Switching and Routing. Dallas, USA: IEEE Press, 2001: 324-329
- [7] Chuang S T, Iyer S, McKeown N. Practical algorithms for performance guarantees in buffered crossbars [C] // IEEE INFOCOM'05. Miami, FL: IEEE Infocom, 2005: 981-991
- [8] Berger M S. Delivering 100% throughput in a Buffered Crossbar with Round Robin scheduling [C] // High Performance Switching and Routing. IEEE HPSR, Poznan, Poland, 2006: 5-10
- [9] Lin M, McKeown N. The throughput of a buffered crossbar switch [J]. IEEE Communications Letters, 2005, 5: 465-467
- [10] Yoshigoe K, Christensen K J. An evolution to crossbar switches with virtual output queuing and buffered crosspoints [J]. IEEE Network, 2003, 17(5): 48-56
- [11] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [12] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [13] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [14] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [15] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [16] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [17] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [18] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [19] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [20] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [21] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [22] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [23] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [24] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [25] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [26] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [27] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [28] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [29] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [30] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [31] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [32] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [33] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [34] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [35] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [36] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [37] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [38] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [39] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [40] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [41] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [42] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [43] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [44] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [45] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [46] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [47] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [48] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [49] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [50] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [51] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [52] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [53] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [54] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [55] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [56] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [57] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [58] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [59] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [60] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [61] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [62] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [63] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [64] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [65] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [66] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [67] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [68] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [69] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [70] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [71] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [72] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [73] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [74] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [75] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [76] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [77] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [78] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [79] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [80] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [81] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [82] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [83] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [84] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [85] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [86] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [87] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [88] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [89] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [90] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [91] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [92] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [93] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [94] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [95] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [96] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [97] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [98] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [99] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [100] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [101] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [102] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [103] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [104] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [105] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [106] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [107] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [108] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [109] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [110] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [111] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [112] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [113] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [114] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [115] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [116] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [117] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [118] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [119] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [120] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [121] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [122] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [123] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [124] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [125] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [126] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [127] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [128] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [129] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [130] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [131] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [132] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [133] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [134] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [135] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [136] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [137] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [138] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [139] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [140] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [141] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [142] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [143] 李勇, 罗军舟, 吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法 [J]. 计算机研究与发展, 2006, 43(12): 2033-2040
- [144] Cordeiro L, Fischer B, Marques-Silva J. Continuous Verification of Large Embedded Software using SMT-Based Bounded Model Checking [C] // 17th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems. Oxford, England, 2009: 160-169
- [145] Kong Wei-qiang, Shiraishi, Mizushima, et al. An SMT Approach to Bounded Model Checking of Design in State Transition Matrix [C] // Proceedings of 2010 International Conference on Computational Science and Its Applications (ICCSA). 2010: 231-238
- [146] 周宽久, 杨广, 柳朕, 等. 基于拓扑排序的数据竞争方向定位 [J]. 计算机科学, 2012, 39(11): 108-116
- [147] Wang Jie, Cui Kai, Zhou Kuan-jiu, et al. Programmable NoC Scheduling Based on Multi-core Processor [C] // Proceedings of International Conference on Electrical and Control Engineering (ICECE). 2011
- [148] Lai Xiao-chen, Wang Xiao-liang, Zhou Kuan-jiu, et al. Research on method of static analysis for safety of C++ program [J]. International Journal of Advancements in Computing Technology, 2012, 4(21): 337-345

(上接第 86 页)

- [142] Das S, Dill D L, Park S. Experience with predicate abstraction [C] // Computer-Aided Verification. Trento, Italy, 1999: 160-171
- [143] Lahiri S K, Nieuwenhuis R, Oliveras A. SMT techniques for fast predicate abstraction [C] // Proceedings of 18th International Conference on Computer Aided Verification (CAV). Seattle, USA, 2006: 424-437
- [144] Cavada R, Cimatti A, Franzen A, et al. Computing Predicate Abstractions by Integrating BDDs and SMT Solvers [C] // Seventh International Conference on Formal Methods in Computer-Aided Design. Austin, Texas, 2007: 69-76
- [145] 何炎祥, 吴伟, 陈勇, 等. 基于 SMT 求解器的路径敏感程序验证 [J]. 软件学报, 2012, 23(10): 2655-2664
- [146] Clarke E M, Grumberg O, Jha S, et al. Counterexample-Guided Abstraction Refinement [C] // CAV. Berlin, Heidelberg, 2000: 154-169
- [147] Henzinger T A, Jhala R, Majumdar R, et al. Lazy Abstraction [C] // Proceedings of the 29th Annual Symposium on Principles of Programming Languages. Oregon, Portland, 2002: 58-70