文章编号:1674-2974(2018)10-0108-07 DOI:10.16339/j.cnki.hdxbzkb.2018.10.015

Cache一致性验证的结构化激励生成算法*

程开丰,罗汉青,梁利平†
(中国科学院大学 中国科学院微电子研究所 嵌入式与多核DSP实验室,北京 100029)

摘 要:为解决Cache一致性验证中传统随机激励方法的冗余覆盖及覆盖死角等问题,提出了一种高层次结构化激励生成算法和相应的高层次功能覆盖率模型.首先根据实际多核应用场景将冲突访存操作分类成基本同步和复杂同步,并进一步抽象成有向二分图模型,由此提出一种通用的层次化输入空间等价类划分算法和对应的高层次HSPC(Host Slave Pair Coverage)功能覆盖率模型,最后基于树的搜索提出了结构化激励生成算法.上述方案成功应用于IME-Diamond SoC的Cache一致性的功能验证中,实际结果表明,相比传统基于代码的覆盖率,高层次HSPC功能覆盖率模型的揭示功能Bug能力更强,而且相对于传统的随机生成,结构化的激励能够将覆盖率收敛所需的激励数减少96.3%.

关键词:Cache一致性;有向二分图模型;等价类划分;高层次功能覆盖率模型;结构化激励生成

中图分类号:TP302 **文献标志码**:A

Structural Stimuli Generation Algorithm for Cache Coherence Verification

CHENG Kaifeng, LUO Hanqing, LIANG Liping†
(University of Chinese Academy of Sciences, Institute of Microelectronics, Beijing 100029, China)

Abstract: In order to deal with the problems of redundant and failed coverage in the traditional random stimuli for cache coherence verification, a high level structural stimuli generation algorithm and the corresponding functional coverage model were presented. Firstly, conflict memory accesses were categorized into basic/complex synchronizations, and thus abstracted into general bipartite graph model. Consequently, a general layered equivalence partition algorithm of ISS and the corresponding high level HSPC (Host Slave Pair Coverage) functional coverage model were proposed. Finally, two structural stimuli generation algorithms based on the search of ISS tree were presented. Experiments were performed in the functional verification of the cache system of IME-Diamond SoC, and the result indicates the HSPC coverage model can help uncover functional bugs more easily when compared with code coverage, and structural stimuli generation can reduce 96.3% stimuli for coverage convergence when compared with random stimuli generation.

Key words: Cache coherence; directed bipartite graph model; equivalence partition; high level functional coverage model; structural stimuli generation

---

* **收稿日期**:2017-09-28
基金项目:新一代宽带无线移动通信网国家科技重大专项(2013ZX03003015)
作者简介:程开丰(1990—),男,安徽黄山人,中国科学院博士研究生
† 通讯联系人,E-mail:liangliping@ime.ac.cn

所有基于仿真的功能验证都需要做好两个环节:输入激励的生成与最后结果正确性的判断. Cache 一致性的验证是多处理器系统功能验证中非常重要的一个部分. 其输入激励是多核并行访存程序,结果正确性的判断是基于特定的内存一致性模型(MCC,Memory Consistency Model).

在已有的相关研究中,1979 年 Lamport[1] 提出了顺序一致性模型(Sequential Consistency, SC),由于 SC 模型最符合程序员的直观思维,所以在多处理器系统设计和验证中一直占主导地位. 1994 年 Gibbons 和 Korach[2] 在理论上分析了存储一致性验证的复杂性,其中把在 SC 模型下的验证定义为 VSC(Verifying Sequential Consistency)问题,并证明了 VSC 是 NP 难问题. 1997 年胡伟武等[3] 给出了另一种 SC 模型下判断多核访存操作正确性的方法. 与 Gibbons 等只根据访存结果来寻找判断是否存在多核顺序交织序列不同,该方法关注记录了多核程序 PRG 的静态程序 PRO(PRG)和动态执行序 E(PRG),并指出最后执行结果符合 SC 的充要条件是 E(PRG)∪PRO(PRG)对应的有向图无环. 上述研究本质都是关于 Cache 一致性结果正确性判断,缺少对输入多核程序生成方法的研究.

1990 年 Wood 等[4] 采用了仿真的手段对多核一致性协议进行了验证,其中多核访存程序采用的是随机生成. 2010 年王朋宇等[5] 在文献[3]的工作基础上实现了一种线性时间复杂度的片上多核处理器存储一致性验证工具 LCHECK,其中多核访存程序生成部分仍采用随机的策略. 可见现有对 Cache 一致性验证时的激励基本都基于随机生成,随机化方法虽然有时能够发现一些 corner bug,但有两个明显的不足:

1)现实中输入激励空间(ISS,Input Stimuli Space)的元素并不是均匀分布的,而随机产生输入激励时并不考虑此等情况,故最后对 ISS 的覆盖是不均匀的,即简单的激励点会被反复覆盖,而困难的激励点则很难甚至无法覆盖;

2)多核相对单核最大的特点是允许无冲突程序间的并行执行,以提高整体性能,故绝大多数实际的多核应用程序不会随意进行 Cache 一致性操作,而是在特定情形下有规律的进行. 随机产生的多核程序也没有考虑此实际情况,本文在考虑了实际的多核应用特点下,针对 Cache 一致性的功能验证提出了一套通用的 ISS 高层次抽象和等价类划分方法,并提出了高层次的 HSPC(Host Slave Pair Coverage)功能覆盖率模型以及基于树遍历的结构化激励生成算法,最后通过实际的工程检验证明:相比传统的基于 RTL 代码的覆盖率模型,HSPC 功能覆盖率模型能更好地帮助发现功能 bug;相比传统随机生成验证激励,同等覆盖率下,结构化激励能够大幅减少测试激励数目,缩短验证周期.

1　具体方案

1.1　有向二分图模型

在 Cache 一致性系统的仿真验证中,其输入激励为多核冲突访存程序. 由于多核相对单核最大的特点是允许无冲突程序间的并行执行,以提高整体性能. 故对实际的多核 Cache 一致性操作的应用场景有如下基本假设:

A1)多核冲突访存程序一般应用于多核间特定节点的同步;

A2)多核同步可分为基本同步(Basic Synchronization,BS)与复杂同步(Complex Synchronization,CS),其中 CS 可以分解为多个 BS 操作;

A3)BS 本质上为单写多读(Single Write Multiple Read,SWMR)的多核冲突访存程序,且要求 BS 中所有读操作最后要能读到 BS 中写操作的值;

A4)同一 CS 中包含的各 BS 之间访存地址不同;但对具体的访存地址的值和数据不敏感.

A5)不同核之间可以区分,即不具有对称性.

A6)有意义的程序中任何核的任何有意义的写操作都会有至少一个读操作与其对应.

上述假设是合理的,因为以下两点:

1)基于 snoopy 的 MESI 等 Cache 一致性协议在设计时就并不关心绝对的访存地址的值,而只关心多核访存地址间是否冲突等相对关系.

2)多核系统中一般都会有主核与从核之分,而且不同核与 snoopy 等一致性模块间的物理延时一般也是不同的,所以不同核之间是可以区分的.

由此可将多核 CS 操作抽象成有向二分图模型(DBGM, Directed Bipartite Graph Model) $G = \langle V_w, V_r, E \rangle$,其中图的各元素含义如下.

1)主点集 V_w:发起写操作的内核抽象成顶点后构成的集合;

2)从点集 V_r:发起读操作的内核抽象成顶点后构成的集合;

3)点集 V:所有参与访存操作的内核抽象成顶点后构成的集合,即 $V = V_w \cup V_r$;

4)有向边集 E:V_w 中顶点 vw_i 到 V_r 中顶点 vr_j 之间存在有向边 $e_{ij}=\langle vw_i,vr_j\rangle$ 的充要条件是 vr_i 对应内核发起的读操作的结果来自 vw_i 对应内核发起的写操作.一个示例 CS 的有向二分图模型如图 1 所示.

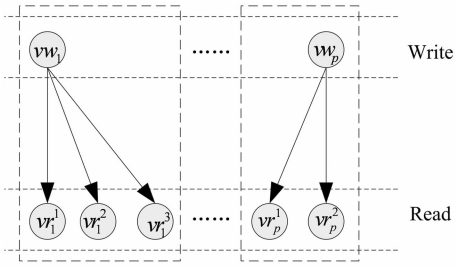

图 1 示例 CS 有向二分图
Fig. 1 DBGM of example CS

其中该 CS 共包含 p 个 BS 操作,每个 BS 都是个全连通子树.

1.2 抽象激励模型

基于访存的有向二分图模型,其激励可以抽象成如下形式:

1)基本同步激励 BsSti:任何时间单个基本同步操作发起的一次激励,对应有向二分图 G 中的一个全连通子树.

2)系统激励 SysSti:任意时刻加载到整个 Cache 系统的激励,是所有 BsSti 的集合,即对应有向二分图 G 中有向边的集合 $SysSti=\{e_{ij}\}$.

1.3 ISS 等价类划分

基于上述抽象后的系统模型和激励模型,可以进行高层次的 ISS 划分,在介绍具体的划分算法之前,先定义几个重要的参数.

1)NC(Number of Cores):系统的核数目;

2)NAHC(Number of Active Host Cores):当下系统活跃的主核数,即发起写操作的核数.很明显 NAHC 的取值范围为:

$$1\leqslant \mathrm{NAHC}\leqslant \mathrm{NC} \tag{1}$$

对 ISS 的等价类划分采用层次化的思想,算法流程如下:

第 1 层:对整个 ISS 进行划分,划分的依据是每个子集的 BS 数(即活跃主核数)相同.根据式(1),ISS 可被划分成 NC 个子集,即 $EP(ISS)=\{S_i\mid 1\leqslant i\leqslant \mathrm{NC}\}$.

第 2 层:对于每个集合 S_i 进行划分,划分的依据是每个子集具有相同的活跃主核.根据基本假设 A5 可知,由于不同核之间可以区分,所以每个 S_i 可被划分成 C_{NC}^i 个子集,即 $EP(S_i)=\{S_{i,j}\mid 1\leqslant j\leqslant C_{NC}^i\}$.

第 3 层:对于每个集合 $S_{i,j}$ 进行划分,划分的依据是每个子集具有相同的活跃从核划分.由于不同主核之间的从核互不冲突,即任意从核只响应一个主核的请求.此时该问题就可等价成集合的子集划分问题,即将 NC 个核构成的集合划分成 i 个不空且互不相交的子集.根据排列组合规律,该过程有 $\{{}_i^{NC}\}$ 种可能性($\{{}_i^{NC}\}$ 是第 2 类 Stirling 数,此处采用的是 Donald Knuth 标记[6]),所以 $S_{i,j}$ 可被划分成个 $\{{}_i^{NC}\}=\frac{1}{i!}\sum_{p=0}^{i}(-1)^p({}_p^i)(i-p)^{NC}$ 个子集,即 $EP(S_{i,j})=\{S_{i,j,k}\mid 1\leqslant k\leqslant \{{}_i^{NC}\}\}$.

第 4 层:对于每个集合 $S_{i,j,k}$ 进行划分,划分的依据是每个子集具有确定的主从核对应关系.基于前述 A4“不同 BS 的从核之间的无冲突”假设和 A5“不同核之间的可区分”假设,由于第 3 层已经确定了参与读操作的从核划分关系(即构建了元素个数为 i 的从集合),但并未确定每组读操作对应的写操作源,所以该层要确定“元素个数为 i 的主核集合”到“元素个数为 i 的从核集合”之间的双射对应关系,显然所有可能的对应数为全排列数 $i!$,所以 $S_{i,j,k}$ 可被划分成 $i!$ 个子集,即 $EP(S_{i,j,k})=\{S_{i,j,k,l}\mid 1\leqslant l\leqslant i!\}$.

经过上述划分,最后可以得到一棵 4 层的输入激励空间树(ISST,Input Stimuli Space Tree).此时 ISST 的每个叶子节点就代表了一个 SysSti.所有的叶子节点数 LeafNum 与核数 NC 之间的关系可以由下面的公式给出:

$$\mathrm{LeafNum}=\sum_{i=1}^{NC}({}_i^{NC})i!\{{}_i^{NC}\}=NC^{NC} \tag{2}$$

可见 LeafNum 与 NC 之间是指数增长关系.

从理论上说 ISST 还可以根据访存的地址继续划分第 5 层,但是本文停留在第 4 层主要考虑到以下两点:

1)根据基本假设 A4 和 A5,对于多核 Cache 访问,最重要的就是要确定由哪些核发起了同步变量的写,以及由哪些核去读取同步变量的值,这些信息被完全包含在上述 4 层 ISST 树的叶子节点中(ISST 树每一层的信息是逐步细化的,第 1、2 和 3 层节点都仅包含局部信息);

2)如果 A4 中关于具体地址不敏感的假设不成立,则记单核处理器的地址位宽为 W,那对应可能的地址数为 2^W.假设 Cache 访问对具体访存地址

是敏感的,则需要对 ISST 第 4 层的每个节点都继续扩展出至少 N 个节点以获得第 5 层,此时得到的 ISST 树的 LeafNum 将至少是 $2^W * NC^{NC}$ 规模,考虑到目前常用处理器的 W 至少是 32,则 LeafNum 的规模是巨大的,有限的时间内很难被全部遍历.

下面以 NC = 2 最简单的多核系统 $V = \{CoreA, CoreB\}$ 为例,具体说明上述算法的流程.

第 1 层:由于 NC=2,则可以将 ISS 划分成两个等价类,即 $EP(ISS) = \{S_1, S_2\}$,其中 S_1 中所有 NAHC 为 1,即表示 SysSti 中只包含一个基本同步操作;S_2 中所有 NAHC 都为 2,即表示 SysSti 是由 2 个基本同步构成的复杂同步操作.

第 2 层:可以将 S_1 划分成两个等价类,即 $EP(S_1) = \{S_{1,1}, S_{1,2}\}$,其中 $S_{1,1}$ 中 BS 的写都是由 CoreA 发起的,$S_{1,2}$ 中的 BS 都是由 CoreB 发起的;可以将 S_2 划分成一个等价类,即 $EP(S_2) = \{S_{2,1}\}$,$S_{2,1}$ 中 SysSti 中第一个 BS 的写由 CoreA 发起,第二个 BS 的写由 CoreB 发起.

第 3 层:由于$\{{}_1^2\}=1$,即所有核都发起同一读操作时参与的从核集只有一种可能性,为 $\{\{CoreA, CoreB\}\}$,故可以将 $S_{1,1}$ 和 $S_{1,2}$ 都划分成一个等价类,即 $EP(S_{1,1}) = \{S_{1,1,1}\}$,$EP(S_{1,2}) = \{S_{1,2,1}\}$,其中 $S_{1,1,1}$ 表示 CoreA 发起的 BS 的目标从核为 $\{CoreA, CoreB\}$,$S_{1,2,1}$ 表示 CoreB 发起的 BS 的目标从核为 $\{CoreA, CoreB\}$;由于$\{{}_2^2\}$,即两个核发起两个不同的读操作时参与的从核集只有一种可能性,为 $\{\{CoreA\}, \{CoreB\}\}$,故可以将 $S_{2,1}$ 划分成一个等价类 $EP(S_{2,1}) = \{S_{2,1,1}\}$,其中 $S_{2,1,1}$ 表示两个核都各自发起一个 BS,而两者的目标读操作核其中一个是 CoreA,另一个是 CoreB.

第 4 层:由于 1! = 1,故 $S_{1,1,1}$ 和 $S_{1,2,1}$ 都可以被划分成一个等价类,即 $EP(S_{1,1,1}) = \{S_{1,1,1,1}\}$ 和 $EP(S_{1,2,1}) = \{S_{1,2,1,1}\}$,其中 $S_{1,1,1,1}$ 对应的 SysSti 为 $\langle A, \{A, B\}\rangle$,即 CoreA 发起 BS 写,CoreA 和 CoreB 发起相应 BS 读;$S_{1,2,1,1}$ 对应的 SysSti 为 $\langle B, \{A, B\}\rangle$,即 CoreB 发起 BS 写,CoreA 和 CoreB 发起相应 BS 读;由于 2! = 2,故 $S_{1,1,1}$ 可以被划分成两个等价类,即 $EP(S_{2,1,1}) = \{S_{2,1,1,1}, S_{2,1,1,2}\}$,其中 $S_{2,1,1,1}$ 对应的 SysSti 为 $\{\langle A, A\rangle, \langle B, B\rangle\}$,即表示 CoreA 发起 BS 写,CoreA 发起相应 BS 读,CoreB 发起另一个 BS 写,CoreB 发起相应 BS 读;$S_{2,1,1,2}$ 对应的 SysSti 为 $\{\langle A, B\rangle, \langle B, A\rangle\}$,即表示 CoreA 发起 BS 写,CoreB 发起相应 BS 读,CoreB 发起另一个 BS 写,CoreA 发起相应 BS 读,最后得到的 ISST 如图 2 所示.

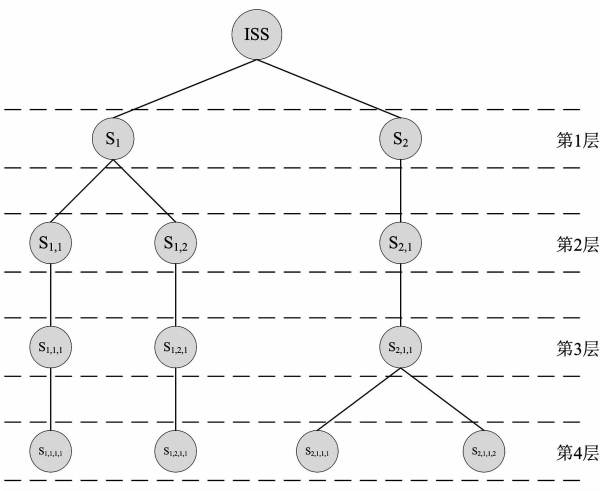

图 2　示例两核系统的 ISST
Fig. 2　ISST of example two core system

高层次功能覆盖率模型

在 ISST 的基础上可以定义一种高层次功能覆盖率模型——主从对覆盖率(Host Slave Pair Coverage,HSPC),模型的三元素定义如下:

1)目标覆盖点集 $\text{Target}_{\text{HSPC}}$:该集合为 ISST 所有叶子节点构成的集合,即 $\text{Target}_{\text{HSPC}} = \bigcup_{i,j,k} EP(S_{i,j,k})$.

2)已覆盖点集 $\text{Covered}_{\text{HSPC}}$:如果一个 SysSti 加载到了 DUT 中,其对应的 ISST 叶子节点为 $S_{i,j,k,l}$,则称 $S_{i,j,k,l}$ 被覆盖,并入 $\text{Covered}_{\text{HSPC}}$,即 $\text{Covered}_{\text{HSPC}} = \text{Covered}_{\text{HSPC}} \cup \{S_{i,j,k,l}\}$.

3)覆盖率值 $\text{Coverage}_{\text{HSPC}}$:$\text{Coverage}_{\text{HSPC}} = \dfrac{|\text{Covered}_{\text{HSPC}}|}{|\text{Target}_{\text{HSPC}}|}$.

1.4　结构化激励生成

结构化激励生成算法分成如下两步:

步骤 1:对 ISST 的叶子节点进行搜索遍历以确定 SysSti 中各活动主从核及相应的主从对应关系;

步骤 2:根据基本假设 A4,SysSti 中 BS 对具体访存地址是不敏感的,故当遍历得到每个叶子节点后,既得到了 Cache 一致性验证激励所需要的三大核心信息:活跃主核、活跃从核划分以及主从之间的对应关系,然后随机产生访存地址及访存数据等细节信息以生成最终的激励.

上述激励生成算法产生的激励不仅保留了底层逻辑信号级随机性,而且能在高层次上充分保证对 ISS 的高覆盖率.

下面重点介绍第一步,即对 ISS 树的搜索遍历.

在介绍具体的搜索算法之前,首先介绍一个重要的转换关系.根据 1.3 节中的划分算法,每个叶节点 $S_{i,j,k,l}$ 可以由其下标四元组 $\langle i,j,k,l\rangle$ 来唯一索引确定,故叶子节点的搜索遍历就转换成了对 $\langle i,j,k,l\rangle$ 四元组的搜索遍历.

由于经典计算机算法中对树的搜索有深度优先搜索(DFS,Depth First Search)和宽度优先搜索(BFS,Breadth First Search)两大类,此处可以加以借鉴,但与经典树搜索需要遍历所有节点不同的是,此 ISS 树只需要搜索叶子节点,所以相应的搜索算法在细节上也有些许不同,具体分别介绍如下.

1.4.1 DFS

其基本思想从激励角度看是个严格的顺序关系,每种特定的主接口组合要驱动完所有可能的从接口,然后再切换到下一种主接口组合;而且对于主接口组合,先考虑单一主接口,然后是 2 个主接口同时发请求,再是 3 个主接口同时,一直到最后所有主接口同时发请求.从 ISS 树角度看,第 3 层所有叶子节点按照从左到右的顺序被遍历,具体算法实现的流程如图 3 所示.

其中 NumChild(S)函数返回节点 S 的儿子节点数目.

1.4.2 BFS

其基本思想从激励角度看是个循环关系,每个循环内部每种特定的主接口组合驱动完一种可能的从接口后就切换到另一主接口组合,直到所有的主接口组合都被遍历到.从 ISS 树角度看,S_1,S_2,…,S_{NHI} 的叶子节点交替地被遍历,具体算法实现的流程如图 4 所示.

其中 CycInc(x)函数是对变量 x 进行循环递增(当 x 超过最大值时,回归到最小值);Searched(S)函数返回 1 时,表示以 S 为根节点的子树已经被搜索,否则返回 0;Dead(S)返回 1 时表示以 S 为根节点的子树中所有叶子节点都已经被遍历,否则返回 0.

2 验证实例

为具体展示上述激励方案的效果,本文选择了 IME-Diamond 系统作为功能验证实例.

IME-Diamond

IME-Diamond 是中科院微电子研究所自主研发的可扩展高性能多核嵌入式 SoC[7-8],其 DCache 之间采用了基于 snoopy 的 MESI 协议进行一致性的维护.

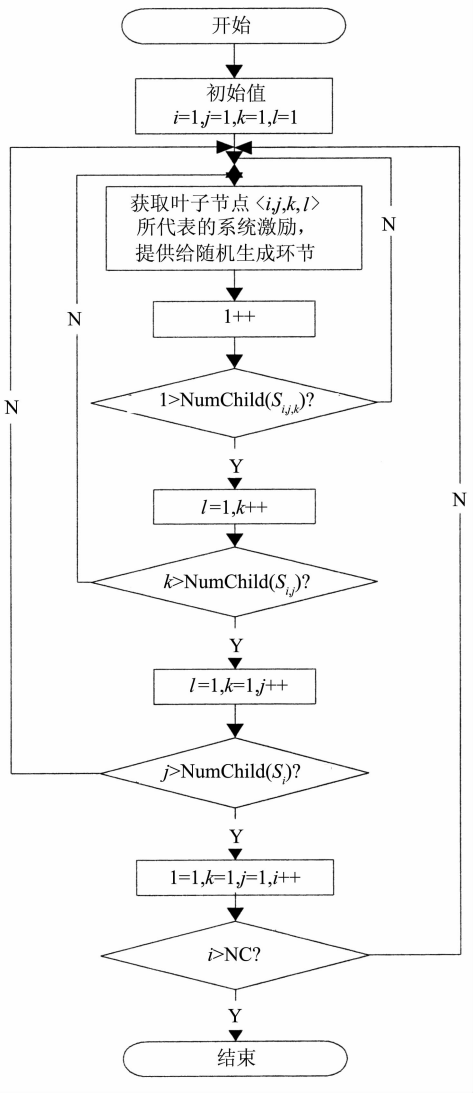

图 3 ISS 树 DFS 实现

Fig. 3 DFS implementation of ISS tree

为比较不同覆盖率模型,帮助揭示 Bug 的能力,实验对比了 Line、Condition 和 Toggle 等基于代码的覆盖率和本文所提出的高层次 HSPC 功能覆盖率,各覆盖率值与揭示的功能 bug 数间关系.考虑到多核 Cache 访问时,最主要的功能缺陷来自多核竞争和异核交互(即多个核有写操作以及一个核读取另一核写的值),所以实验时的 bug 模型也模拟了这种错误,分别在多核竞争和异核交互时引入缺陷(比如多个核有写操作时仲裁错误以及一个核读取另一核写的值时错误).最终的实验结果统计如图 5 所示.

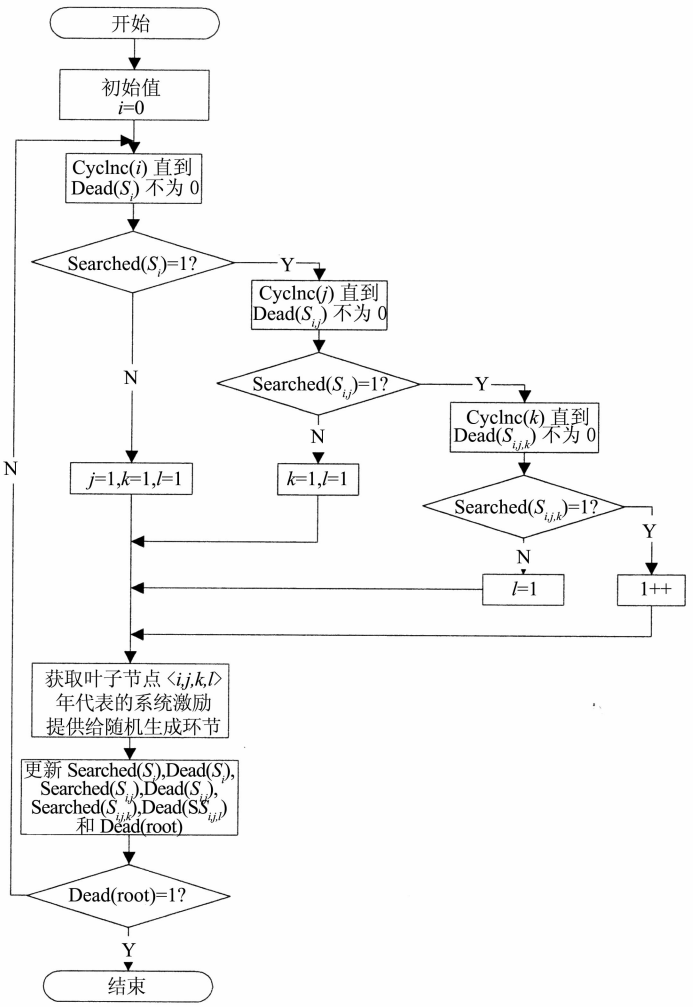

图 4　ISS 树 BFS 实现

Fig. 4　BFS implementation of ISS tree

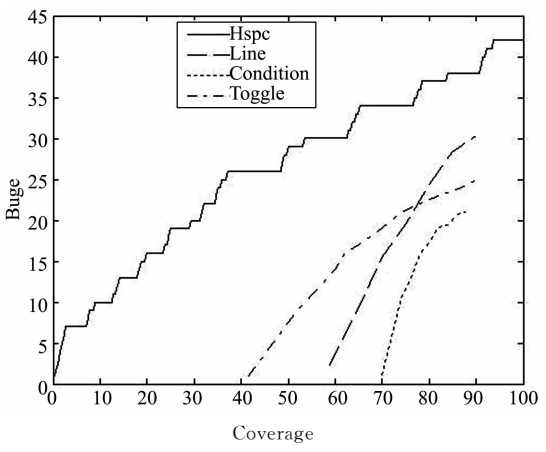

图 5　不同覆盖率模型与揭示 bug 数关系

Fig. 5　Uncovered bug numbers v. s different coverage models

由图 5 可以有以下结论:

1)相对所有的代码覆盖率,HSPC 都能够在很小的覆盖率下发现同样数目的功能 bug;

2)代码覆盖率很难达到 100%,而 HSPC 却能够达到 100%;

3)最终覆盖率收敛时,HSPC 指导下发现的功能 bug 数要多于代码覆盖率. 综上可见,HSPC 揭示功能 bug 的能力比传统的代码覆盖率要更强大.

为比较不同的激励生成算法所产生激励与功能覆盖率之间的关系,实验比较了不同核数 NC 下为达到相同 100% 的 HSPC 覆盖率,本文结构化方法所需激励数 StructStiNum 相对传统随机方法所需激励数 RanStiNum 减少的比例 ReduceRatio=1−$\frac{\text{StructStiNum}}{\text{RanStiNum}}$,如图 6 所示.

由图 6 可见,随着核数的增加,ReduceRatio 一直呈上升趋势,即随机化方法产生的激励中冗余激励的比例越来越高. NC=8 时,ReduceRatio 已经达到了 96.3%.

此外在 16 核 Intel Xeon E5520、主频2.27 GHz、

内存 25 GB、Linux 系统环境下，LeafNum 及激励生成时间(StrucTsti 为结构化激励时间，RanTsti 为随机激励时间)与 NC 之间的关系统计如表 1 所示.

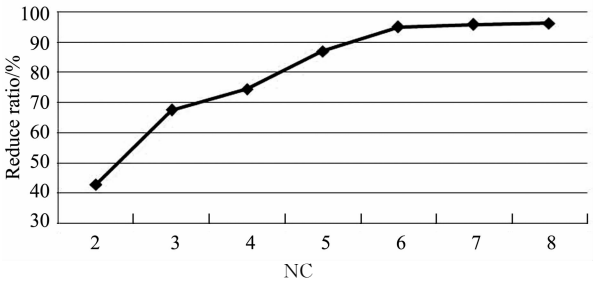

图 6 Reduce ratio 与核数 NC 间关系
Fig. 6 Reduce ratio v. s NC

表 1 LeafNum 及激励生成时间与核数关系
Tab. 1 LeafNum and Tsti v. s NC

NC	LeafNum	StrucTsti/s	RanTsti/s
2	4	0.000 043	0.000 053
3	27	0.000 138	0.000 98
4	256	0.000 547	0.011
5	3125	0.006 9	0.18
6	46 656	0.101	4.9
7	823 543	1.98	112
8	16 777 216	44	3 883

由表 1 可有以下结论：

1)激励生成时间与 LeafNum 一样，与 NC 之间都是指数增长关系，符合公式(2).

2)在 NC≤8 时结构化激励生成时间最大也就 44 s，而随机激励已经到了 3 883 s，是结构化方法的 87 倍.

3)本文的方法适合核数 NC 不是很大的多核系统，当 NC 超过 8 时，为了控制激励生成时间，需要采取进一步的输入空间化简方法.

3 结 语

为解决传统多核处理器 Cache 一致性验证中随机激励方法的冗余覆盖及传统代码覆盖率模型揭错率低等问题，本文根据多核应用程序本身的特点，提出了一种多核冲突访存抽象建模的方法，并由此进一步给出了相应的高层次 HSPC 功能覆盖率模型和结构化激励生成算法. 实际系统的验证效果表明相比传统的代码覆盖率，HSPC 覆盖率模型的揭示功能 Bug 的能力更强；而且相对于传统的纯随机激励，结构化的激励消除了冗余，能够更快地加快覆盖率收敛.

此外对于单核发起多个写操作的情形，由于物理上不同的写也是顺序发起的，根据前述基本假设 A6，每个写都有各自对应的读操作，所以可以根据不同写发起的时间先后，将不同的写划归入不同的时间片，每个时间片内每个核只有一个写操作，此时本文的方法仍然适用.

参考文献

[1] LAMPORT L. How to make a multiprocessor computer that correctly executes multiprocessor programs [J]. IEEE Transaction on Computers, 1979, C－28(9): 690－691.

[2] GIBBONS P B, KORACH E. On testing cache-coherent shared memories [C]// ACM Symposium on Parallel Algorithms and Architectures. 1994: 177－188.

[3] 胡伟武，夏培肃. 顺序一致共享存储系统中的乱序执行技术-基本理论[J]. 计算机学报，1997，20(6)：481－490.
HU W W, XIA P S. Out-of-order execution in sequentially consistent shared memory systems: principles [J]. Chinese Journal of Computers, 1997, 20(6): 481－490. (In Chinese)

[4] WOOD D A, GIBSON G A, AND KATZ R H. Verifying a multiprocessor cache controller using random test generation [J]. IEEE Design and Test, 1989, 7(4): 13－25.

[5] 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.
WANG P Y, CHEN Y J, SHEN H H, *et al*. Memory consistency verification of chip multi-processor[J]. Journal of Software, 2010, 21(4): 863－874. (In Chinese)

[6] GRAHAM R L, KNUTH D E, PATASHNIK O. Concrete mathematics[M]. USA: Addison-Wesley, 1994: 257－267.

[7] 王志君，梁利平，吴凯，等. 一种面向多媒体和通信应用的处理器指令集及架构实现[J]. 湖南大学学报(自然科学版)，2014，41(10)：108－114.
WANG Z J, LIANG L P, WU K, *et al*. Architecture and implementation of an application specific instruction set for multimedia and communication [J]. Journal of Hunan University (Natural Sciences), 2014, 41(10): 108－114. (In Chinese)

[8] 王志君，梁利平，洪钦智，等. 一种 DSP 和通用 CPU 一体化的处理器架构及其 4 核实现[J]. 微电子学与计算机，2014，31(10)：32－38.
WANG Z J, LIANG L P, HONG Q Z, *et al*. The architecture of an unified DSP plus general-purpose CPU and the implementation of a 4-core homogenous processor[J]. Microelectronics & Computer, 2014, 31(10): 32－38. (In Chinese)