

基于覆盖率驱动的 SoC 验证技术研究

朱车壮^{1,2}, 陈 岚¹, 冯 燕¹

(1 中国科学院 微电子研究所, 北京 100029; 2 北京大学, 北京 100871)

摘 要: 覆盖率数据是验证工程师判定 SoC 验证完备程度的定性度量, 为 SoC 验证完全性提供了保障, 指明了方向. 文中以 SoC 总线仲裁器验证为例, 对其结构覆盖率、功能覆盖率、断言覆盖率等多种覆盖率进行了全面的分析, 然后根据覆盖率分析结果反馈到 RTL 设计代码和测试激励进行修正, 直到验证的完整性满足设计的要求.

关键词: SoC 验证方法; 覆盖率驱动; 结构覆盖率; 功能覆盖率; 断言覆盖率

中图分类号: TN911

文献标识码: A

文章编号: 1000-7180(2011)11-0048-05

Research of SoC Verification Technology Based on Coverage—Driven

ZHU Che-zhuang^{1,2}, CHEN Lan¹, FENG Yan¹

(1 Institute of Microelectronics, Chinese Academy of Science, Beijing 100029, China;

2 Peking University, Beijing 100871, China)

Abstract: Coverage data—base is the qualitative measurement for the verification engineer to judge the complete degree of SoC verification, and it provides the safeguard for the SoC validation completeness. This paper takes the verification of SoC bus arbiter as an example, analyzes many kinds of coverage in detail, for example, structure coverage, functional coverage, assertion coverage and so on, then modifies the RTL design code and testcases based on the analysis results of these coverage, until the integrity of the verification meet the design requirements.

Key words: SoC verification method; coverage—driven; structural coverage; functional coverage; assertion coverage

1 引言

随着 SoC 功能复杂度的提高, 对验证工作提出了更高的要求, 所以必须在投片前对芯片设计进行全面、可信的验证. 在大部分 SoC 开发项目中, 验证时间占整个芯片开发周期的 70% 左右, 于是验证的正确完备及其耗时的缩短成为了 SoC 设计的关键. SoC 验证的完备性可由覆盖率指标去定性度量, 而覆盖率反馈使得问题收敛速度加快, 有效缩短了验证时间.

2 SoC 系统总线验证

良好的 SoC 总线架构能够保证各个 IP 模块之间进行高效的数据交互, 它对整个系统的性能有着

决定性的影响. 图 1 是 SoC 总线的示意图. 主要包括 3 个 master, 2 个 slave, 1 个 arbiter, 1 个 decoder. 需要使用总线的主设备向仲裁器提出使用总线请求, 仲裁器经过判定后授权给指定的主设备. 任一时间周期内只能有一个主设备可以接入总线, 对其指定的从设备进行读写操作. 译码器将主设备的信息解码提高给从设备, 并建立一个指定的主设备和从设备之间的连接, 及成功管理这个事务.

总线是连接各个功能模块的桥梁, 只有对总线进行充分的验证, 才能保证整个芯片可以正常工作. SoC 系统总线的验证主要是当有主设备请求总线进行事务传输时, 仲裁器进行正确判定并授予总线, 译码器负责将主设备和从设备正确连接起来. 在这里只列举仲裁器模块的验证情况, 其他 IP 模块的验证

收稿日期: 2011-01-05; 修回日期: 2011-02-07

基金项目: 03 重大专项(2009ZX03005-002); 国家“八六三”计划项目(2009AA012002); 中科院知识创新项目

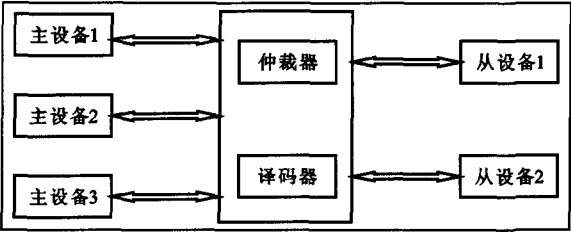


图1 SoC 系统总线架构示意图

过程类似。

2.1 SoC 总线验证方法

SoC 验证方法主要有动态模拟仿真和静态形式验证。在 RTL 级常用到动态模拟仿真,测试激励使用随机约束生成。SVA 语言可以很好地帮助验证,通过对代码执行情况、功能点覆盖情况、接口协议信号断言情况的收集,使得对这个 SoC 系统总线的完备验证指明了方向,当这些覆盖率达到要求时,断定该系统总线已经验证完毕,没有出现功能性和时序性错误,为后续集成其他的模块提供了保障。

2.2 测试平台搭建

为了测试 SoC 模块的正确性,搭建了测试平台,如图 2 所示。测试平台的目的是产生测试激励,并将测试激励施加到 DUT 上,然后自检 DUT 输出信号正确性,收集覆盖率数据。通常测试平台还会产生日志文件帮助调试。

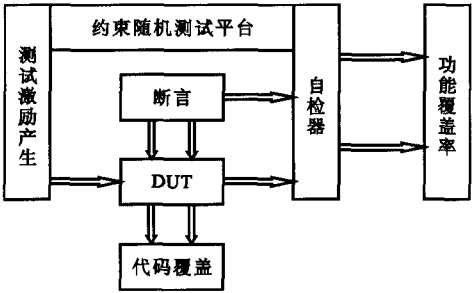


图2 测试平台示意图

2.3 被测设计分析

总线仲裁器是提供仲裁逻辑来决定哪个主设备与从设备之间进行事务交互。所有主设备都能请求授予总线,仲裁器按照轮转方式决定哪一个主设备获得总线。一旦一个主设备得到许可,它将利用总线来执行一些事务。在事务结束时,主设备会通知仲裁器,然后释放总线。一旦总线被释放,如果主设备有挂起的事务,它们可以再次请求总线。关键点是确保仲裁器不会让某个主控设备一直得不到许可。

其中主要验证核心是仲裁器状态机的正确转移。图 3 为仲裁器状态转换图。

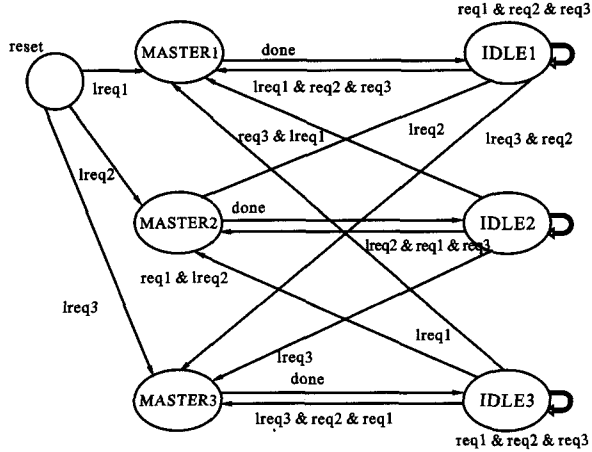


图3 bus_arbiter 模块状态机转换图

3 覆盖率驱动 SoC 验证

验证的目标是彻底地验证被测设计(DUT),确保其中没有功能缺陷。在这个过程中,应该有一个方法来衡量验证的完整性。

覆盖率一般表示一个设计的验证进行到什么程度,是一个决定功能验证是否完成的重要量化标准之一。它包括功能覆盖率、结构覆盖率和断言覆盖率。

覆盖率分析方法在验证工作中被统称为以覆盖率为驱动的验证技术。根据统计^[1],如图 4 所示。覆盖率驱动验证与传统验证相比较,问题收敛速度加快,验证时间缩短了三个月而且验证过的功能规格点都是可信、可用的功能点。在流片之前,芯片的风险得到了有效控制。

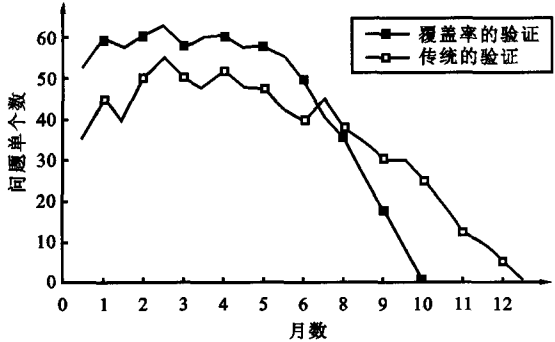


图4 覆盖率驱动验证与传统验证的问题收敛对比

3.1 结构覆盖率分析

结构覆盖率又称代码覆盖率,主要包括行覆盖率、分支覆盖率、条件覆盖、翻转覆盖率和 FSM 覆盖率等。到目前为止,还很难给出每种覆盖率与其错误发现能力之间的定性关系,至于在实际应用中该采用哪种覆盖率评估准则只能依靠经验来判断。结构覆盖率分析主要是指芯片设计代码中的错误分

析. 在测试仿真结束时, 如果代码覆盖率没有达到 100%, 说明有些代码没有被执行到. RTL 代码中可能存在 bug, 这就需要分析 RTL 设计代码和测试激励是谁出现了问题. 高效率结构覆盖率评估准则是指出设计中的 bug, 使得 RTL 代码更加高效地执行.

3.1.1 行覆盖率

语句、行或块覆盖率能衡量验证方法运行了整个代码中的多少行. EDA 工具图形化的用户界面可以让用户浏览源代码并快速识别未运行的语句.

使用 cmView 代码覆盖率收集工具对 bus_arbiter 模块进行了代码覆盖率收集, 如图 5 所示. bus_arbiter 模块的代码覆盖率达到 95.24%, 查看没有覆盖到的语句时, 发现测试激励没有覆盖到该情况, 也即在初始状态 IDLE 下, 当 3 个主设备同时申请总线时, 仲裁器都将总线使用权授予了主设备 1, 故其余两个主设备申请没有得到同意, 故它们的语句执行并没有覆盖到. 因为测试激励是约束随机的, 3 个主设备在初始时刻总线申请也是随机产生的, 故此情况属正常.

Instance	Module	Blk C(%)	Line C(%)	Stmnt C(%)
tb	tb	100.00	100.00	100.00
u1	master_bfm	100.00	100.00	100.00
u2	master_bfm	100.00	100.00	100.00
u3	master_bfm	100.00	100.00	100.00
u4	bus_arbiter	95.12	95.24	95.24

图 5 bus_arbiter 模块行覆盖率情况

3.1.2 分支覆盖率

运行一组语句的路径通常不止一种, 分支覆盖用来检查一组语句运行的所有可能分支路径.

除了可以使用 cmView 覆盖率图形工具查看覆盖率之外, 还可以通过 urg 命令将覆盖率数据转换成网页形式, 也可以转换成文本形式.

	Line No.	Total	Covered	Percent
Branches	—	31	29	93.55
TERNARY	129	2	2	100.00
TERNARY	130	2	2	100.00
TERNARY	131	2	2	100.00
IF	52	2	2	100.00
CASE	63	23	21	91.30

图 6 bus_arbiter 模块的路径覆盖率情况

图 6 是 bus_arbiter 的分支覆盖率情况. 图中有两个分支路径没有覆盖到, 这跟语句覆盖同出一辙, 也就是主设备 2 和主设备 3 的请求情况没有得到许可, 导致代码执行不到.

3.1.3 条件覆盖率

条件覆盖也称为表达式覆盖. 由于导致控制流变化的原因可能不止一个, 例如两个相互独立的条件都可以触发一条语句时, 语句覆盖率达到 100%, 而表达式覆盖率只有 50%. 因此, 当表达式覆盖率未达到 100% 时, 有必要弄清楚表达式的控制条件为什么没有运行, 是不是这个条件不可能发生, 还是被忽略了.

图 7 是 bus_arbiter 的条件覆盖率情况. 图中有 9 个条件分支, 其中已经覆盖到 7 个条件, 有 2 个条件并没有触发到. 查看是哪两个条件没有覆盖到, 发现 frame 信号和 irdy 信号 0-1, 1-0 情况没有触发. 因为在设计中就规定这两个信号只能同时为 0 或 1, 故测试激励中就没有出现以上的测试激励了.

	Total	Covered	Percent
Conditions	9	7	77.78
Logical	5	5	100.00
Non-Logical	4	2	50.00
Event	0	0	

LINE 46

EXPRESSION (frame && irdy)

--1-- --2--

-1-	-2-	Status
0	1	Not Covered
1	0	Not Covered
1	1	Covered

图 7 bus_arbiter 模块的条件覆盖率情况

3.1.4 有限状态机覆盖率

语句覆盖能检测到未访问的状态. 通常有限状态机(FSM)的每个状态都明确地与 case 语句的一个分支相对应, 因此任何未被访问的状态都可以通过未覆盖的语句清楚地识别出来. 验证过程中一个未被覆盖的 case 语句对应了一个未访问的状态.

Summary for FSM::state

	Total	Covered	Percent
Transitions	22	17	77.27
Sequences	4	4	100.00

State, Transition and Sequence Details for FSM::state

States	Covered
0	Covered
IDLE	Covered
MASTER1	Covered
LDLE1	Covered
MASTER2	Covered
IDLE2	Covered
MASTER3	Covered
IDLE3	Covered

图 8 bus_arbiter 模块的 fsm 覆盖率情况

图 8 是 bus_arbiter 的状态机覆盖率情况. 从图中可以知道 8 个状态都已经覆盖到, 只是传输情况并没有全部覆盖, 这是因为在设计中就规定好了状态的转移路径, 只要规定的转移路径传输满足了, 那么该状态机设计就是正确的了.

Instance	Module	Total C(%)	RegBits C(%)	NetBits C(%)	LogicBits C(%)	BitBits C(%)
u4	bus_arbiter	100.00	—	—	100.00	100.00

图 9 bus_arbiter 模块的 toggle 覆盖率情况

3.2 功能覆盖率分析

代码覆盖衡量了多少代码被执行过, 而功能覆盖则衡量设计的原始要求实现了多少.

功能覆盖是对代码覆盖的补充, 代码覆盖关心的是如何记录代码的执行, 而功能覆盖关心的是所执行的功能的意图或目的.

功能覆盖可以检测出忽略的错误. 代码覆盖工

3.1.5 翻转覆盖率

翻转覆盖对于 RTL 级代码的分析没有实际意义, 其比较常用于门级分析各逻辑门的输入输出信号的翻转情况, 文中不做详细分析, 从图 9 看到各个信号的翻转情况都满足了要求.

具要报告一段未被覆盖的代码的首要条件是该代码必须存在. 而功能覆盖检查则不依赖于实际的代码, 它通过检测代码是否处理了期望值来报告覆盖率.

针对 Bus_arbiter 模块的每个必须执行的功能点设置了功能覆盖检查. 测试激励采用约束随机产生, 从图 10 中可以确定该模块的所有功能都正确设计实现了.

Cover Properties	Category	Severity	Attempts	Matches	Incomplete
tb, u4, ul, c_forbid_trans1	0	0	10433	2605	0
tb, u4, ul, c_forbid_trans2a	0	0	10433	6970	0
tb, u4, ul, c_forbid_trans2b	0	0	10433	6948	0
tb, u4, ul, c_forbid_trans2c	0	0	10433	6934	0
tb, u4, ul, c_forbid_trans3a	0	0	10433	1741	0
tb, u4, ul, c_forbid_trans3b	0	0	10433	1737	0
tb, u4, ul, c_forbid_trans3c	0	0	10433	1732	0
tb, u4, ul, c_fsm2_encoding	0	0	10433	10432	0
tb, u4, ul, c_req1	0	0	10433	481	0
tb, u4, ul, c_req2	0	0	10433	474	0
tb, u4, ul, c_req3	0	0	10433	505	0
tb, u4, ul, c_req_gnt	0	0	10433	869	0
tb, u4, ul, c_req_gnt_1	0	0	10433	288	0
tb, u4, ul, c_req_gnt_2	0	0	10433	290	0
tb, u4, ul, c_req_gnt_3	0	0	10433	291	0

图 10 Bus_arbiter 模块功能覆盖率情况

对于验证工程师来说, 100% 的功能覆盖率并不意味着验证的结束, 它只是反映当前设计所定义的功能和相关的情况被正确设计验证, 它是否正确实现了所需的实际功能, 这要由当初所定义的功能覆盖模型和功能点所决定.

3.3 断言覆盖率分析

基于断言的验证 (ABV) 是硬件设计功能验证

的一种有效方法. 断言是设计的属性描述. 如果一个在模拟中被检查的属性不像期望的那样表现, 那么这个断言失败. 如果一个被禁止在设计中出现的属性在模拟过程中发生, 那么这个断言失败^[2]. 一系列的属性可以从设计的功能描述中推知, 并且被转换成断言. 这些断言能在功能的模拟中不断地被监视. 断言已被运用于 RTL 为主的侦错中, 在确认错误

及使错误区域化的方面有显著的效用^[3]. 断言增加了可观测性,减少了调试的时间,提高了验证效率.

传统覆盖分析要专门为覆盖分析而写大量的代码,而断言的覆盖分析可以直接使用在协议检查或者事件描述中用到的那些时序表达式,因此编码会更加灵活、简洁.

在 bus_arbiter 的设计中,将 bus_arbiter 的一系列属性转换成断言,这些断言在功能模拟中不断地被监视. 当该断言违反时,则自动显示错误和位置,十分方便调试. 针对每个属性定义的断言是否被正确监视,可以从日志中可以查看到. 图 11 为断言覆盖率情况,这些定义的断言都被正确监视了.

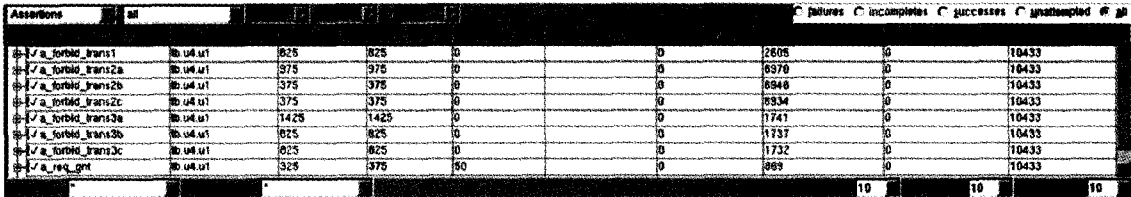


图 11 bus_arbiter 模块断言覆盖率情况

4 结束语

文中分析了 SoC 总线架构,以总线仲裁器验证为例,对其结构覆盖率、功能覆盖率、断言覆盖率等多种覆盖率进行了详细分析,然后根据这些覆盖率分析结果反馈到 RTL 设计代码和测试激励进行修正,直到验证的完整性满足设计的要求. 以此表明覆盖率数据成为验证工程师判定 SoC 验证完整程度的定性度量,为 SoC 验证完全性提供了保障. 随着在 SoC 系统中集成了越来越多的 VIP,可以将这些覆盖率评估总则编写成 IP,作为控制验证质量的方法总则,确保与相关协议的所有被测设计都能完全验证.

参考文献:

[1] 罗登富,赵建明. 基于覆盖率的集成电路验证[J]. 现代电子技术,2009(13):115-119.
[2] 徐盛,章玮,金钊. 基于断言的验证方法在总线协议验证中的应用[J]. 电子设计应用,2006(11):89-92.
[3] 褚晓滨,陆铁军,宗宇. 结合断言与覆盖率为导向的验证方法[J]. 微电子学与计算机,2008,25(11):139-142.

作者简介:

朱车壮 男,(1985-),硕士研究生. 研究方向为 SoC 系统设计与验证.
陈 岚 女,(1968-),博士生导师. 研究方向为超深亚微米芯片设计方法学及计算机系统机构.
冯 燕 女,(1980-),助理研究员. 研究方向为 SoC 系统设计与验证.

(上接第 47 页)

[6] Petri C A. Kommunikation mit automaten [D]. Bonn, Germany: Institut für Instrumentelle Mathematik, 1962.

作者简介:

冷 明 男,(1975-),博士,副教授. 研究方向为 Petri 网理论、组合分析与优化等.

孙凌宇 女,(1976-),硕士,副教授. 研究方向为 Petri 网理论、组合算法等.
郭恺强 男,(1978-),硕士,讲师. 研究方向为 Petri 网理论等.
朱 平 男,(1955-),硕士,教授. 研究方向为算法分析与设计等.