SoC 软硬件协同验证技术的应用研究

方应龙[1],赵 勇[1],幸 强[2]
(1.北京大学 深圳研究生院信息学院,广东 深圳 518055;
2.东南大学 电子工程系,江苏 南京 210018)

摘 要:介绍了软硬件协同验证的原理,给出了笔者在实际 SoC 开发中采用的四种软硬件协同验证方案。根据软硬件协同仿真原理提出 CFM方案。对几种方案进行比较,并提出在实际 SoC 设计中选取软硬件协同验证方案的建议。

关键词:软硬件协同验证 ISS 方案 CVE 方案 CFM 方案 FPGA/EMULATOR 方案

软硬件协同验证的概念虽已提出多年,但直到近些年随着 SoC 技术的发展,软硬件协同验证技术才得到更多的关注和重视,并得到发展。软硬件协同验证技术是指在最终硬件没有准备好之前进行软件和硬件的协同验证,其目的是希望在设计的早期验证系统软硬件的正确性,特别是功能的正确性和性能的高效性[1]。

目前,软硬件协同验证技术对低层次仿真的研究比较成熟,其一般模式是软件调试环境、微处理器模型和硬件 RTL 描述的协同仿真。它可以在尚未生产出硬件之前对虚拟模型进行早期调试,并为软件调试提供虚拟平台,从而对包含硬件在内的整个系统进行功能验证,节约了准备硬件平台的时间。由于考虑了软件实际运行的情况,所以验证能够更接近实际应用环境,更容易发现设计中的问题,避免早期设计错误,降低设计风险以及设计对交货期的影响。

1 软硬件协同验证简述

传统的验证是软件和硬件分开验证。在硬件设计周期,进行硬件验证,而软件验证必须等到硬件平台搭建好后才能进行。例如,一个 SoC 系统的验证,必须等到芯片流片之后才验证软件,这样设计周期非常长,验证效果也不理想。软硬件协同验证技术的提出改变了这种局面[2]。

软硬件协同验证的核心是处理器。处理器是软硬件交互的界面,它一方面负责执行软件,另一方面通过系统接口访问外设和内存。CPU 是将硬件仿真与软件执行相结合的关键,通过 CPU 模型(虚拟 CPU),用硬件仿真器实现软硬件协同验证的仿真。软硬件协同验证系统通常包括图 1 所示的四个部分。这四部分加上仿真器就构成了完整的软硬件协同仿真环境。

图 1 软硬件协同验证系统

其中,处理器执行软件并与外设内存通信,其可以是真实的 CPU、RTL 级的加密 CPU 核、指令执行的仿真器或者是 BFM。调试器用来进行软件调试。可以用软件调试器实现,也可以用硬件调试程序实现,还可以在硬件仿真器的环境下通过检查执行后的硬件波形间接了解软件执行情况。外设和内存是硬件部分,可以是 C 模型、RTL 级硬件描述,也可以是 FPGA 的原型实现。系统接口是连接处理器和外设内存的界面。

不同的模块及仿真器造成了软硬件协同验证方案的不同。下面对几种软硬件协同验证方案进行介绍。

2 软硬件协同验证的实际应用方案

在实际应用中,有四种软硬件协同验证方案:ISS、CVE、FPGA、EMULATOR。此外,笔者在实际应用中,根据项目情况,提出一种 CFM 方案。

2.1 ISS 方案

ISS 方案采用 ISS(例如 ARMulator[3])代替 CPU 执行软件,并通过接口与外设及内存通信。该方案中的外设模型一般用 C 语言建立,形成如图 2 所示的结构。其中 ISS 可以是指令级精确的,也可以是指令周期级精确的,还可以是具有硬件系统的时钟级精确,又称为总线功能模型(BFM)。但是,如果所有的外设模块都是 C 模块,

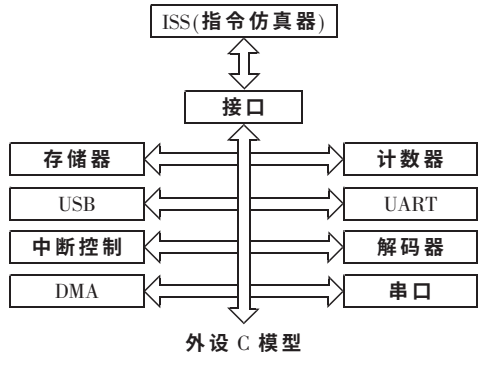

图 2 ISS 方案结构图

则整个系统不能达到时钟级精确度。通常 ISS 都带有调试程序,用于控制 ISS 执行指令。

2.2 CVE 方案

CVE 方案以 Seamless 的 CVE 软件为基础,是一种使用两个仿真器进行仿真的方案。CVE 通过自身的一个内核将软件仿真与硬件仿真结合。它支持多 CPU 的模型,具有高效的软件调试能力,可以单步执行 CPU 指令,随时查看寄存器和内存的情况;同时,它提供了强大的信号观测能力,调试者可以通过设置断点、触发条件等进行有效调试。图 3 是 CVE 方案的系统结构图。

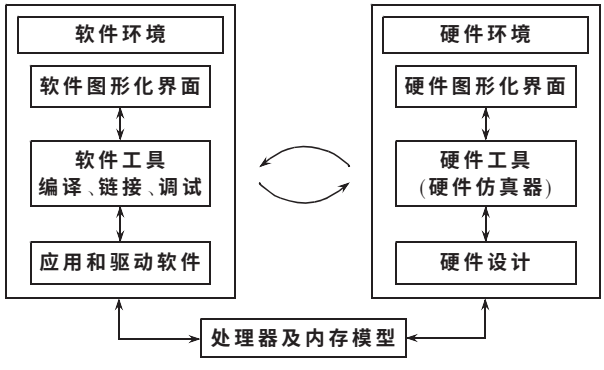

图 3 CVE 方案结构图

2.3 FPGA/EMULATOR

FPGA/EMULATOR 方案是将外设、内存等部件综合后用 FPGA 实现,而 CPU 则用真实的 CPU 或 FPGA 本身集成的 CPU 实现。这些系统一般可以加一个专门的硬件仿真器进行调试,控制 CPU 的执行,并读取系统状态。FPGA 是最典型的原型技术。EMULATOR 与 FPGA 相比的最大特点是集成了多个 CPU 及多个 FPGA,其功能更强,但造价更高。图 4 为此方案结构图。

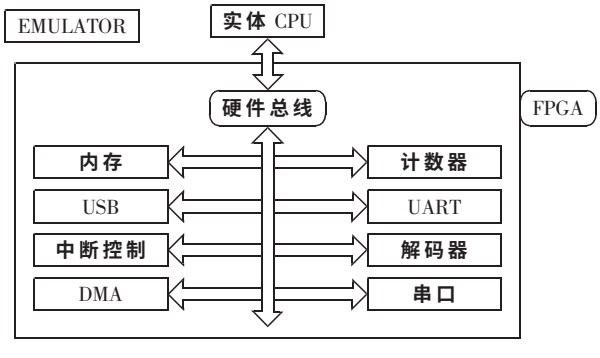

图 4 FPGA/EMULATOR 方案结构图

2.4 CFM 方案

CFM 方案是笔者在实际项目进程中,根据需要并依据协同仿真原理采取的方案。如图 5 所示,CFM 所用的 CPU 是软 IP 核,一般都是加密的,外设及总线都是 RTL 级的代码。仿真开始前,通过与 CPU 相对应的编译工具将软件程序编译成 CPU 可识别的二进制文件,存放在系统的 ROM 中;仿真开始后,CPU 通过读取 ROM 中的二进制文件运行代码,即可利用硬件仿真器实现软硬件

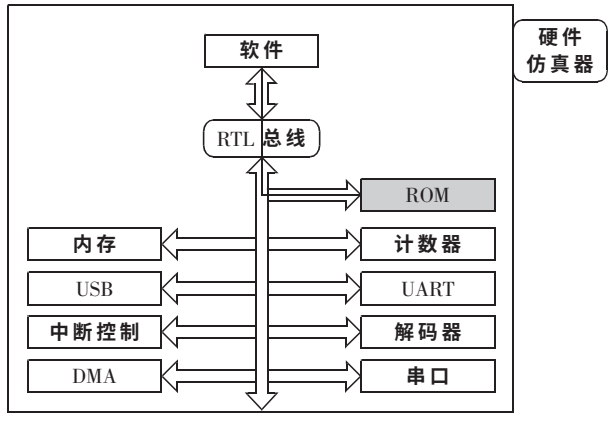

图 5 CFM 方案结构图

的协同验证。这种方法,速度比 CVE 快,价格比 FPGA/EMULATOR 便宜,比 ISS 方案更易于发现硬件错误。但如果仿真过程中截取较多的波形,则速度比 ISS 慢,实际应用中应采用硬件触发方法进行调试。仿真过程中,如果结果正确,就向某个寄存器写入特定数据,并通过 verilog 系统函数打印信息;否则写入另外一个值,或者写入另外一个寄存器,并打印错误信息,这样不用读取波形即可达到调试目的。通常在协同仿真前,硬件模块级的验证都已经完成,因此,采用这种方法也可以加快协同仿真验证的速度。在实际项目中,采用过多种 CPU 甚至 DSP 搭建上述环境,结果都很成功。

3 几种方案的比较

以上方案各有优缺点,本文从几个方面进行比较。

(1)验证速度

在仿真速度方面,FPGA/EMULATOR 采用硬件实现,其速度最快;ISS 方案速度中等;若 CFM 方案在 Testbench 中只取少量波形,则速度与 ISS 接近;而 CVE 方案则最慢。

(2)时间精度

在时间精度方面,FPGA/EMULATOR、CFM、CVE 都可以达到时钟级精确,而 ISS 方案在一般情况下精确度不是很高,大部分为指令级精确。

(3)调试性能

ISS 方案一般用来验证系统在算法级上的正确性,对硬件的调试帮助不大;CVE 方案具有最强大的调试性能,支持软件的单步执行,随时查看寄存器、内存状态,还可以用硬件仿真器生成波形来调试硬件。因此无论硬件还是软件,CVE 的可调试性能都非常高;FPGA/EMULATOR 的调试性能比较差,它一般是在经过仿真器仿真后再做原型验证,调试时需要通过增加 JTAG 之类的工具才能看到内部状态;CFM 则与 ISS 相反,其对软件的调试能力不是很强,只能通过硬件间接观察软件的运行情况。

(4)准备工作

ISS 方案需要外设的 C 模型,一般通过工程师做大量前期积累或者其他途径得到;CVE 方案只需做好软硬件配置即可,相对工作量最少;FPGA/EMULATOR 需将

(接上页)

硬件下载到 FPGA 中，只有少量的准备工作；CFM 方案需要在 Testbench 中做信号触发以方便调试，准备工作也较少。

(5)价格成本

ISS 方案成本不高，需要一个 ISS 和一些外设模型；CVE 方案需要购买 Seamless CVE 软件，比 ISS 成本高；成本最高的是 FPGA /EMULATOR，硬件购买费用很高；成本最低的方案是 CFM，只需要硬件仿真器即可实现。

(6)适用范围

ISS 方案比较适合算法验证和具有一定程度的硬件系统调试。CVE 方案适合软硬件的联合调试，尤其在系统未经过充分验证时，需要较好的性能调试。FPGA/EMULATOR 适合在经过仿真器验证无误之后，用于原型验证。CFM 方案较适合做回归测试，以及经过一定程度验证及对系统做少量修改后的测试。

各种方案的性能比较如表 1。

表 1 各种软硬件协同验证方案的比较

方案	验证速度	时间精度	调试性能	准备工作	价格成本	适用范围
ISS	中等	指令级	中等	大量	低	早期
CVE	慢	时钟级	高	少量	中等	中期
FPGA/EMULATOR	快	时钟级	低	中等	高	后期
CFM	中等	时钟级	中等	中等	低	中期

软硬件协同验证技术是 SoC 设计的一项关键技术，其中 CPU 的模型选择最为重要。在实际项目中，要根据项目的需要，考虑速度、性能、成本、消耗资源等诸多因素，选择一种或者多种方案来进行验证，加快验证收敛时间，节约验证成本。

参考文献

1 Tuck Barbara.The hardware/software co-verification challenge. Computer Design, 1998;(4)

2 Dreike Phil, McCoy James.Co-simulating software and hardware in embedded systems.Embedded Systems Programming, 1997;(6)

3 The ARMulator.Application note 32.http://www.arm.com.

4 Seamless CVE.Hardware/software co-verification technology. http://www.mentor.com.

5 Kresta Dave, Johnson Tony.FPGA High-level design methodology comes into its own.Electronic Design, 1999;(6)

(收稿日期：2006-06-20)