文章编号:1674-2974(2018)04-0148-07 DOI:10.16339/j.cnki.hdxbzkb.2018.04.019

面向总线系统的高层次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 微电子研究所,北京 100029)

摘 要:为了应对大规模设计中逻辑信号级输入激励空间爆炸的问题,针对总线系统提出了一种高层次结构化激励生成算法和相应的功能覆盖率模型.首先将总线系统抽象成通用有向二分图模型,然后建立相应激励的高层次数学模型,由此提出一种通用的层次化输入激励空间等价类划分算法和对应的高层次功能覆盖率模型,最后基于树的搜索提出了2种结构化激励生成算法.上述方案成功应用于IME-Diamond SoC的总线系统的功能验证中,实际结果表明,相比代码覆盖率,高层次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,结构化的激励生成能够将覆盖率收敛所需的激励数减少96%.

关键词:总线系统;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

Bus System Oriented High Level Structural Stimuli Generation Algorithm

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

Abstract: In order to deal with the problem of the explosion of the signal level input stimuli space in the massive design, a high level structural stimuli generation algorithm and corresponding functional coverage model targeted at bus system are presented. Firstly, the bus system is abstracted to be a general bipartite graph model (BGM), and then, the mathematical model of the stimuli is established. Consequently, a general layered equivalence partition algorithm of ISS and the relevant high level functional coverage model are proposed. Finally, two structural stimuli generation algorithms based on the search of ISS tree are presented. Experiments are performed in the functional verification of bus system of IME-Diamond SoC. The result indicates that the high level functional coverage model can help uncover functional bugs more easily when compared to code coverage, and structural stimuli generation can reduce 96% stimuli for coverage convergence when compared to random stimuli generation.

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

* **收稿日期**:2017-08-17
基金项目:国家科技重大专项项目(2013ZX03003015), National Science and Technology Major Program of China(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: chengkaifeng@ime.ac.cn

由于现在集成电路的集成度越来越高,电路系统逻辑信号级的输入激励空间(ISS,Input Stimuli Space)规模随之以指数关系增长,激励生成时对逻辑信号级的 ISS 进行完备的穷举遍历变得越来越不可能,所以如何在有限的验证时间内对 ISS 进行较完备和均匀覆盖的激励生成已经成了广泛研究的热门.

在已有的相关研究中,文献[1]指出面对如此巨大的 ISS,目前最常用的应对方法是随机搜索,随机方法实现简单,但是其无法避免冗余激励的产生,而且 ISS 规模越大,随机方法产生的冗余激励越多. 文献[2]指出等价类划分的方法可以很好地解决避免纯随机方法的产生冗余激励的弊端,其基本思想是将整个 ISS 根据一些性质划分成若干子集,然后从各个子集中选取若干元素共同构成最后输入待验证设计(DUT,Design Under Test)的激励集合,但是不同的 DUT 性质各不相同,其没有给出通用的划分性质. 文献[3]从另一个角度来处理规模巨大的 ISS,其指出可以借助数学方法和原理对 DUT 进行高层次抽象,抽象后的模型虽然忽略了原系统的一些具体细节,但抓住了其最重要的特性,而且此时其 ISS 规模相对逻辑信号级的要小很多,因此可以先针对高层次抽象模型的 ISS 进行搜索产生高层次激励,然后增加一些细节转换成最后的激励. 但是该文章的目标 DUT 是时序控制系统,并没有讨论另一类重要逻辑——总线系统.

激励生成时另一个重要的问题就是覆盖率,而覆盖率本质上反映了已加载激励在 ISS 的分布情况,其能够很好地反映当前功能验证的完备程度,所以一直以来也是研究的热点. 目前覆盖率模型主要分为两大类:结构覆盖率和功能覆盖率,其中结构覆盖率关注 HDL 层,是一种比较低层次的覆盖率. 其优点是统一模型,能够得到目前 EDA 工具非常好的支持,但是其缺点也非常明显,100%的结构覆盖率并不能保证功能的完全正确性;功能覆盖率关注设计需求中的功能定义,是高层次的覆盖率. 其优点是能够保证功能正确性,但缺点就是由于不同的 DUT 彼此之间差别太大,所以缺乏统一的功能覆盖率模型.

文献[4]提出了一种基于可观测性的覆盖率模型,从层次上说其高于传统的结构覆盖率,但是它仍然是基于 HDL 代码的,所以并不是完全的功能覆盖率. 文献[5]提出用时序逻辑(TL,Temporal Logic)来描述系统的功能需求,并将不同的时序事件进行组合生成复杂的覆盖事件. 相对结构覆盖率,TL 虽然能够比较高层次地描述系统的功能需求,但由于还是使用了具体的逻辑信号,不同系统之间的通用性仍不是很高. 文献[6]提出了一种基于 SystemC 变异测试的覆盖率模型. 变异测试是近些年比较热门的一种验证测试手段,其源于软件测试领域,基本思想是模拟实际代码实现者一些经常性的错误,对 DUT 的代码实现进行一些小的改动变异,得到 DUT 的一些变异体,然后将同样的激励同时测试 DUT 和其变异体,观察变异体的暴露情况. 由此可见变异测试本质上还是需要依赖 HDL 代码的,而且其效果的好坏决定于变异算子的定义,所以也不是一种高层次的功能覆盖率.

总的来看在已有的相关工作中,存在以下不足:1)虽然引入了等价类划分和高层次抽象等方法来应对 ISS 规模爆炸的问题,但是只讨论了时序控制系统,缺少对另一类重要逻辑——总线系统的讨论;2)虽然有提出一些新的覆盖率模型,但大部分还是处在逻辑信号级或基于具体代码实现的,其抽象层次不够高,不能很好地帮助揭示高层次功能 Bug.

本文针对 SoC 中最常见的总线系统提出了一套通用的 ISS 高层次抽象和等价类划分方法,并提出了相应的高层次功能覆盖率模型,最后给出基于树搜索的结构化激励生成算法.

1　具体方案

1.1　有向二分图模型

由于任何总线系统所接的设备都具备主从属性,因此可将任何总线系统 X 对外总线接口集合划分成两个子集:主接口集 host_if 和从接口集slave_if.

1)host_if(X):其元素对应总线接口的从设备是 X;

2)slave_if(X):其元素对应总线接口的主设备是 X.

然后将 X 进一步抽象成有向二分图模型 $G=(V_h,V_s,E)$,其中图的三元素每个含义如下.

1)主点集 V_h:host_if(X)中每个元素抽象成顶点后构成的集合;

2)从点集 V_s:slave_if(X)中每个元素抽象成顶点后构成的集合;

3)有向边集 E:V_h 中顶点 vh_i 到 V_s 中顶点 vs_i 之间存在有向边 $e_{ij}=\langle vh_i,vs_j\rangle$ 的充要条件是 vh_i 对应主接口的激励可以激发 vs_i 对应的从接口.

例如某总线系统的结构如图 1 所示. 总线系统 C 通过总线(if1、if2、if3、if4 和 if5)与其它 5 个模块相连接,其中 C 是总线 if1 和 if2 的从设备,接受来自 A 和 B 的激励;C 是总线 if3、if4 和 if5 的主设备,驱动模块 D、E 和 F. 此外由于系统功能定义,模块 A 通过 if1 发的激励只能驱动模块 D 和 E,模块 B 通过 if2 发的激励能驱动模块 D、E 和 F.

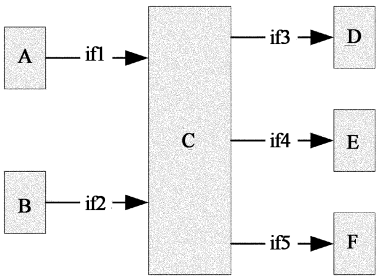

图 1 示例总线系统结构图

Fig. 1 Structure of example bus system

对于目标总线系统 C,按照上述对模块总线分类的标准,可以得到如下两个集合:

host_if(C)={if1,if2}

slave_if(C)={if3,if4,if5}

其有向二分图模型可以抽象如图 2 所示,图的三元素具体分别为:主点集,$V_h=\{1,2\}$,与 host_if(C)一一对应;从点集,$V_s=\{3,4,5\}$,与 slave_if(C)一一对应;有向边集,$E=\{\langle 1,3\rangle,\langle 1,4\rangle,\langle 2,3\rangle,\langle 2,4\rangle,\langle 2,5\rangle\}$.

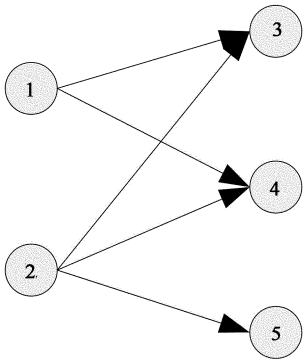

图 2 总线 C 的有向二分图模型

Fig. 2 BGM of bus C

其中由于模块 C 功能属性中约束了模块 A 通过 if1 发的激励不能通过 if5 驱动模块 F,所以有向边〈1,5〉不属于 E.

1.2 抽象激励模型

基于总线系统本身的性质和其有向二分图模型,其激励可以抽象成如下形式:

1)接口激励 IfSti. 任何时间单个主接口发起的一次激励,可抽象为有向二分图 G 中的一条有向边 $\mathrm{IfSti}=e_{ij}=\langle vh_i,vs_j\rangle$,即第 i 个主接口发起请求,目标驱动第 j 个从设备.

2)系统激励 SysSti. 任意时刻加载到整个总线系统的激励,是所有活动主接口的 IfSti 集合,即对应有向二分图 G 中有向边的集合 $\mathrm{SysSti}=\{e_{ij}\}$. 由于任意时刻单一主接口只能驱动单一从接口,所以对于 SysSti 集合,有一约束,即任意两条有向边的起点都不相同.

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 3 个重要的参数.

1)NHI(Number of Host Interfaces). 系统的主接口数目,即集合 host_if(X)的元素个数;

2)NSI(Number of Slave Interfaces). 系统的从接口数目,即集合 slave_if(X)的元素个数;

3)NAH(Number of Active Host interfaces). 某时刻系统活跃的主接口数,即 SysSti 中所包含的 IfSti 数.

根据激励模型中 SysSti 的约束和鸽巢原理,NAH 的取值范围为:

$$1\leqslant \mathrm{NAH}\leqslant \mathrm{NHI} \tag{1}$$

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内的 SysSti 具有相同的 NAH. 根据式(1),ISS 可被划分成 NHI 个子集,即 $EP(\mathrm{ISS})=\{S_1,S_2,\cdots,S_{\mathrm{NHI}}\}$,其中子集 S_i 的下标 i 表示其内所有激励数目,取值范围为 $i(1\leqslant i\leqslant \mathrm{NHI})$;

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内发 i 个 IfSti 对应的主接口是确定的. 根据排列组合规律,S_i 可被划分成个 C_{NHI}^i 个子集(从 NHI 个主接口中选取 i 个),即 $EP(S_i)=\{S_{i,1},S_{i,2},\cdots\}$,其中子集 $S_{i,j}$ 有两个下标:第一个 i 表示是对 S_i 的划分;第二个 j 是特征编号,其取值范围为 $1\leqslant j\leqslant C_{\mathrm{NHI}}^i$;

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内 IfSti 的主从接口对应关系是确定的. 由于不同主接口的 IfSti 从接口可以相同,记 $S_{i,j}$ 内激励所占用的 i 个主接口对应 G 中的点分别为 $vh_1^j,vh_2^j,\cdots,vh_i^j$,则 $S_{i,j}$ 可被划分成 $\deg^+(vh_1^j)*$

$\deg^+(vh_2^j)$,…,$\deg^+(vh_i^j)$个子集($\deg^+(v)$表示点 v 的出度),即 $EP(S_{i,j})=\{S_{i,j,1},S_{i,j,2},\cdots\}$,其中子集 $S_{i,j,k}$有 3 个下标:前两个 i,j 表示是对 $S_{i,j}$的划分;第三个 k 是特征编号,其取值范围为 $1\leqslant k\leqslant \deg^+(vh_1^j)*\deg^+(vh_2^j),\cdots,\deg^+(vh_i^j)$.

经过上述算法的划分,最后整个细化的 ISS 是个 3 层的树状结构,该树的每个叶子节点就代表了一个高层次的 SysSti,因为其描述了有哪些主接口是活动的,并给出了每个活动主接口所驱动的从接口.

下面以图 1 中的模块 C 为示例,具体说明上述算法的流程.

第 1 层:由于模块 C 的 NHI 为 2,则可以将 ISS 划分成两个等价类,即 $EP(\mathrm{ISS})=\{S_1,S_2\}$,其中 S_1 中所有 SysSti 的 NAH 都为 1,即只有单个主接口发请求;S_2 中所有 SysSti 的 NAH 都为 2,即 2 个主接口同时发请求.

第 2 层:由于模块 C 的主接口有 2 个,则可以将 S_1 划分成两个等价类,即 $EP(S_1)=\{S_{1,1},S_{1,2}\}$,其中 $S_{1,1}$中的 IfSti 都是由主接口 if1 发起的,$S_{1,2}$中的 IfSti 都是由主接口 if2 发起的;可以将 S_2 划分成一个等价类,即 $EP(S_2)=\{S_{2,1}\}$,$S_{2,1}$中所有 SysSti 都包含两个 IfSti,其中第一个是由主接口 if1 发起,第 2 个由主接口 if2 发起.

第 3 层:由于模块 C 的主接口 if1 可以驱动从接口 if3 和 if4,主接口 if2 可以驱动从接口 if3、if4 和 if5,则可以将 $S_{1,1}$ 划分成 2 个等价类,即 $EP(S_{1,1})=\{S_{1,1,1},S_{1,1,2}\}$,其中 $S_{1,1,1}$ 中的 SysSti 为 $\{\langle 1,3\rangle\}$,$S_{1,1,2}$中的 SysSti 为$\{\langle 1,4\rangle\}$;将 $S_{1,2}$划分成 3 个等价类,即 $EP(S_{1,2})=\{S_{1,2,1},S_{1,2,2},S_{1,2,3}\}$,其中 $S_{1,2,1}$中的 SysSti 为$\{\langle 2,3\rangle\}$,$S_{1,2,2}$中的 SysSti 为$\{\langle 2,4\rangle\}$, $S_{1,2,3}$中的 SysSti 为$\{\langle 2,5\rangle\}$;将 $S_{2,1}$划分成 6 个等价类,即 $EP(S_{2,1})=\{S_{2,1,1},S_{2,1,2},S_{2,1,3},S_{2,1,4},S_{2,1,5},S_{2,1,6}\}$,其中 $S_{2,1,1}$中的 SysSti 为$\{\langle 1,3\rangle,\langle 2,3\rangle\}$,$S_{2,1,2}$中的 SysSti 为$\{\langle 1,3\rangle,\langle 2,4\rangle\}$,$S_{2,1,3}$中的 SysSti 为$\{\langle 1,3\rangle,\langle 2,5\rangle\}$,$S_{2,1,4}$中的 SysSti 为$\{\langle 1,4\rangle,\langle 2,3\rangle\}$,$S_{2,1,5}$中的 SysSti 为$\{\langle 1,4\rangle,\langle 2,4\rangle\}$,$S_{2,1,6}$中的 SysSti 为$\{\langle 1,4\rangle,\langle 2,5\rangle\}$,最后得到的 ISS 树如图 3 所示.

1.4　高层次功能覆盖率模型

上述等价类划分算法将 ISS 划分成了 3 个层次,理论上每一层都可以定义一种相应的高层次功能覆盖率,分别为:活跃主接口数覆盖率(Number of Active Host interface Coverage,NAHC),活跃主接口覆盖率(Active Host interface Coverage, AHC)和主从对覆盖率(Host Slave Pair Coverage, HSPC),每一种的具体定义如下.

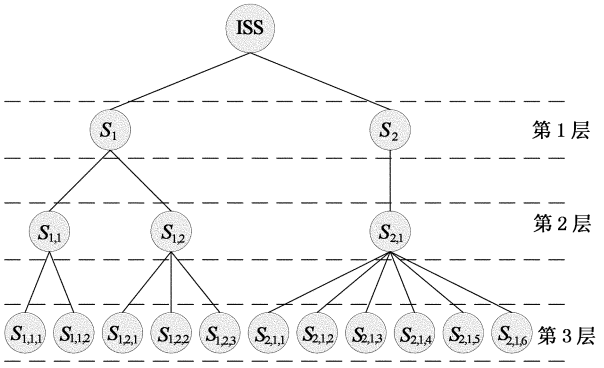

图 3　总线 C 的 ISS 等价类划分树

Fig. 3　Equivalence partition tree of bus C's ISS

1.4.1　NAHC

目标覆盖点集 $\mathrm{Target_{NAHC}}$:该集合为 ISS 树中第 1 层的节点构成的集合,即$\mathrm{Target_{NAHC}}=\{S_1,S_2,\cdots,S_{\mathrm{NHI}}\}$.

已覆盖点集 $\mathrm{Covered_{NAHC}}$:如果一个 SysSti 加载到了 DUT 中,其对应的 ISS 树叶子节点与根节点间路径所经过的第 1 层节点为 S_i,则称 S_i 被覆盖,并入 $\mathrm{Covered_{NAHC}}$,即$\mathrm{Covered_{NAHC}}=\mathrm{Covered_{NAHC}}\cup\{S_i\}$.

覆盖率值:$\mathrm{Coverage_{NAHC}}$:$\mathrm{Coverage_{NAHC}}=\dfrac{|\mathrm{Covered_{NAHC}}|}{|\mathrm{Target_{NAHC}}|}$.

1.4.2　AHC

目标覆盖点集 $\mathrm{Target_{AHC}}$:该集合为 ISS 树中第 2 层的节点构成的集合,即$\mathrm{Target_{AHC}}=\bigcup_i EP(S_i)$.

已覆盖点集 $\mathrm{Covered_{AHC}}$:如果一个 SysSti 加载到了 DUT 中,其对应的 ISS 树叶子节点与根节点间路径所经过的第 2 层节点为 $S_{i,j}$,则称 $S_{i,j}$被覆盖,并入已覆盖集$\mathrm{Covered_{AHC}}=\mathrm{Covered_{AHC}}\cup\{S_{i,j}\}$.

覆盖率值:$\mathrm{Coverage_{AHC}}$:$\mathrm{Coverage_{AHC}}=\dfrac{|\mathrm{Covered_{AHC}}|}{|\mathrm{Target_{AHC}}|}$.

1.4.3　HSPC

目标覆盖点集 $\mathrm{Target_{HSPC}}$:该覆盖率模型的目标覆盖集为 ISS 树中叶子节点构成的集合,即$\mathrm{Target_{HSPC}}=\bigcup_{i,j} EP(S_{i,j})$.

已覆盖点集 $\mathrm{Covered_{HSPC}}$:如果一个 SysSti 加载到了 DUT 中,其对应的 ISS 树叶子节点为 $S_{i,j,k}$,则称 $S_{i,j}$ 被覆盖,并入已覆盖集 $\mathrm{Covered_{HSPC}}=$

$\text{Covered}_{\text{HSPC}} \cup \{S_{i,j,k}\}$.

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

1.4.4 对比

从以上 3 种覆盖率模型的定义可以发现，NAHC 的目标覆盖点数$|\text{Target}_{\text{NAHC}}|=\text{NHI}$，AHC 的目标覆盖点数$|\text{Target}_{\text{AHC}}|=2^{\text{NHI}}-1$，HSPC 的目标覆盖点数是 $2^{\text{NHI}}-1$ 个正整数的和，一般$|\text{Target}_{\text{HSPC}}| \gg 2^{\text{NHI}}-1$. 而在一般的总线系统中 NHI 都不会超过 3 位数，即 NAHC 的收敛速度会太快；AHC 已经相对 NHI 具备了指数规模，而 HSPC 的规模比 AHC 还大，所以 HSPC 的收敛速度会太慢，故从性能和收敛速度两方面综合考虑，一般优先采用 AHC.

1.5 结构化激励生成

本文的结构化总线系统验证激励生成算法分成如下两步：

步骤 1. 对 ISS 树叶子节点进行搜索遍历以确定 SysSti 中各活动主接口及其对应的主从驱动关系信息；

步骤 2. 对于 SysSti 中的每个 IfSti，随机产生从接口内偏移地址、数据流方向及数据等细节信息.

上述激励生成算法产生的激励不仅保留了逻辑信号级随机性，而且在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 $S_{i,j,k}$可以由其下标三元组$\langle i,j,k\rangle$来唯一索引确定，故叶子节点的搜索遍历就转换成了对$\langle i,j,k\rangle$三元组的搜索遍历.

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.

1.5.1 DFS

其基本思想是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 4 所示.

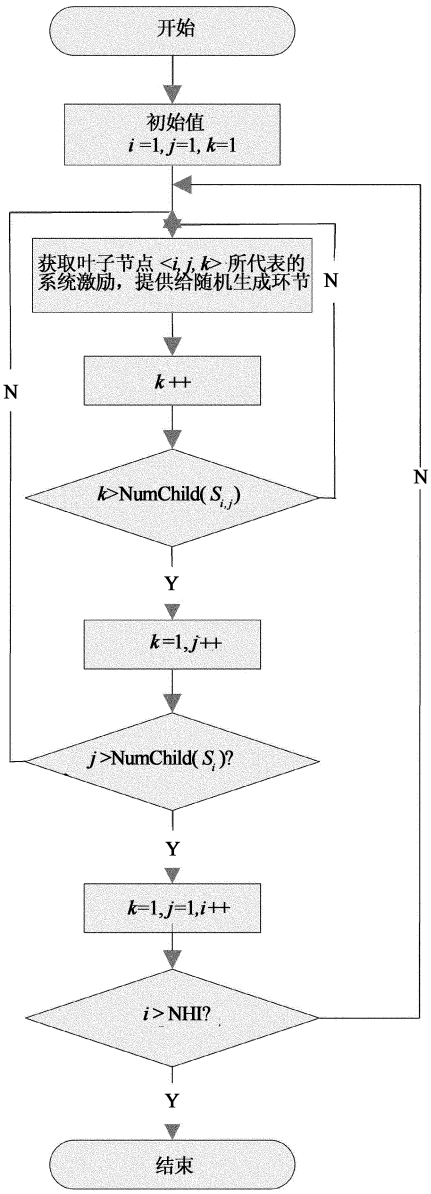

图 4 ISS 树 DFS 实现

Fig. 4 DFS Implementation of ISS tree

其中 NumChild(S)函数返回节点 S 的儿子节点数目.

按照上述深度优先搜索算法，图 3 中的叶子节点搜索遍历顺序为 $S_{1,1,1}\rightarrow S_{1,1,2}\rightarrow S_{1,2,1}\rightarrow S_{1,2,2}\rightarrow S_{1,2,3}\rightarrow S_{2,1,1}\rightarrow S_{2,1,2}\rightarrow S_{2,1,3}\rightarrow S_{2,1,4}\rightarrow S_{2,1,5}\rightarrow S_{2,1,6}$.

1.5.2 BFS

其基本思想是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 5 所示.

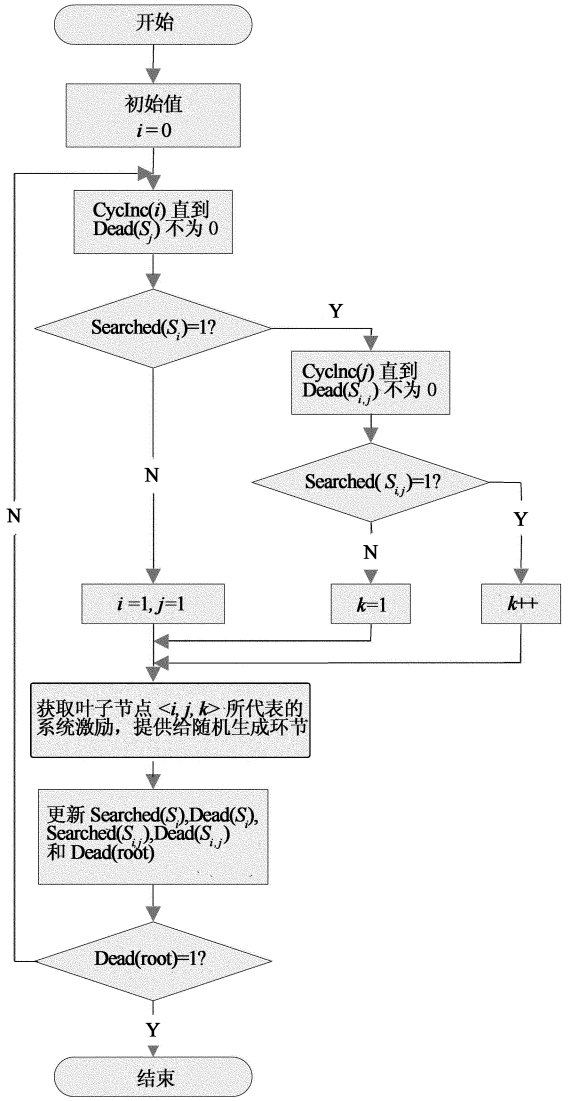

图 5　ISS 树 BFS 实现

Fig. 5　BFS implementation of ISS tree

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.

按照上述宽度优先搜索算法,图 3 中的叶子节点搜索遍历顺序为 $S_{1,1,1}\rightarrow S_{2,1,1}\rightarrow S_{1,2,1}\rightarrow S_{2,1,2}\rightarrow S_{1,1,2}\rightarrow S_{2,1,3}\rightarrow S_{1,2,2}\rightarrow S_{2,1,4}\rightarrow S_{1,2,3}\rightarrow S_{2,1,5}\rightarrow S_{2,1,6}$.

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 系统中的总线接口单元(BIU,Bus Interface Unit)作为功能验证实例.

2.1　IME-Diamond BIU

IME-Diamond 是中科院微电子研究所自主研发的高性能多核嵌入式 SoC[7-8],其中 BIU 是其内核与主存和外设之间的系统总线接口模块,它是典型的总线型系统,有 13 个主接口,3 个从接口,其有向二分图的连接矩阵如表 1 所示.

表 1　BIU 的有向二分图的连接矩阵

Tab. 1　Adjacent matrix of BIUs BGM

Host	Slave		
	MC	SPM	PBU
CPUA	1	1	1
CPUB	1	1	1
CPUC	1	1	1
CPUD	1	1	1
ICacheA	1	1	0
ICacheB	1	1	0
ICacheC	1	1	0
ICacheD	1	1	0
DCacheA	1	1	0
DCacheB	1	1	0
DCacheC	1	1	0
DCacheD	1	1	0
DMA	1	0	0

其中第 1 列为 13 个主接口 Host,第 1 行为 3 个从接口 Slave.矩阵元素$\langle h_i, s_j\rangle$为 1 时表示主接口 h_i 可以驱动从接口 s_j,为 0 时表示不能驱动.

2.2　ISS 树

按照上述层次化划分算法,最后得到的 ISS 树节点数统计如表 2 所示.

表 2　BIU 的 ISS 树各层节点统计

Tab. 2　Node numbers of BIUs ISS tree

第 1 层节点编号	第 2 层节点数	第 3 层节点数
1	13	29
2	78	386
3	286	3 122
4	715	17 117
5	1 287	67 169
6	1 716	194 048
7	1 716	417 776
8	1 287	670 112
9	715	790 624
10	286	666 624
11	78	380 160
12	13	131 328
13	1	20 736

表中第 2 列和第 3 列的节点数都是指在以 S_i 为根节点的子树中.

3 结果分析

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 等基于代码的结构覆盖率和本文所提出的 AHC 高层次功能覆盖率，各覆盖率值与揭示的功能 Bug 数间关系比较如图 6 所示.

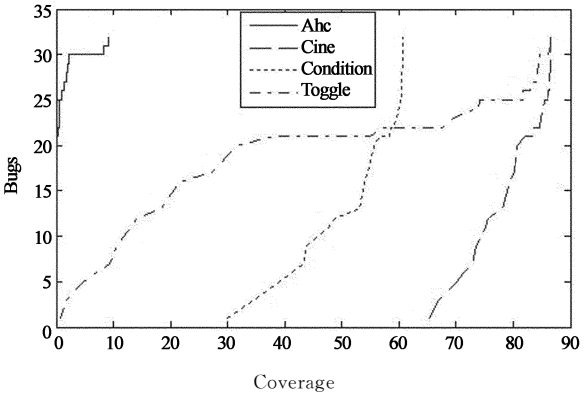

图 6 不同覆盖率模型与揭示 bug 数关系

Fig. 6 Uncovered bug numbers vs different coverage models

由图 6 可见，相对所有的结构覆盖率，AHC 都能够在很小的覆盖率下发现同样数目的功能 Bug，这也就意味着 AHC 揭示功能 Bug 的能力比传统的结构覆盖率要更强大.

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，验证策略 V1 中采用纯随机算法；验证策略 V2 中采用了本文所提的基于 DFS 或 BFS 的结构化算法；功能覆盖率模型采用 AHC，不同验证策略下覆盖率值与所加载的 SysSti 数关系如图 7 所示.

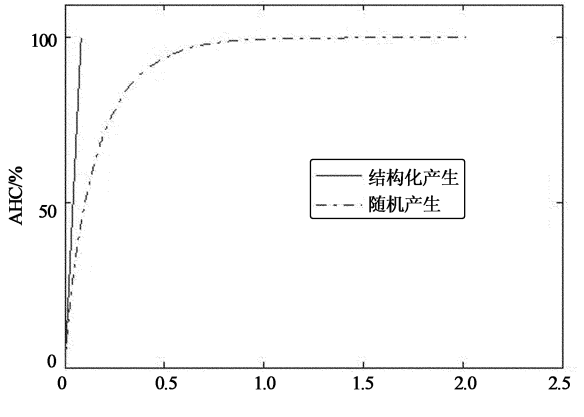

图 7 不同验证策略下 AHC 覆盖率与激励数关系

Fig. 7 AHC value vs SysSti number

为达到 100%的覆盖率，所加载的 SysSti 数及实现各策略所需实现成本统计如表 3 所示.

表 3 不同验证策略 100%覆盖率所需激励及实现成本

Tab. 3 Implementation cost

覆盖率模型	随机产生	结构化产生
AHC	201 930	8 191
时间成本/天	0.5	1
代码成本/行	30	100

从上述数据可以看出，虽然结构化产生方法相对随机产生方法的实现成本有 1～3 倍的增加，但是实际运行时，结构化方法的覆盖率收敛速度是线性的，而随机化方法是非线性的，且越接近 100%时越慢，其中为了达到 100%的 AHC 覆盖率，结构化方法所需的激励数仅是随机化方法的 1/24，即覆盖率收敛速度提高了 96%.

4 结 语

为了应对逻辑信号级输入激励空间爆炸的问题，本文针对总线系统提出了一套高层次结构化激励生成算法和相应的功能覆盖率模型.

1)将总线系统抽象成通用的有向二分图模型，并构建了其输入激励的高层次数学模型；

2)利用等价类划分思想将高层次输入激励空间划分成 3 层的树状结构；

3)基于输入激励空间树提出了对应的高层次功能覆盖率模型；

4)基于输入激励空间树的搜索给出了对应的高层次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结构覆盖率，高层次功能覆盖率模型的揭示 Bug 能力更强，而且相对于传统的纯随机生成，结构化的激励生成能够更快地加快覆盖率收敛.

参考文献

[1] ARCURI A, IQBAL M Z, BRIAND L. Random testing: theoretical results and practical implications[J]. IEEE Transactions on Software Engineering, 2011, 38(2):258－277.

[2] WEYUKER E J, JENG B. Analyzing partition testing strategies[J]. IEEE Transactions on Software Engineering, 1991, 17(7):703－711.

[3] SUMNERS R, BHADRA J, ABRAHAM J. Automatic validation test generation using extracted control models[C]// Thirteenth International Conference on VLSI Design. IEEE, 2000:312－317.

[4] DEVADAS S, GHOSH A, KEUTZER K. An observability-based code coverage metric for functional simulation[C]// IEEE/ACM International Conference on Computer-Aided Design. IEEE Computer Society, 1997:418－425.

(下转第 160 页)