Journal of Sensor Technology and Application 传感器技术与应用, 2022, 10(2), 91-98
Published Online April 2022 in Hans. http://www.hanspub.org/journal/jsta
https://doi.org/10.12677/jsta.2022.102012

Hans汉斯

基于SV的NAND Flash控制器功能验证

吴录辉，颜炳佳

合肥工业大学，安徽 合肥

收稿日期：2022年1月25日；录用日期：2022年3月27日；发布日期：2022年4月6日

摘 要

随着存储工艺的发展，NAND Flash存储架构被提出后，凭借存储量大，读写速度快等优势，迅速成为存储介质的首选。目前，存储系统通过控制器完成与Flash闪存颗粒的交互，但随着存储数据越来越巨大，NAND Flash控制器的稳健性、完备性也越来越受重视。本文采用System Verilog语言搭建测试环境，并结合SVA断言技术提出新的验证策略，即收集代码覆盖率、功能覆盖率、断言覆盖率三个重要指标，对NAND Flash控制器模块进行全面有效的验证，确保同步时钟单通道模式下，控制器使用双FSM设计依然符合设计要求。

关键词

System Verilog，SVA，功能覆盖率，断言覆盖率，NAND Flash

Function Verification of NAND Flash Controller Based on SV

Luhui Wu, Bingjia Yan

Hefei University of Technology, Hefei Anhui

Received: Jan. 25th, 2022; accepted: Mar. 27th, 2022; published: Apr. 6th, 2022

Abstract

With the development of storage technology, NAND Flash storage architecture has quickly become the first choice of storage media with the advantages of large storage capacity and fast reading and writing speed. At present, the storage system completes the interaction with flash particles through the controller, but with the increasing storage data, the robustness and completeness of NAND Flash controller are paid more and more attention. In this paper, the system Verilog language is used to build the test environment and combined with the SVA assertion technology, a new verifi-

文章引用: 吴录辉, 颜炳佳. 基于 SV 的 NAND Flash 控制器功能验证[J]. 传感器技术与应用, 2022, 10(2): 91-98.
DOI: 10.12677/jsta.2022.102012

cation strategy is proposed, that is, the code coverage, function coverage and assertion coverage are collected to comprehensively and effectively verify the NAND Flash controller module, so as to ensure that the dual FSM design of the controller still meets the design requirements in the synchronous clock single-channel mode.

Keywords

System Verilog, SVA, Function Coverage, Assertion Coverage, NAND Flash

1. 引言

随着存储行业的迅猛发展，NAND (Not AND)闪存由于其独特的特性，如非易失性、低功耗和快速访问时间，被广泛应用于软硬件系统[1]。随着系统越来越复杂，数据量越来越大，如何验证系统的功能正确性是设计的重要因素；因此验证工作在整个设计生命周期内占据着相当大的比重。

功能验证主要分为基于形式验证的静态验证和基于仿真的动态验证。形式验证借助验证工具，以严密的逻辑方式穷尽所有可能去尝试着证明所有可能的输入条件以及设计中任意点的可控性和可视性。动态验证对设计文档提炼测试点，编写测试激励，选择随机测试方式，查找设计代码中的错误。此种方法凭借易于实现被广泛采用。System Verilog 中的 OOP (Object Oriented Programming)概念能够支撑起功能验证的需求，形成一个高效完善的验证策略[2]。

文献[3]基于 UVM (Universal Verification Methodology)的 Flash 控制器模块验证研究，采用以 System Verilog 实现的通用验证方法 UVM 搭建验证平台，对 Flash 控制器进行验证。但其只收集了代码覆盖率和功能覆盖率，并没有收集断言覆盖率去判断时序的正确性。所以该平台覆盖率数据需要进一步提升。文献[4]基于 UVM 的闪存控制器可扩展性通用验证架构的研究，对验证平台的扩展性做了细致的阐述，但是并未使用断言验证技术，缺乏对复杂设计的时序检查。

本文提出基于 System Verilog 的验证方法实现的验证策略，包含覆盖率组件、SVA (System Verilog Assertion)断言技术，不仅对 NAND Flash 控制器进行时序检查，还验证双 FSM 设计的可操作性。通过分析代码覆盖率、功能覆盖率和断言覆盖率数据，检测和提高控制器的功能实现情况。

2. NAND Flash 控制器概要描述

本文的验证对象是 Verilog 编写的 NAND Flash 控制器模块，其左侧为 Host 端接口，右侧为存储芯片。其中存储芯片选用的是 SAMSUNG 公司生产的 K9F1208 系列 NAND Flash 存储器芯片。NAND Flash 控制器系统的整体框图如图 1 所示。

由图可知，系统主要分为四个模块，分别为Host端控制信号产生模块、Buffer缓冲区模块、NAND Flash控制器模块和 K9F1208 闪存存储模块。其中 Host 端控制信号产生模块主要是产生闪存写操作、读 ID、擦除操作等命令，以及接收操作错误反馈信号，实现与 Host 的数据交互。Buffer 模块由异步 FIFO 实现，以便写操作和读操作时的数据缓存。NAND Flash控制器模块实现解释Host端命令、控制命令的Main FSM [5]、闪存端口控制信号(Flash_FSM 实现)和基础的 ECC 检错等功能。

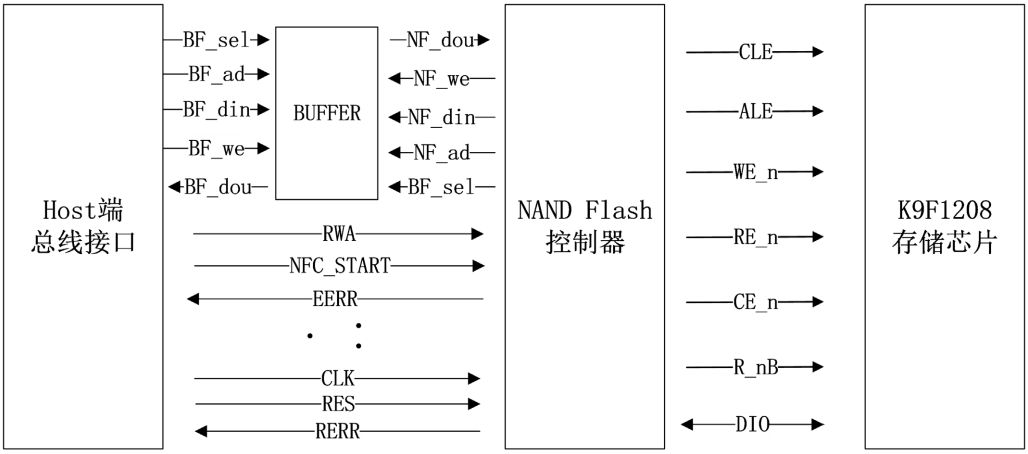

Figure 1. NAND Flash controller system block diagram
图 1. NAND Flash 控制器系统框图

主要研究同步时钟单通道模式下，双 FSM 实现对 NAND Flash 存储控制的可操作性，确保 Reset、Read ID、Erase block、Program page、Read page、Read stated 等基础命令被正确执行。此外研究新的验证策略的可行性。通过搭建验证平台，分析三个重要指标确保设计代码的功能实现。本文为多通道模式的设计和验证打下基础。如图 2 所示，用于闪存端口操作的 Flash_FSM 如下：

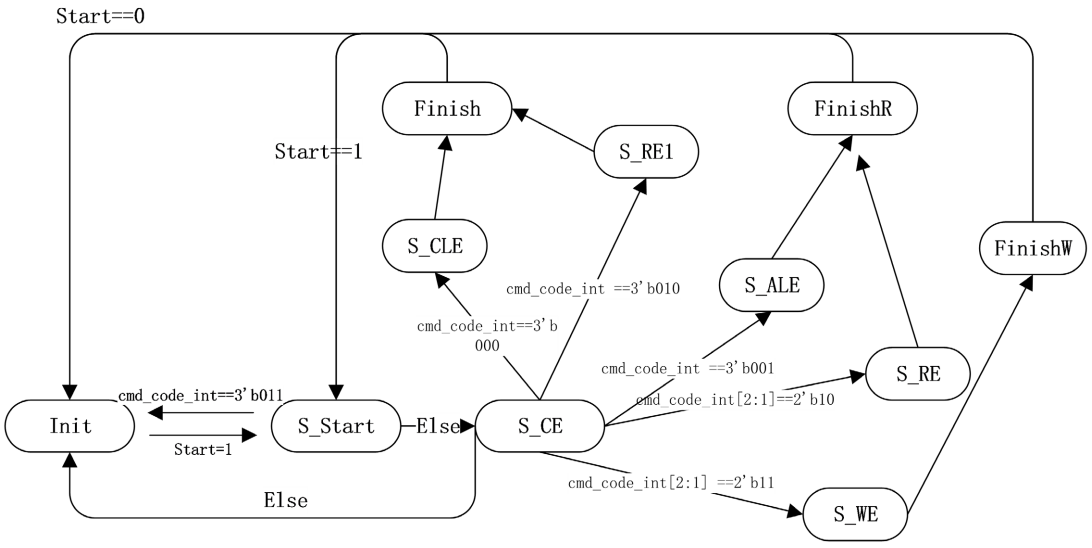

Figure 2. State transition diagram of NAND Flash interface
图 2. NAND 闪存接口的状态转移图

3. 验证方法

Verilog 和 VHDL 语言难以表达对功能覆盖的分析，因为 HDL (Hardware Description Language)是从硬件设计的角度去实现的，这促使形成非常适合验证和设计的新 HDL。System Verilog 就是为解决上述问题而引入的，它具有 HDL 的所有结构以及 C++和 Java 等数据结构语言的许多其他特性[6]。而且支持随机约束的产生、代码覆盖率的检查、功能覆盖点和断言验证。其最突出的优点在于面向对象的编程结构，有助于采用事务级的验证和提高验证的重用性[7]。所以选择 SV 作为验证语言。

3.1. 验证功能点

下面根据 NAND Flash 控制器的设计需求，选择使用 excel 表格的形式列出各项功能点。这种描述方法能简介明了的表达各项内容，工程中常与验证计划相结合，有助于验证 case 的编写，为以后的验证环节打下牢固的基础。除此之外，凭借功能点表格的描述，可以减少验证人员和设计人员的分歧，提高工作效率。表 1 列出部分测试功能点。

Table 1. Part of the test function point analysis table
表 1. 部分测试功能点分析表格

编号	功能点描述	类型
1	BUFFER 的数据读写操作	基础
2	Host 端命令响应及解析	基础
3	写操作命令：program page	基础
4	读 ID 命令：read ID	基础
5	擦除操作命令：erase	基础
6	读操作命令：read page	基础
7	复位操作命令：reset	基础
8	……	……

3.2. System Verilog Assertion

静态验证方法和动态验证方法都不能检查 NAND 闪存接口的时序问题，所以需要一种针对接口时序的验证方法。基于断言的验证(Assertion Based Verification)是当今验证技术中一种主流验证技术[8]，因为其可以在短时间内更快找到设计中存在的功能缺陷。本文选择断言技术搭建验证环境，以此来提高验证环节中检错的效率。

SVA 是 System Verilog 语言的重要组成部分。其作为新的一代标准化硬件描述验证语言，能够精确地描述和判断时序，反应时序中不正确的关系。SVA 本身简洁易读容易维护，有助于设计工程师和验证工程师对复杂设计的行为进行定义。System Verilog 对断言的支持是为了验证此类复杂的设计目标[9]。

以 NAND Flash 控制器系统中第 1 个测试功能项为例，根据异步 FIFO 数据传输协议，采用 System Verilog 编写断言。当 reset 复位时，地址寄存器和满信号应该为 0，空信号为 1；具体断言代码如下：

```
property wrstProp;
    @(posedge wclk) !(wrst_n) |-> (wfull==0);
Endproperty
property rrstProp;
    @(posedge rclk) !(rrst_n) |-> (rempty==1);
Endproperty
```

3.3. 覆盖率

集成电路之初，工程师写完设计代码之后，会编写测试 case 去测试设计代码的正确性；但随着工程的规模越来越大，设计工程师无法兼任设计和验证庞大的工作量；与此同时，对验证需求、正确率和效率的要求不断增加，促使验证技术不断创新。验证的目标是彻底的验证设计代码(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。

代码覆盖率是基础的衡量标准。测试设计代码的执行情况，看重的是每行代码是否被执行，而不是针对设计功能。衡量标准细分为行覆盖(line coverage)、翻转覆盖(toggle coverage)、分支覆盖(branch coverage)、条件覆盖(condition coverage)等。

功能覆盖率是衡量验证完备性的重要指标。功能覆盖率和设计意图息息相关，其往往基于验证计划而设计的。重点在于收集信息而不是数据，这里的信息指设计功能点是否被执行等。如下面代码所示，功能覆盖率关心 nfc_cmd 所指的信息是否全被覆盖，而不是执行次数。

```
covergroup Mix_Inputs;
Commands: coverpoint bfm.nfc_cmd {
        bins CMDS = {program_page, read_page, erase, reset, read_id};
}
endgroup
```

如果某个模块功能在设计中被遗漏，代码覆盖率不能发现这个错误，但功能覆盖率可以。且随着时间的推移，验证 case 越来越丰富，一个项目的漏洞会不断减少。虽然建立新的验证 case 会短暂拉高漏洞率，但会大幅度减低漏洞率。

断言覆盖率主要检查设计代码中的时序正确性。断言是显式的声明一段时间内，一个或两个信号之间关系的代码。所以常常用于查找错误，例如两个信号是否互斥访问或者请求是否被许可，或者是用 assert property 语句对 FIFO、算法等硬件模块做检查[10]。现实工程中，断言常常占整个设计的比例至少大于 30%，其不仅可以与设计平台一起仿真，还可以与形式验证工具所结合，大大提高验证工作的效率[11]。代码具体如下：

```
property StartCheck;
   @(posedge clk) (!(rst_n)) |-> (nfc_start==0);
endproperty
asrtStasrtCheck: assert property (StartCheck) else
   `uvm_fatal("Assrt","start is not 0, when reseting");
cvrStartCheck: cover property (StartCheck);
```

3.4. 验证平台

当 DUT 视为黑盒来验证时，测试失败则无法更深层次地定位问题；所以采用灰盒验证的方式将监视器、参考模型一同用来完善验证[12]。本文采用 System Verilog 中 class 类搭建验证组件，不仅能十分高效的完成组件功能，而且提供了宝贵的可扩展性。此外验证组件以不同的形式创建，与设计环境交互时会有不同的表现[13]，具体表现在健壮性、耗时等指标上。综上考虑，验证组件分别是：Driver 主要负责产生激励，如发送数据、发送命令等，一方面传递给 DUT，另一方面发送给参考模型；Interface 负责 DUT 和验证平台的数据交互，内部包含 input、output 信号，而且用 clocking block 减少竞争现象；Refmodel 实现 DUT 相似功能的命令解析和数据处理，然后发送给 Scoreboard；Scoreboard 实现数据自动化对比。其收到两份数据，一份来自 DUT 实际数值，另一份来自参考模型模拟实现的数值，最终判断操作是否正确。Monitor 负责监控 DUT 的输出数据。在 if 条件下抓取符合条件的数据，然后送给 Scoreboard。K9f1208 是 Verilog 实现的闪存存储模型，模拟现实中的数据存储。Env 是容纳这些组件的顶层，其作用是与 basic_case 连接，将数据/命令生成器与 Driver 连接，与此同时提供 run()任务，调起整个平台工作。如图 3 所示。

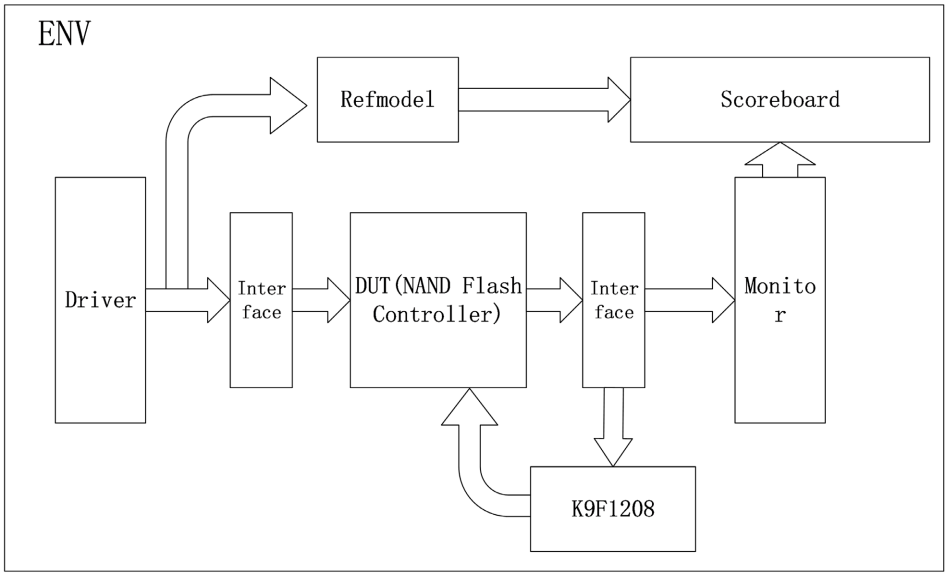

Figure 3. Verification platform construction
图 3. 验证平台搭建

平台的工作过程大致如下，顶层 tb 中调用 case 的 run()任务，先启动 env 的环境，让所有组件一起运行，等待数据发送到 driver；然后启动数据/命令生成器发送数据到 driver；当 driver 接收到数据就会立即发送给 interface。到此环境运行起来。任务 run()代码如下：

```
task run();
    fork
        env.run();
    join_none
    fork
        generater();
    join_any
    $finish()
endtask
```

4. 验证结果

在验证工具 Questasim 10.6c 上，通过 System Verilog 建立的验证环境对 NAND Flash 控制器的功能进行了完整测试。并通过 Vcover merge 命令实现覆盖率合并到一个 ucdb 文件中，如图 4 所示，方便我们的分析总结。

在以覆盖率为导向的验证计划过程中，收集覆盖率是非常重要的指标。验证平台收集的覆盖率包括代码覆盖率、功能覆盖率和断言覆盖率。代码覆盖率借助验证工具，只对 RTL 设计代码进行收集；功能覆盖率先通过编写覆盖组 covergroup，对功能点进行信息描述，再利用系统函数 sample()收集覆盖率；断言覆盖率通过 assert property，判断断言是否成功，然后用 cover property 收集当前覆盖率。

图 4 所示为回归测试后，总体覆盖率数据。%HIT 表示收集到的 Hits 信息占总体 Bins 的百分比，Weight 显示个个覆盖率点权重为 1，覆盖率之间优先级一样。由图可知总体覆盖率已经达到 90%以上，且功能覆盖率和断言覆盖率均达到 100%，表示各功能均被实现。断言覆盖率检查控制器时序等都没有错误。SVA

断言技术是验证平台的一个优势，不仅比传统波形分析更加直接，而且以数据化直观表示，进一步证实验证的完备性。

Coverage Summary by Type:

Total Coverage:					93.77%	**92.89%**
Coverage Type	**Bins**	**Hits**	**Misses**	**Weight**	**% Hit**	**Coverage**
Covergroups	33	33	0	1	100.00%	**100.00%**
Directives	2	2	0	1	100.00%	**100.00%**
Statements	1322	1288	34	1	97.42%	**97.42%**
Branches	500	464	36	1	92.79%	**92.80%**
FEC Expressions	8	8	0	1	100.00%	**100.00%**
FEC Conditions	62	39	23	1	62.90%	**62.90%**
Toggles	720	648	72	1	90.00%	**90.00%**
Assertions	22	22	0	1	100.00%	**100.00%**

Figure 4. Statistical analysis of coverage
图 4. 覆盖率统计分析

5. 结束语

本文基于 System Verilog 语言搭建验证平台，采用收集代码覆盖率、功能覆盖率和断言覆盖率等指标，对 NAND Flash 控制器进行验证。该验证平台通过 System Verilog 随机化语法构造随机激励，确保 RTL 代码被充分被执行。一方面通过分析覆盖率数据，可以发现设计的不足，这显著提高了验证的工作效率；另一方面结合 SVA 断言对 DUT 进行监控，进一步提升验证质量。在功能和断言覆盖率都达到 100% 时，证明在同步时钟单通道模式下，该设计的控制器能够正确控制 NAND Flash 存储器的操作行为，也为多通道模式打下基础。此外证明基于 SV 功能验证策略的可行性，而且凭借其可重用性和可扩展性，也将用于后续的验证研究。

参考文献

[1] Wang, Y., *et al*. (2016) A Real-Time Flash Translation Layer for NAND Flash Memory Storage Systems. *IEEE Transactions on Multi-Scale Computing Systems*, **2**, 17-29. https://doi.org/10.1109/TMSCS.2016.2516015

[2] Yadu, K.K. and Bhakthavatchalu, R. (2019) Block Level SoC Verification Using Systemverilog. 2019 3*rd International Conference on Electronics, Communication and Aerospace Technology* (*ICECA*), Coimbatore, 12-14 June 2019, 878-887. https://doi.org/10.1109/ICECA.2019.8821909

[3] 龙永萍. 基于 UVM 的 Flash 控制器模块验证[D]: [硕士学位论文]. 杭州: 杭州电子科技大学, 2018.

[4] Khalifa, K. (2017) Extendable Generic Base Verification Architecture for Flash Memory Controllers Based on UVM. 2017 *IEEE* 21*st International Conference on Computer Supported Cooperative Work in Design* (*CSCWD*), Wellington, 26-28 April 2017, 584-589. https://doi.org/10.1109/CSCWD.2017.8066759

[5] Huang, Q., Wang, Z. and Lu, W. (2020) A Design of Four Dies Parallel NAND Flash Memory Controller Supporting Toggle and ONFI mode. 2020 *IEEE* 15*th International Conference on Solid-State & Integrated Circuit Technology* (*ICSICT*), Kunming, 3-6 November 2020, 1-3. https://doi.org/10.1109/ICSICT49897.2020.9278398

[6] Dwivedi, P., Mishra, N. and Singh-Rajput, A. (2021) Assertion & Functional Coverage Driven Verification of AMBA Advance Peripheral Bus Protocol Using System Verilog. 2021 *International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies* (*ICAECT*), Bhilai, 19-20 February 2021, 1-6.

https://doi.org/10.1109/ICAECT49130.2021.9392518

[7] (2009) IEEE Standard for System Verilog—Unified Hardware Design, Specification, and Verification Language—Redline. IEEE Std 1800-2009 (Revision of IEEE Std 1800-2005)—Redline, 11 December 2009, 1-1285. https://doi.org/10.1109/IEEESTD.2009.5354441

[8] [美] Srikanth Vijayaraghavan, Meyyappan Ramanathan. System Verilog Assertions 应用指南[M]. 陈俊杰, 译. 北京: 清华大学出版社, 2007: 5-6.

[9] Biswal, B.P., Singh, A. and Singh, B. (2017) Cache Coherency Controller Verification IP Using Systemverilog Assertions (SVA) and Universal Verification Methodologies (UVM). 2017 11*th International Conference on Intelligent Systems and Control* (*ISCO*), Coimbatore, 5-6 January 2017, 21-24.

[10] 阎芳, 李翔, 徐双平, 范毓洋, 田毅. 基于 SVA 的 AFDX 网络 MAC IP 核功能验证[J]. 电子技术应用, 2020, 46(7): 70-73+77. https://doi.org/10.16157/j.issn.0258-7998.191349

[11] 柯志鸣. 基于 SystemVerilog 的芯片时钟模块验证[D]: [硕士学位论文]. 沈阳: 辽宁大学, 2021. https://doi.org/10.27209/d.cnki.glniu.2021.000763

[12] 张瑞. 基于断言的形式化验证与 UVM 的综合应用[D]: [硕士学位论文]. 西安: 西安电子科技大学, 2018.

[13] (2020) IEEE Standard for Universal Verification Methodology Language Reference Manual. IEEE Std 1800.2-2020 (Revision of IEEE Std 1800.2-2017), 14 September 2020, 1-458. https://doi.org/10.1109/IEEESTD.2020.9195920