

基于 SPARC V8 的事务级 SoC 验证平台设计

杨 奕, 付方发, 王晗宇, 来逢昌

(哈尔滨工业大学 微电子中心, 黑龙江 哈尔滨 150001)

摘 要: 针对片上系统 SoC 架构设计和嵌入式软件开发的需求, 采用事务级建模方法使用 SystemC 完成了基于 SPARC V8 的事务级 SoC 验证平台的设计. 为降低设计复杂度和提高仿真速度, 基于解释—执行技术完成 SPARC V8 处理器指令精确事务级模型建模, 并利用 SystemC 中的分层通道机制完成 AMAB 总线、中断控制器、UART、定时器等设备的事务级建模. 完成事务级 SoC 验证平台的构建后, 使用测试基准程序组 Mibench 对该验证平台的功能和仿真速度进行了验证. 仿真结果证明了其功能正确, 并且仿真速度相对于 RTL SoC 验证平台有大幅度的提高.

关键词: SPARC V8; SoC; 事务级; SystemC; 仿真速度

中图分类号: TN402

文献标识码: A

文章编号: 1000—7180(2014)03—0159—05

Design of SPARC V8 Based Transaction Level SoC Verification Platform

YANG Yi, FU Fang-fa, WANG Han-yu, LAI Feng-chang

(Microelectronics Center, Harbin Institute of Technology, Harbin 150001, China)

Abstract: For SoC architecture design and embedded software development needs, use transaction-level modeling method with SystemC to complete the design of SPARC V8 based transaction-level SoC verification platform. To reduce the design complexity and improve the simulation speed, completing transaction level modeling of instruction accurate SPARC V8 processor based on interpretive-execution technology, using hierarchical channel mechanism in SystemC to design transaction level models of AMAB bus, interrupt controller, UART, timer etc. Using Mibench to verify function and simulation speed of the verification platform after completing building the transaction level SoC verification platform. The results show that its functionality is correct and its simulation speed has greatly improved compared with RTL SoC verification platform.

Key words: SPARC V8; SoC; transaction-level; SystemC; simulation speed

1 引言

随着集成电路设计和制造能力日益发展, 片上系统 SoC(System on chip)设计规模不断提高, 有关 SoC 架构设计及验证已经成为集成电路设计领域的研究热点. 设计规模较大的 SoC, 按照传统的设计流程, 首先需要大量时间进行各个 IP 的 RTL 建模, 在此基础上进行系统级仿真验证和嵌入式软件的开发. 受到 RTL 仿真速度的影响, 需要花费更多的时间对整个系统进行仿真验证. 为了加速 SoC 整体设计及验证工作, 可以对 SoC 中复杂程度较高的 IP,

甚至整个 SoC 进行较高层次的建模.

SoC 按照设计抽象层次的高低可分为算法级、事务级、行为级和 RTL. 事务级建模将模块内部具体行为抽象为事务, 并可将模块间的通信由具体时序协议抽象为无时序的函数调用. 基于事务级建模的 SoC 在保证一定仿真精度的同时, 在仿真速度上比基于 RTL 的 SoC 有很大提高, 如基于事务的总线周期精确(Transaction based Bus Cycle Accurate, T-BCA)模型的仿真速度比 RTL 模型快 2~3 个数量级, 建模复杂度只有 RTL 模型的 1/10 左右^[1]. 完成 SoC 的事务级建模后, 嵌入式软件的开

发与 SoC 的 RTL 建模能够同时进行,大大缩短了设计时间.

国内外对事务级建模进行了大量研究工作,如文献[2]中基于 TLM2.0 标准完成 SPARC 处理器的事务级建模,文献[3]中使用 SystemC 完成事务级 AMBA 总线模型建模.以上研究大多集中于提高处理器或者总线建模的抽象层次以提高仿真速度,在利用其研究成果进行系统仿真时,仍需要对平台进行开发.为此,本文提出一种事务级 SoC 验证平台建模方法,使用 SystemC 作为建模语言,采用指令解释—执行技术完成指令精确的 SPARC V8 处理器建模,利用分层通道机制完成 AMAB 总线、中断控制器、UART、定时器等 SoC 基本组件的事务级建模,并使用上述模型构建事务级 SoC 验证平台.

2 基于 SystemC 的事务级建模方法

SystemC^[4]是基于 C++ 的建模及仿真语言,具有面向对象编程的特点,并且提供了 3 种基本进程,包括方法进程 SC_METHOD、线程进程 SC_THREAD 和钟控线程进程 SC_CTHREAD.基本进程可被诸如时钟信号上升沿的事件触发,并可通过调用方法 wait()、next_trigger() 控制进程的执行或产生延时,故使用 SystemC 可描述硬件的时序行为,完成系统级软/硬件协同设计.

寄存器传输级(RTL)或行为级模型为接口精确,模型之间依靠信号进行通信,仿真速度较慢.事务级模型之间的通信依赖于事务,将具体信号的时序行为抽象为函数调用过程,能够更有效地传递控制和数据,仿真速度较快.本文使用 SystemC 提供的端口(sc_port、sc_export)、接口(sc_interface)和通道(sc_prim_channel、sc_channel)机制实现事务级模型之间的通信.接口用于声明通道中包含的方法,这些方法在通道中具体实现,通道中的成员变量作为通信的载体,事务级模型中的进程通过特定接口类型的端口调用通道中的方法访问成员变量,从而实现模型间的通信.

通道分为基本通道和分层通道两大类.基本通道只包含特定的接口、方法和成员变量;分层通道具有明确的结构层次,包含特定的接口、端口、方法、成员变量和进程.模块可以通过接口调用分层通道中

的方法,分层通道中的进程可以通过端口调用其他通道中的方法.使用分层通道可以实现复杂的通信协议,完成片上总线的事务级建模.

3 事务级 SoC 验证平台设计

本文设计的事务级 SoC 验证平台的整体架构如图 1 所示,使用 AMBA 总线^[5](包括 AHB 仲裁器、AHB 译码器和 AHP/APB 总线桥等)完成包括 SPARC V8 处理器、中断控制器、UART、定时器等各个设备之间的互连.基于 RTL 的 SoC 验证平台在仿真时,由于涉及处理器通过 AHB 访问片上存储器或片外存储器接口以完成取指令等总线操作,使得仿真速度受到了严重的制约,为此,本文对 SPARC V8 处理器添加了一个附属虚拟存储器,通过访问附属虚拟存储器来完成取指令操作,无需发起总线传输,极大地加快了处理器行为的仿真速度.

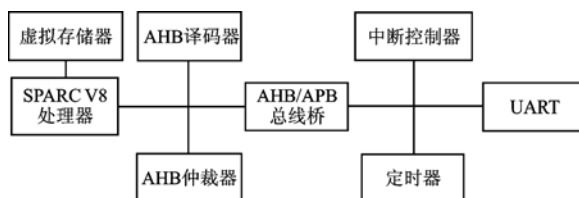


图 1 事务级 SoC 验证平台架构

3.1 基于 SystemC 的 AMBA 总线事务级建模

3.1.1 AHB 事务级建模

AHB 总线在传输时,地址和数据分离,具有流水特性,支持突发传输和多主设备.本文使用分层通道完成 AHB 事务级模型建模,实现总线协议中描述的功能和时序特性,如图 2 所示.该方法使主从设备间的数据交互均通过同一个分层通道完成,相对于使用基本通道(如 sc_signal)建模的方法,减少了通道的数量,可有效提高仿真速度.

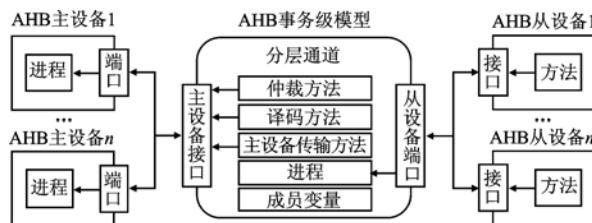


图 2 AHB 事务级模型

主设备接口中声明了实现仲裁、译码和主设备传输方法;分层通道中的成员变量用于保存 AHB 传输地址阶段和数据阶段的信息;进程由时钟边沿触发,通过从设备端口调用 AHB 从设备中实现的

方法进行 AHB 传输,实现 AHB 传输信息的同步与更新。

AHB 主设备通过主设备接口调用分层通道中的仲裁方法申请并获得总线授权,调用译码方法和主设备传输方法发起 AHB 传输,使用成员变量保存 AHB 传输控制、地址和写传输数据。分层通道中的进程根据译码结果通过目标 AHB 从设备接口调用方法对其进行访问;对于读传输,根据传输地址读取相应寄存器变量,使用成员变量保存 AHB 传输响应和读传输数据;对于写传输,根据传输地址将写传输数据写入寄存器变量,使用成员变量保存 AHB 传输响应。由此可见,本方法在保证仿真速度的同时,实现了总线传输协议的细节,提高了仿真的精度。

3.1.2 AHB/APB 总线桥事务级建模

AHB 和 APB 通过 AHB/APB 总线桥相连,AHB/APB 总线桥为 AHB 从设备,同时也是唯一的 APB 主设备。APB 传输固定两个时钟周期完成,无等待周期和传输响应,使用分层通道完成 AHB/APB 总线桥事务级模型建模,如图 3 所示。

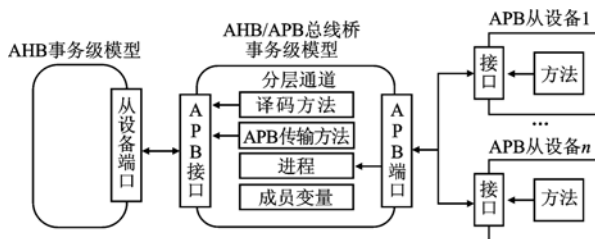


图3 AHB/APB 总线桥事务级模型

APB 接口中声明了译码方法和 APB 传输方法;分层通道中的成员变量用于保存 APB 传输的信息;进程由时钟边沿触发,通过 APB 端口调用 APB 从设备中实现的方法进行 APB 传输,实现 APB 传输信息的同步与更新。

当主设备访问 APB 从设备时,AHB 事务级模型调用分层通道中的译码方法和 APB 传输方法发起 APB 传输,使用成员变量保存 APB 传输控制、地址和写传输数据。分层通道中的进程根据译码结果通过目标 APB 从设备接口调用方法对其进行访问;对于读传输,根据传输地址读取相应寄存器变量,并使用成员变量保存读传输数据;对于写传输,根据传输地址将写传输数据写入寄存器变量。

3.2 基于 SystemC 的 SPARC V8 事务级建模

SPARC V8 是 SUN 公司推出的一种精简指令

集体系结构(RISC ISA),其功能与性能具有可扩展性。处理器指令集模拟器(ISS)根据仿真精度的不同可分为指令精确和周期精确两大类。指令精确的 ISS 实现指令的执行,不实现指令执行过程中具体时序特性,抽象层次较高,仿真速度快,并在功能上具有很高的仿真精度。

3.2.1 SPARC V8 指令集架构

SPARC V8 指令集规范^[6]中定义的寄存器堆包括通用寄存器堆和控制/状态寄存器堆,通用寄存器堆具有窗口特性,相邻的窗口寄存器组共享 8 个窗口寄存器。指令执行时,根据窗口指针的值访问 8 个全局寄存器 %g0~%g7 和 24 个窗口寄存器(包含 8 个输入寄存器 %i0~%i7,8 个本地寄存器 %l0~%l7,8 个输出寄存器 %o0~%o7),程序执行过程中发生的函数调用、函数返回、进入陷阱和陷阱返回都会使窗口指针发生变化,实现参数传递。通用寄存器的数量与窗口寄存器组的数量相关,计算方式为:窗口寄存器组 $\times 16+8$ 。

控制/状态寄存器堆包括 IU 控制/状态寄存器堆和 FPU 控制/状态寄存器堆,具体为处理器状态寄存器 PSR、窗口无效掩码寄存器 WIM、陷阱基址寄存器 TBR、乘法/除法寄存器 Y、程序计数器 PC 和 nPC、浮点状态寄存器 FSR、浮点寄存器 FR、辅助状态寄存器 ASR。上述寄存器的位宽均为 32 位,均可定义为整型变量,作为指令精确 ISS 的成员变量:

```
/* IU 通用寄存器堆 */
uint32 r[NWIN * 16]; /* 窗口寄存器 */
uint32 g[8]; /* 全局寄存器 */
/* IU 控制/状态寄存器堆 */
uint32 PSR; /* 处理器状态寄存器 */
uint32 WIM; /* 窗口无效掩码寄存器 */
uint32 TBR; /* 陷阱基址寄存器 */
uint32 Y; /* 乘法/除法寄存器 */
uint32 PC, NPC; /* 程序计数器 */
uint32 ASR[16]; /* 辅助状态寄存器 */
/* FPU 控制/状态寄存器堆 */
uint32 FR[32] /* 浮点寄存器 */
uint32 FSR; /* 浮点状态寄存器 */
```

SPARC V8 处理器的指令集中的指令均为 32 位定长指令,可分为三种指令格式,根据各指令中的 op、op2、op3 字段对指令进行译码,如图 4 所示。

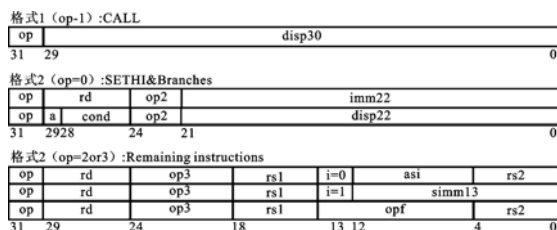


图4 SPARC V8 处理器的指令格式

3.2.2 SPARC V8 事务级模型设计

SPARC V8 事务级模型的结构如图 5 所示,为了提高指令执行过程的可控制性和可调试性,主进程使用 SystemC 中的线程进程 SC_THREAD 实现,并采用指令解释—执行技术模拟目标指令集架构处理器的指令执行过程:

(1)复位信号处于低电平时,整个模型处于复位状态,对模型的寄存器堆进行初始化,并将目标程序(包含 .text 段、.data 段、.bss 段、.stack 段等)初始化到模型附属的虚拟存储器中,复位状态结束后的下一个时钟上升沿将激活模拟器主进程;

(2)调用方法 fetch()根据当前的 PC 值从虚拟存储器中取指令;

(3)调用方法 decode()对指令进行译码;

(4)调用方法 execution()执行指令;

(5)调用方法 wait()将模拟器主进程挂起,下一个时钟上升沿将重新激活模拟器主进程;

(6)重复(2)~(5)步直到目标程序执行完毕。

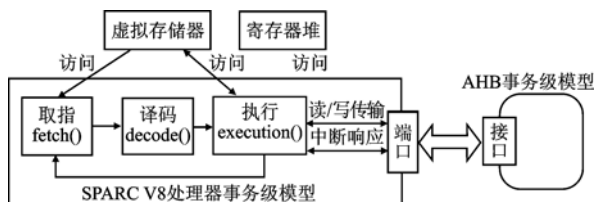


图5 SPARC V8 事务级模型结构

模型在上述指令执行过程的第(4)步时,根据执行指令的不同,分为以下几种情况:

(1)访问寄存器堆;

(2)执行 load/store 类型的指令时,若目的地址位于虚拟存储器的地址空间内,则访问虚拟存储器;

(3)执行 load/store 类型的指令时,若目的地址位于 AMBA 总线从设备地址空间内,则通过端口调用 AMBA 总线事务级模型中的方法对从设备发起总线传输请求,从设备对请求做出应答并完成总线传输。

(4)若有外部中断产生,则捕获该中断,并对中

断进行响应。

由于程序执行时,相邻指令间可能存在数据相关,为保证程序执行的正确性,总线传输采用阻塞方式,即在从设备完成总线传输之前,将模型的主进程挂起,待总线传输完成之后重新激活。

3.3 基于 SystemC 的总线从设备建模

总线从设备的事务级模型的结构如图 6 所示,AMBA 总线事务级模型可调用从设备接口定义的方法访问成员变量,成员变量可作为从设备的控制/状态寄存器,进程每个时钟周期根据控制寄存器的值完成相应操作,并更新状态寄存器的值。

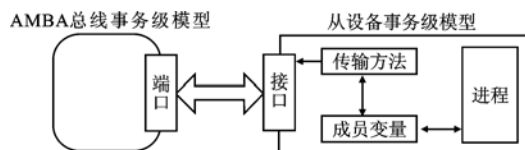


图6 总线从设备事务级模型结构

中断控制器、UART、定时器等均为 APB 从设备,本文参考 Gaisler 公司的 grlib 中对相关 IP 的描述进行实现^[7].中断控制器监视 SoC 中各从设备是否有中断产生,将具有最高优先级的中断传递给处理器,处理器捕获中断后,对中断控制器进行应答。UART 是一种串行通信接口,广泛应用于 SoC 和芯片外围设备之间的数据交换,定时器通过对系统时钟分频的方式进行计时,并根据需求产生中断。

4 验证方法和结果分析

Mibench 是开源的嵌入式测试基准程序组^[8],每个测试基准程序包含各类整型运算指令、读指令、写指令、非条件跳转指令、条件跳转指令和浮点运算指令。使用 Mibench 中的 basicmath、bitcount 和 qsort 对事务级 SoC 验证平台进行验证。

使用交叉编译器 sparc-elf-4.4.2 对上述程序的代码进行编译、链接,生成目标二进制文件。将该文件内容作为验证激励初始化事务级 SoC 验证平台中的虚拟存储器,使用 Mentor 公司的 Modelsim 10.1a 进行仿真(宿主机配置为酷睿 CPU 3.06 GHz、4 GB 内存、Windows 操作系统,平台本身使用 gcc-4.2.1-mingw32vc9 编译)。将相同架构的 RTL SoC 验证平台作为黄金模型,在同样仿真环境下使用相同的验证激励完成仿真,并将两次仿真得到的程序运行结果进行对比,验证事务级 SoC 验证

平台功能的正确性.

仿真完成后统计仿真时间和仿真期间指令处理器执行的指令数量,计算验证平台每秒执行的指令数量,其结果如表 1 所示.

表 1 验证平台仿真速度对比

测试基准程序	RTL SoC(指令数/s)	事务级 SoC(指令数/s)
basicmath	3 135.50	1 049 163.22
bitcount	2 972.28	1 051 616.17
qsort	1 391.46	1 044 526.37

根据仿真结果,事务级 SoC 验证平台针对各测试基准程序的仿真在保证结果正确的前提下,相对于 RTL SoC 验证平台而言,仿真速度分别提高 334.61 倍、353.81 倍和 750.67 倍.

5 结束语

本文采用事务级建模方法设计了基于 SPARC V8 的 SoC 验证平台,其中 SPARC V8 事务级模型采用指令精确建模策略,AMBA 总线、中断控制器、UART、定时器等的事务级模型使用 SystemC 提供的分层通道机制实现,各外设间通过端口、接口和通道实现通信.使用 Mibench 对验证平台的功能和仿真速度进行了验证,实验结果表明,本设计与传统 RTL SoC 验证平台相比在仿真速度上有很大的提高,适用于系统级仿真验证和嵌入式软件的开发,能够有效缩短片上系统 SoC 的设计时间.

参考文献:

[1] Pasricha S, Dutt N, Ben-Romdhane M. Using TLM for exploring bus-based SoC communication architec-

tures[C] //Application Specific Systems, Architecture Processors. Irvine;IEEE,2005:79-85.

[2] 周海洋,葛宁,于立新,等.基于 TLM2.0 的 SPARC 事务级建模[J].计算机工程,2011,37(14):248-250.

[3] Caldari M, Conti M, Coppola M, et al. Transaction-level models for AMBA bus architecture using systemC2.0[C] //Design, Automation and Test in Europe Conference and Exhibition. Munich;IEEE,2003:26-31.

[4] Man K L. An overview of systemC[C]//Research in Microelectronics and Electronics. Eindhoven;IEEE,2005:145-148.

[5] AMBA Specification (Rev 2.0)[M]. [s.l.]: ARM Inc. 1999.

[6] The SPARC architecture manual (version 8)[M]. [s.l.]:SPARC International Inc,1992.

[7] GRLIB IP Core User's Manual (Version 1.2.0 - B4121)[M]. [s.l.]:Aeroflex Gaisler Inc,2012.

[8] Guthaus M R, Ringenberg J S, Ernst D, et al. Mibench: A free, commercially representative embedded benchmark suite [C]//Workload Characterization. Ann Arbor;IEEE,2001:3-14.

作者简介:

杨奕男,(1989—),硕士.研究方向为先进集成系统与 SoC.

付方发男,(1984—),博士,讲师.研究方向为片上网络体系结构研究.

王晗宇女,(1990—),硕士.研究方向为数字下变频 IP 设计.

来逢昌男,(1961—),副教授.研究方向为超大规模集成电路设计、深亚微米芯片物理实现技术.