基于功能虚拟原型的验证方法

江洪波[1]，黄 盈[2]

(1. 华中科技大学电子与信息工程系 武汉 510006; 2. 电子科技大学计算机科学与工程学院 成都 610054)

【摘要】为解决片上系统验证和设计不能同步、系统级验证效率低下的问题，该文基于统一验证方法提出一种基于可演化模型的三级验证过程模型。该模型由系统级、行为级和RTL级三级功能虚拟原型演化模型构成，在不同设计阶段复用相同的系统级验证环境，可减少验证的重复工作，将其应用于设计的整个流程，可成功地实现验证和设计同步，提高验证效率。

关 键 词 仿真; 功能虚拟模型; 片上系统; 统一验证方法

中图分类号 TP391.9 文献标识码 A

Verification Methodology Based on FVP

JIANG Hong-Bo[1] and HUANG Ying[2]

(1. Department of Electronics and Information Engineering, Huazhong University of Science and Technology Wuhan 510006;
2. School of Computer Science and Engineering, University of Electronic Science and Technology of China Chengdu 610054)

Abstract Given the complexity of the functionality of system on chip (SOC), one of the main challenges of current IC design is no more than verification. We implemente a functional virtual prototype (FVP) according to the blueprint of unified verification methodology to solve the synchronization between design and verification in SOC development. The FVP is based on three levels of models: system-level, behavior-level, and register transfer level (RTL). The efficiency of verification can be improved by using FVP as a system-level model.

Key words emulation; functional virtual prototype; system on chip; unified verification methodology

随着芯片生产工艺的快速发展，片上系统(system on chip，SOC)设计落后于生产工艺，即对片上系统设计的验证不够充分的问题[1-2]，成了制约SOC设计发展的关键瓶颈问题。

在传统的开发方法中，芯片开发被分成了互相独立的子系统阶段，每个阶段都有明确的开始和完成的衡量标准[3]，测试与设计不同步，验证工作滞后，子系统开发团队不得不自己开发验证测试环境。因为没有系统级测试环境，子系统只有在集成测试时才能够发现模块间的错误[4]。传统开发流程使得发现错误、修改错误的代价非常高，严重地影响了项目开发的正常进度。传统验证方法的另外一个缺点是系统验证时的效率非常低，通常只有模块验证的10%。

本文基于功能虚拟原型(functional virtual prototype，FVP)建立了一个系统级的验证模型，解决了验证和设计无法同步，以及验证效率低下的问题。采用该模型，验证工程师不但在项目早期就可以开发验证环境，而且可将FVP服务于整个设计开发过程。

1 功能虚拟原型(FVP)

FVP是由Cadence Design Systems在统一验证方法(unified verification methodology，UVM)中首先提出[5]，目的是建立一个统一高效的、开发环境与开发流程无关的测试环境。FVP是一种指导性框架，每个开发团队可以根据UVM蓝皮书上建议的策略和自己所使用的技术、工具，建立具体的验证模型。本文提出的FVP模型是基于C/C++、SystemC和HDL的三级演化模型。

FVP由系统架构师和验证工程师共同开发完成。验证工程师基于系统架构师和设计师提出的功能、特性和性能，提出测试计划。在UVM方法中，验证工程师在项目早期就可以开发验证环境，缩短了整个项目开发的周期。

FVP是设计和验证的系统级模型。它与传统的只分析系统结构和性能的系统级模型的最大的区别是，FVP包含了功能验证，减少了开发系统级验证

收稿日期：2007－06－04；修回日期：2007－12－08
作者简介：江洪波(1968－)，男，在职博士生，主要从事计算机应用、片上系统方面的研究.

环境的工作量。另外，FVP是一个不断演化的模型，FVP模型被用于系统开发的各个阶段。随着开发的深入，模型内部的模块被各个实现模块所代替，逐步演化成具体的实现。

2 三级FVP模型

本文提出的三级FVP模型的基本结构如图1所示，它是基于系统级、行为级和RTL级的三级演化模型，由验证环境、接口监视和功能模块组成。每个功能模块都被接口监视器封装，功能模块和其他模块以及验证环境之间的数据交互都通过接口监视器来完成。接口监视器还负责完成数据转换功能，即将特定的功能模块的数据转换成统一的事务级数据。通过对功能模块的封装，验证开发工程师和子系统开发工程师可以同步开发，不需要考虑模块的内部实现。

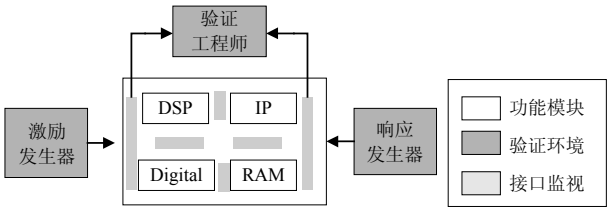

图1　FVP模型结构

FVP的高级模型通过模块替换可以向低级模型演化，而不需要开发全新的低级模型。FVP高级模型是事务级模型，如图2所示。

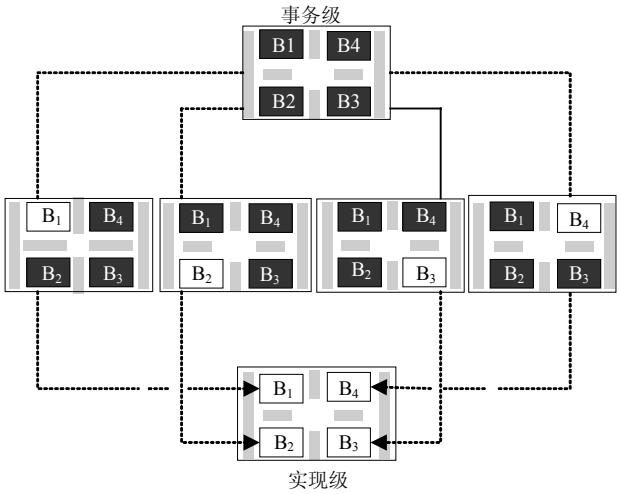

图2　FVP由高级模型向低级模型演化过程图

子系统开发团队用各自的实现模块替换事务级模型中的对应模块(即图中的B_1～B_4模块分别被替换)，然后使用FVP提供的验证环境测试子模块的功能和接口是否满足系统的要求。通过验证后，功能模块被提交给整个开发团队使用。在替换过程中，子模块替换是独立进行的，不需要与其他开发团队同步。模块替换完成后，开发团队不需等待集成测试完成，就可以继续下一级模型的开发。当所有同级的子模块都实现后，就更换模型中的所有功能模块，利用FVP的验证环境完成集成测试，事务级的FVP就被演化到了实现级的FVP。整个演化过程中，只有功能模块需要更换，验证环境不需要变换。

3 系统级FVP

FVP的系统级模型是基于事务级的。事务级验证可以极大提高验证生产率，很容易实现自核对和定向随机测试。本文提出的构架中，系统级模型使用C/C++实现。

实现系统级模型的主要难点在于如何确定系统的事务和如何实现总线功能模型。本文提出的模型中，使用C++对象的概念表示总线。总线对象提供接口函数给其他功能模块实现总线功能的调用。事务的实现也是基于对象的接口，每个对象的接口就是一种类型的事务。

FVP模型支持混合仿真，因此用C/C++构建验证程序既可以满足效率的要求，也可以很方便地实现系统级模型和低级模型之间的通信。另一方面，C/C++支持复杂的数据结构，因此用C/C++描述测试向量比用传统方法描述更加直观、简洁和高效[6]，可以大大减少验证工程师开发、检查和维护的时间。

系统级FVP可以作为IC设计中的可执行规范以及以后其他模型的黄金模型来使用。系统级FVP被同时提交给系统分析师、子系统开发工程师、验证工程师和软件开发人员，这在传统的开发方法中是不可能的。传统方法中验证工程师只有等待RTL级模型完成后才能开始构建验证环境。使用FVP模型可以使硬件开发和构建验证环境同时进行，因此极大地缩短了SoC芯片的设计开发周期。

3.1 行为级FVP

行为级FVP用于实现行为级模型和系统级模型的通信，SystemC能够很容易地实现上述目标。

SystemC实质上是在C++的基础上添加硬件扩展库和仿真核[7]。因为SystemC可以建模不同抽象级别硬件的复杂系统，SystemC和C/C++又有很强的交互能力[8]，所以本文提出的构架中的FVP行为级模型选择SystemC作为描述语言。通过在接口监视器里增加行为级的信号到系统级的事务转换模块，实现混合模型通信。

如前所述，行为级模型的实现可以直接使用系统级验证环境。开发工程师用SystemC实现的行为级

模型替代FVP中对应的子功能模块，然后使用FVP系统级验证环境测试行为级模型是否满足系统的要求。这样做的优点是：(1) 可以减少工程师开发验证环境的工作量；(2) 可以减少以后进行集成测试时出现错误的可能性。

因为FVP作为系统级模型和验证环境已经为子系统提供了集成所需要的测试环境和测试向量，如果行为级模型在FVP中验证通过，那么在以后的集成中也不会有太大问题。

3.2 RTL级FVP

本文提出的FVP模型用Verilog HDL作为RTL级的描述语言。RTL级模型和前面的两级模型的混合仿真是FVP能否成功的关键[8]。

实现Verilog和SystemC以及C/C++通信需要依靠具体工具的支持。本文利用ModelSim支持混合语言仿真的特性，解决RTL级模型和系统级、行为级模型的协同仿真。

如图3所示，Verilog和其他模块的通信通过Wrapper进行。Wrapper就是前面提到的数据转换接口，但是需要加入一个Foreign Class。Foreign Class是Verilog模块在SystemC中的接口。SystmeC的其他模块要访问Verilog模块可直接访问Foreign Class的对象，而Foreign Class和Verilog模块的连接在仿真时由ModelSim动态完成。

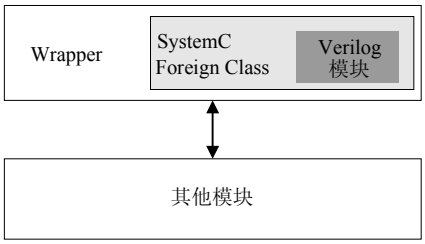

图3 Verilog模块与SystemC通信图

通过Wrapper实现对RTL模块的包装，将模型的实现和外部接口完全分离，可以方便不同开发小组的使用。

4 1553B芯片验证的FVP模型的实现

本文在1553B航空通信总线控制芯片设计的项目中实现了基于FVP模型的验证。因为项目资源的限制，1553B的FVP模型由三级演化过程简化为两级演化过程，即系统级模型和实现级模型。为了方便IP核开发，IP核的系统级模型没有建立事务级模型，而是直接使用SystemC生成行为级模型，因此IP核的接口都是信号层次。

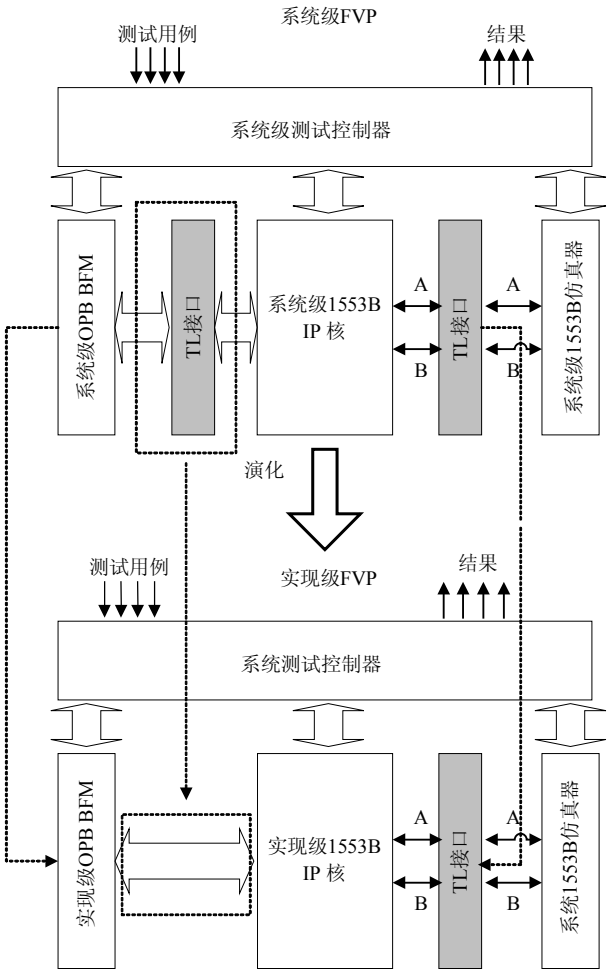

图4 1553B FVP模型演化

在1553B的FVP两级模型演化过程中，只需要替换FVP模型中的OPB总线功能模型以及IP核级模型，如图4所示。系统级FVP和实现级FVP之间的差别在于1553B IP核模型和OPB总线功能模型是使用不同的硬件描述语言来实现的。系统级的FVP基于SystemC实现，而实现级的FVP的OPB总线功能模型和IP核级模型使用Verilog进行描述。

1553B系统级FVP模型选择SystemC作为其主要的仿真描述语言。1553B系统级FVP包含4个部分，其中测试台的所有模块都是事务级模块，而IP核则是行为级模型，如图4所示。IP核使用非事务级模型，主要是为了方便验证IP核的外部信号级接口，可以使IP核的接口在系统级验证阶段就能够确定，方便进一步的验证开发。

测试台中最重要的模块是OPB总线功能模型。OPB总线功能模型主要是提供OPB总线读写事务以及一些特定的总线操作事务。通过借用C++面向对象的概念，本文将OPB总线功能模型定义为一个类对象，而所有的总线事务都定义为类对象的接口函

数。与此类似的是，1553B总线仿真器也是一个1553B总线的功能模型，因此可以同样地通过类对象来描述。

根据FVP模型实现中提出的方法，对1553B IP核进一步的开发是使用标准的硬件描述语言，本文选择Verilog。测试台继续使用系统级模型的模块，但是需要开发对应的事务级转换接口。因为本项目的系统级IP核模型的外部接口是信号层次，因此不需要再单独地开发对应的事务级转换接口。在实现级的FVP开发中，主要的工作就是开发验证的测试用例，以及在系统级FVP中验证测试用例。

对传统的从顶向下的验证方法和基于FVP模型的验证方法的比较如表1所示。通过1553B芯片项目的验证，可以证明基于FVP模型的验证方法能够提升仿真验证的生产效率。

表1　验证方法对比表

	基于FVP模型的验证方法	传统由上至下的验证方法
验证抽象层次	系统级、行为级和RTL级验证都使用事务级验证，提高验证的抽象层次，使得验证效率得到提升。	系统级使用事务级验证，行为级和RTL级验证使用效率比较低的基于事件或者基于周期的验证。
功能验证开始时间	功能验证可以在系统级模型开发时就并行开始，以缩短整个设计验证的周期。	功能验证只能在行为级模型实现后才能够开始。
验主功能等价性	是	是
验证模型复用	支持系统级、行为级、RTL级验证验证模型之间的复用。	每级模型都需要开发对应的验证模型，无法复用。

5 结　论

传统的SoC设计方法中，验证工作和设计不能同步，必须在设计完成后才能够开始，系统级验证效率低下，使验证成为整个设计过程的瓶颈。本文针对这些问题，采用基于系统级、行为级和RTL级的三级演化模式，实现了UVM方法提出的FVP模型。该模型可在SoC系统芯片验证中作为系统级模型，使验证工程师在项目的早期就可以开始对验证环境的开发，从而缩短整个项目的开发周期。同时，该模型的FVP验证环境是事务级的，验证效率也得到极大提高。

目前该模型已基于ModelSim和SystemC库实现。为提高其适应性，还需要把FVP实现扩展到更多的工具。另一方面，SPI模型比较简单，因此需要在更复杂的SoC项目中验证FVP模型在提高验证效率和速度方面的优点。

参　考　文　献

[1] Collett International Research. 2000, 2002 functional verification studies; 2003 design closure study[R]. [S.l.]: Collett International Research, 2004.

[2] 詹瑾瑜, 熊光泽, 桑　楠. Formal Co-verification for the correctness and timing requirements of SoC design[J]. 四川大学学报, 2005, 37(2), 93-98.

[3] Khan A, Recent developments in high-performance system-on-chip IC design[C]//In: Proceedings of IEEE ICICDT. Austin Texas: [s.n.], 2004.

[4] 韩俊刚. 系统级芯片设计语言和验证语言的发展[J]. 现代电子技术, 2005, 28(3): 1-4.

[5] Rashinkar P, Paterson P, Singh L. System-on-a-chip verification: methodology and techniques[M]. [S.l.]: Springer, 2000.

[6] Clarke E, Kroening D, Yorav K. Behavioral consistency of c and verilog programs using bounded model checking[C]// Design Automation Conference. [S.l.]: [s.n.], 2003.

[7] 陈　曦, 徐宁仪. SystemC片上系统设计[M]. 北京: 科学出版社, 2004.

[8] RANDJIC A, OSTAPCUK N, SOLDO I, et al. Complex ASICs verification with systemC[C]//PROC. 23rd International Conference on Microelectronics(MIEL 2002). YUGOSLAVIA: [s.n.], 2002.

[9] JINDAL R, JAIN K. Verification of transaction-level systemC models using RTL testbenches[C]//In: Proceedings of the First ACM and IEEE International Conference on Formal Methods and Models for Co-Design. [S.l.]: [s.n.], 2003.

编　辑　熊思亮