

一种基于可配置随机测试生成的多核验证方法

罗 汉 青, 梁 利 平, 叶 甜 春

(中国科学院 微电子研究所, 北京 100029)

摘 要: 多核设计规模和复杂度的不断提高, 使功能验证变得越来越具挑战性. 通过分析基于模拟仿真的多核验证方法, 提出了一种基于可配置随机测试生成的多核 cache 一致性验证方法. 该方法以随机测试生成为基础, 通过配置随机生成参数来产生特定结构的多核验证指令流. 此指令流的特点是, 通过内存地址访问约束和多核同步操作的设置, 来达成多核系统执行顺序的准确预测, 进而通过自检测指令组的配置来完成自动快速结果比较. 实验结果表明, 该方法对多核一致性的验证是高效的.

关键词: 多核设计; cache 一致性; 指令流配置; 同步设置; 自检测

中图分类号: TP302

文献标识码: A

文章编号: 1000-7180(2014)01-0088-04

A Method Based on Configurable Random Test Generation for Multi-core Verification

LUO Han-qing, LIANG Li-ping, YE Tian-chun

(Institute of Microelectronics of Chinese Academy of Sciences, Beijing 100029, China)

Abstract: As multi-core designs architectural complexity increases, the challenge of verification grows dramatically. Through analyzing the methods of simulation based verification of multi-core designs, in this paper we present a new method based on configurable random test generation to verify multi-core cache coherence. Configurable random test generation is used to generate specified instruction streams by controlling generation variables. The execution sequence of the instruction stream in multi-core design can be predicted exactly by using memory address accessing constraints and synchronization operation. Further more, the self-checking instruction groups are used to check the simulation results automatically. The experiment result shows that our method is efficient.

Key words: multi-core; cache coherence; random instruction stream; synchronization operation; self-checking

1 引言

多核验证中, 存储一致性和 cache 一致性的验证是至关重要的. 针对这两方面, 目前已有不少模拟仿真方法的研究成果. 2002 年, A. Adir 等人^[1]提出了一种产生并行带有碰撞的多核验证程序来进行模拟仿真验证. 其中碰撞指的是多个核多次访问同一个内存地址, 由于不能预测执行顺序, 他们使用多个预测结果不断传播的方法来判断待验证设计执行的正误. 2008 年, I. Wagner 等人^[2]提出了一种验证多核一致性的工具 MCjammer, 以 cache 一致性协议的状态转移为覆盖率度量, 为系统中每个核提供一

个测试生成控制模块, 通过覆盖率分析以及测试生成控制模块的相互配合, 实现一个闭环的自适应验证方法. 2010 年, P. Singh 等人^[3]也提出了基于随机测试生成的多核验证方法, 同样使用 cache 一致性协议的状态转换为覆盖率度量, 通过添加约束控制来弥补随机测试的覆盖率不足, 同时提出了一个根据核的数量、cache 块数量以及状态转移数来估计覆盖率的公式, 用以估计验证进程. 同年, 王朋宇等人^[4]提出了一种线性复杂度的存储一致性验证工具 LCHECK. 该工具利用时间序就验证局部化, 结合表示程序序执行结果的有向图和序关系边, 进行推导和正确性检测.

借鉴上述验证成果,本文提出一种基于可配置随机测试生成的多核验证方法.该方法以可配置随机测试生成 IME-ARTGD 为基础,通过配置可配置参数来实现多核测试程序的产生,并通过同步操作实现多核系统执行顺序的准确预测,进而使用自检机制快速发现并定位设计缺陷.

2 多核随机生成介绍

IME-ARTGD 是本文设计的多核随机测试生成工具,如图 1 所示,它主要包含约束控制、生成引擎和参考模型三部分.约束控制主要用于接收验证人员的配置,生成引擎用于产生随机测试程序,参考模型用于整个随机生成的动态策略以及自检检测机制的设置.IME-ARTGD 可配置参数主要包括指令参数配置、访问内存地址控制、多核同步设置以及多核调度配置.

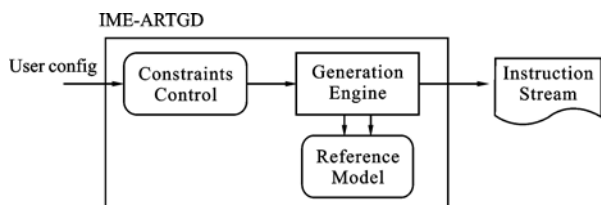


图 1 多核随机生成框图

(1) 指令参数配置:①指令类别配置,根据验证场景的不同,各种指令产生的概率有所不同,多于多核 cache 一致性的验证,指令类别可注重于 load/store 等存储类指令;②特殊指令组的配置,例如具体的同步指令组、检查指令组、地址产生指令组等设置.

(2) 内存访问地址设置:①虚拟存储地址访问区域的设置,主要指可否高速缓存以及可否映射的设置,多核 cache 一致性的访问,则需要多数情况下在可高速缓存的区域访问;②限制访问范围的设置,为覆盖多核 cache 一致性协议状态转换中的每一条转换路径,需要多个核对同一个 cache 块的多次读写,这就需要设置每个核对访问地址的范围从而控制访问频次.

(3) 多核同步设置:①锁变量的配置,通过锁变量的访问来实现多核同步,锁变量存储地址的配置可与内存访问地址的配置相结合,可同时验证原子操作机制和 cache 一致性的多个验证场景;②锁变量的访问机制设置,通过原子操作访问或非原子操作方式;③同步点设置,通过同步点的设置可控制同步代码段的长度、多核访问的频率等.

(4) 多核调度配置:①操作对象的设置,哪个核

发起,哪个核响应;②受调度核的 PC 值.

3 多核验证指令流配置

3.1 多核验证设计框图

图 2 中给出了本文的多核验证方法设计框图.结合 IME-ARTGD 的可配置参数,针对多核系统中的每个核都进行一套相应的配置.根据不同的配置,IME-ARTGD 分别为每个核各自产生一套由指令流组成的测试程序.这些测试程序在统一的地址空间内被分别安放在不重叠的连续空间内.然后将各核的起始 PC 地址调度到该核对应的程序地址空间上.各核在执行各自指令流的同时,本文实现了自检检测机制来帮助验证人员快速发现并定位设计缺陷.该自检检测机制是结合 IME-ARTGD 中参考模型以及指令流的配置来实现,并通过参考模型的实时模拟结果信息,来配置产生相应的结果检查自比较指令,然后由待验证设计自动执行并完成检测.

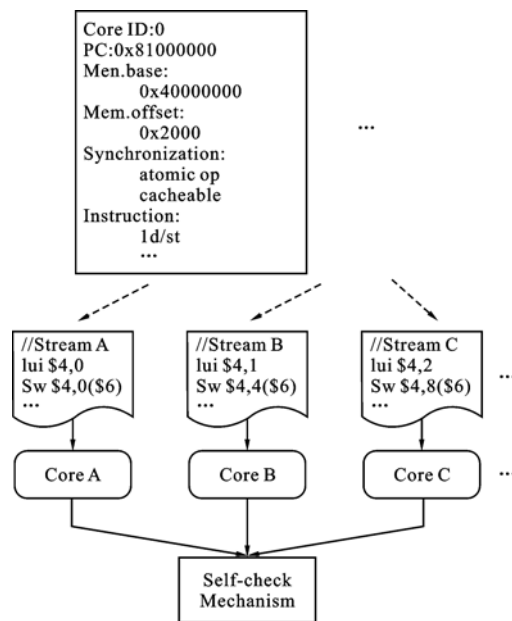


图 2 多核验证设计框图

3.2 指令流配置

本文设计了一套简便的指令流生成方法,使多核系统的运行顺序可以被预测,从而能够准确预测执行结果.图 3 中给出了指令流配置的示意图.该方法有如下三个关键环节:①将每个核的指令流分成若干段,每段代码主要由 load/store 类指令组成,段与段之间设置多核同步操作;②两同步点之间各核的指令流所访问的内存地址范围不重叠,目的是避免多个核在很短的时间内连续访问同一内存地址;③设置锁变量,通过各核对锁变量的读写来实现同步操作.总的说来,该方法是通过内存地址访问顺

序的严格控制来实现对执行结果的预测.

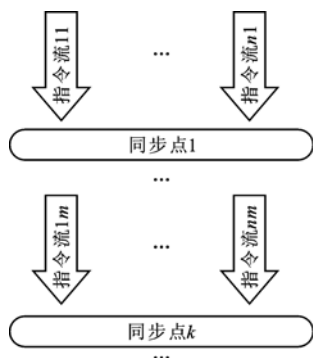


图3 指令流结构

3.3 同步设置

锁变量的设置和操作是同步操作的关键环节, 如果处理不好很可能会产生死锁情况. 根据 IME-ARTGD 提供的对锁变量的配置, 其访问方式可分为一般 load/store 访问和原子操作访问. 如果通过一般 load/store 方式进行访问, 当多个核同时对一个锁变量进行读写操作时, 由于竞争关系而产生写操作丢失, 最终导致死锁. 为避免此情况发生, 需要给多核系统中的每个核分配一个只有本核才能写的锁变量, 然后由其他核来读, 从而避免死锁实现同步. 但这种方法需要使用的变量较多, 当系统中核的数量多时, 会出现读写操作多而不方便管理且效率低的问题.

由于上述原因, 本文更多的使用原子操作访问锁变量. 这种访问方式所需锁变量的个数与核的数量无关, 降低了锁变量的管理复杂度. 使用原子操作实现同步的方法如下: ①设置一个锁变量, 其值为 0; ②各核读该变量, 累加 1, 并写入该变量; ③读该变量, 如果读取值等于核的总个数, 则通过该同步点, 否则重复该步骤. 但当有多个同步点时, 会出现以下问题, 如图 4(a) 所示, A 与 B 是两个核, a 是锁变量, 当 A 和 B 都累加写入后, A 先读到这个值随即清 0, 然而 B 由于某些原因延迟, 还未能读到这个值, 导致 B 一直循环而死锁. 为解决这个问题, 本文设计了一种循环使用 3 个锁变量的同步设置方法. 该方法如图 4(b) 所示, 若当前同步点使用 Lock 0 时, 那么同步之后立刻清除 Lock 2, 下一个同步点使用 Lock 1. 即同步点 n 使用 Lock $(n\%3)$, 并清除 Lock $((n+2)\%3)$, 下一个同步点使用 Lock $((n+1)\%3)$.

3.4 指令流生成流程

结合上述设置, 本文的指令流生成流程如下:

步骤 1: 初始化当前同步点使用的锁变量

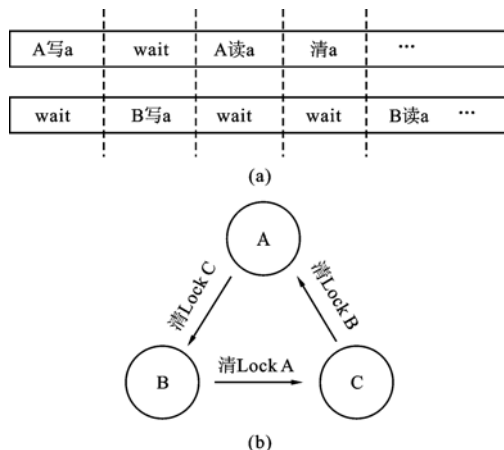


图4 同步设置示意图

Lock n ;

步骤 2: 为多核系统中的每个核分配初始访问地址范围;

步骤 3: 根据指令流的配置分别为每个核产生一段随机指令流;

步骤 4: 使用 Lock n 做同步操作, 读取 Lock n 的值累计 1 并存入 Lock n ;

步骤 5: 延时读取 Lock n 的值;

步骤 6: 判断是否等于核的总个数, 如果是, 则进入步骤 7, 否则, 回到步骤 5;

步骤 7: 清除 Lock $((n+2)\%3)$, 将 $(n+1)\%3$ 的值赋予 n ;

步骤 8: 判断代码长度是否达到配置值, 如果不是, 则回到步骤 2, 否则结束.

3.5 自检测指令流配置

本文使用产生自检测指令的方法来实现自检测机制. 该方法使用比较指令将待验证设计与参考模型提供的实时模拟结果相比较, 来判断待验证设计的执行正误. 控制自检测指令生成的配置参数主要有两部分: 存入内存值的设置以及结果判断指令的设置.

为快速辨别一个值是有哪个核如何存入内存的, 本文对于存入内存的值有特殊的设置. 每次使用 store 类指令存入内存的值被分为高低位两部分: ①高位是执行此操作核的 ID; ②低位是高位指示的核执行的 store 操作的次数. 表 1 中给出了设置以及相应生产指令的示例. Core B 的 ID 是 2, 那么存入值的高位是 0x2000, 根据核的个数可扩大高位的使用位数. 本次 store 操作是该核的第 10 次操作, 那么低位的值就是 0x000a, 最终本次存入内存的值即为 0x2000000a. 通过这样的设置, 当有 bug 发现时, 可以根据存入内存的值快速查出引起 bug 的是

哪个核的第几次 store 操作引起的。

为了能在 bug 出现后尽早发现它,本文对结果判断指令做了相应设置。当随机产生了一条 load 指令后,随即产生一个自检测指令组,以此立即检查从待验证设计的内存或 cache 中读取的值是否正确。表 1 中也给出了自检测指令的配置以及生产的示意指令组,首先配置指令写入寄存器预测的结果值,然后使用寄存器比较指令 bne 来做比较。

表 1 自检测指令配置示例

...	...
//配置	//生成指令
Mem. base:0x81000000	lui \$4,0x8100
Mem. offset:0x20	ori \$4,\$4,0x20
Repeat:	...
lui \$reg←Core ID	lui \$5,0x2000 //Core B
ori \$reg←ST_time	ori \$5,\$5,10//10th store
sw	sw \$5,0(\$4)
...	...
lw	lw \$6,4(\$4)
...	...
Check:	...
lui \$reg←RM_res_h	lui \$7,0x1000//Core A
ori \$reg←RM_RES_L	ori \$7,\$7,8 //8th store
bne \$reg,\$dst,err	bne \$7,\$6,err
...	nop
...	...

4 实验结果

本文提出的方法已经应用于实际多核系统的验证中。如果图 5(a)所示,该多核系统由四个核组成,四个核各自有 cache 并共享主存,cache 一致性协议使用的是写无效的监听式协议,它只含三个状态:共享(Shared)、修改(Modified)和无效(Invalid),其状态转换如图 5(b)所示。

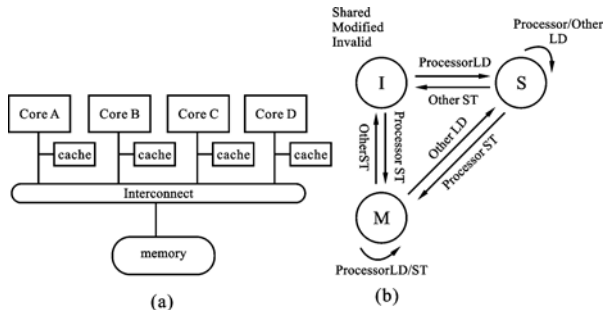


图 5 待验证设计框图及 cache 一致性协议

本文验证方法的有效性主要从系统中每个核的 cache 状态转移覆盖以及总的 cache 状态转移次数来体现。表 2 中给出了四个核 cache 状态转移覆盖

情况,以及达成最终覆盖率所使用的指令条数。本文总共使用了 64 个 cache 块,四个核分别用了 368 条、1 378 条、432 条和 895 条指令就达成了其中一个 cache 块 100% 的状态覆盖。图 6 中分别给出了使用 32 个、6 个、128 个和 256 个 cache 块的情况下,总共运行 20 万条指令,平均每个 cache 块状态转移次数最终分别达到 24 959 次、12 511 次、6 188 次和 3 103 次。在验证过程中,发现很多与 cache 状态转移有关的 bug,例如,多个核对同一个 cache 块有从状态 Modified 到 Invalid 的转移,但最终到 Shared 状态读出的结果不同。自检测指令都在第一时间发现了 bug,验证人员根据核的 ID 和存储次数快速定位了 bug。

表 2 状态转移覆盖情况

待验证核	状态转移覆盖率/%	使用的指令条数
Core A	100	368
Core B	100	1 378
Core C	100	432
Core D	100	895

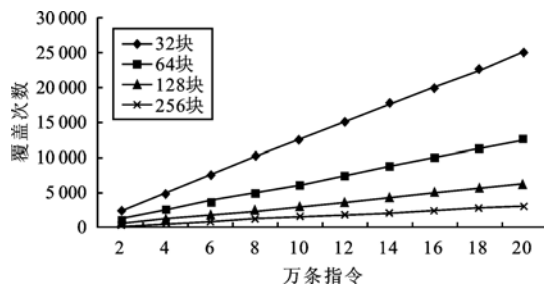


图 6 cache 块覆盖次数

5 结束语

本文提出了一种基于可配置随机测试生成的多核验证方法。对四核系统 cache 一致性的验证的实验数据表明,它不仅可以很好覆盖 cache 一致性协议中的状态转移,并且在发现 bug 时,能够迅速查出出错的方位。与此同时,发现当代码段比较短时,频繁使用同步操作会使指令流的效率变低。在未来的工作中,我们将完善这方面的工作。

参考文献:

- [1] Adir A, Shurek G. Generating concurrent test-programs with collisions for multi-processor verification [C]//proceedings of the 7th IEEE International High-Level Design Validation and Test Workshop. Cannes, France:IEEE,2002:77-82.

(下转第 98 页)

- Decision & Control Manchester Grand Hyatt Hotel. San Diego, CA, USA;IEEE,2006.
- [10] 陈宝云. 密集型 RFID 系统读写器防碰撞算法 DPC-EdiCa 的提出与研究[D]. 广州:华南理工大学,2011.
- [11] Gandino F, Ferrero R, Montrucchio B. Introducing probability in RFID reader-to-reader anti-collision [C]//Eighth IEEE International Symposium on Network Computing and Applications. Italy:IEEE,2009.
- [12] Waldrop J, Engels D W, Sanna S E. Colorwave: a MAC for RFID reader networks[C]// IEEE Wireless Communications and Networking Conference (WCNC). New Orleans, Louisiana, USA;IEEE,2003.
- [13] Ho H, Engels D W, Sarma S E. HiQ: a hierarchical Q-learning algorithm to solve the reader collision problem [C]// International Symposium on Applications and the Internet Workshops. [s. l.]:IEEE,2006.
- [14] Ferrero R, Gandino F, Montrucchio B. A fair and high throughput reader-to-reader anticollision protocol in dense RFID networks[J]. IEEE Transactions on industrial informatics,2012,8(3):697-705.
- [15] Joshi J P, Mamun K M A, Kim S W. A reader anti-collision MAC protocol for dense reader RFID system [C]// International Conference on Communications and Mobile Computing. [s. l.]:IEEE,2009:313-316.
- [16] Eom J B, Yim S B, Lee T J. An efficient reader anti-collision algorithm in dense RFID networks with mobile RFID readers[J]. IEEE Transactions on industrial electronics,2009,56(7):2326-2336.
- [17] 信部无[2007]205 号文件,《关于发布 800/900 频段射频识别(RFID)技术应用试行规定的通知》[S]. 2007.

作者简介:

邓敦建 男,(1986—),硕士研究生.研究方向为射频通信系统、UHF RFID 系统.

阎跃鹏 男,(1963—),博士,研究员.研究方向为通信系统与信息科学.

牟荣增 男,(1972—),博士,研究员.研究方向为无线传感器网络.

(上接第 91 页)

- [2] Wagner I, Bertacco V. MCjammer: Adaptive Verification for Multi-core Designs [C]//Proceedings of the Conference on Design, Automation and Test in Europe. EDA Consortium San Jose, CA, USA: ACM, 2008:670-675.
- [3] Singh P, Landis D L. Test generation for CMP designs [C]//proceeding of Microprocessor Test and Verification (MTV). Austin, TX;IEEE,2010:67-70
- [4] 王朋宇, 陈云霁, 沈海华, 等. 片上多核处理器存储一致性验证[J]. 软件学报, 2010,21(4):863-874.
- [5] Taylor S, Ramey C, Barner C, et al. A simulation-based method for the verification of shared memory in multiprocessor systems [C]//IEEE/ACM ICCAD. [s. l.]:IEEE,2001:10-17.
- [6] Chen X, Yang Y, Gopalakrishnan G, et al. Reducing verification complexity of amulticore coherence protocol using assume/guarantee [C]//proceeding of FMCAD. San Jose, California, USA;IEEE,2006:81-88.
- [7] Meixner A, Sorin D. Dynamic verification of memory consistency in cache-coherent multithreaded computer architectures[J]. IEEE Transactions on Dependable and Secure Computing, 2009,6(1):18-31.
- [8] Manovit C, Hangal S. Efficient algorithms for verifying memory consistency [C]//Proc of the 17th ACM Symp on Parallelism in Algorithms and Architecture. Las Vegas, Nevada, USA;ACM,2005:245-252.

作者简介:

罗汉青 男,(1985—),博士研究生.研究方向为处理器验证与测试.

梁利平 男,(1969—),博士生导师.研究方向为高性能数字信号处理器、微处理器、高速混合信号集成电路设计技术.

叶甜春 男,(1965—),博士生导师.研究方向为集成电路制造技术、新型器件及微细加工技术.