

基于 SC 的多核处理器并行仿真机制的研究

李德明¹, 全盛程², 叶 进¹

(1 桂林电子科技大学 信息与通信学院, 广西 桂林 541004;

2 桂林电子科技大学 计算机科学与工程学院, 广西 桂林 541004)

摘 要: 针对 SystemC(SC)原有串行仿真内核无法充分利用多核处理器的处理能力问题,提出了一种基本 SC 的多核处理器并行仿真方案.新方案充分利用多线程操作系统及线程池技术的并行处理能力,通过改进 SC 原有串行内核的线程调度方式,对其底层仿真过程进行改进,使改进后的 SC 能够更好地利用多核处理器的处理能力加速仿真模拟过程.此外,新方案还对原有 SC 仿真过程及框架进行了分层处理,从而简化了仿真系统内部的模块相互之间的连接及其数据传输,缩短了仿真系统的建模及处理时间,大幅提高系统的仿真效率.

关键词: 并行仿真; SystemC; 仿真框架; 线程池; 多核处理器

中图分类号: TP393

文献标识码: A

文章编号: 1000-7180(2014)02-0023-04

Research of SC-based Multi-core Processors Parallel Simulation Mechanism

LI De-ming¹, QUAN Sheng-cheng², YIE Jin¹

(1 College of Information and Communication, Guilin University of Electronic Technology, Guilin 541004, China;

2 College of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin 541004, China)

Abstract: SystemC (SC) original serial simulation kernel can not take full advantage of multi-core processors, the paper put forwards a SC-based multi-core processors parallel simulation method. It make use of the parallel processing capabilities of multi-threaded OS & thread pool technology, improved SC original serial kernel, and also hierarchical processing the original SC simulation process. The new framework can take advantage of the processing capacity of multi-core CPU and simplified the connection and data transmission between system's modules, shorten time of system modeling and processing, improved the efficiency of the system simulation.

Key words: parallel simulation; SystemC; simulation framework; thread pool; multi-core processors

1 引言

随着芯片设计及制造技术的飞速发展,芯片的结构及处理过程越来越复杂,其成本也随之急剧提高,设计者为提高芯片的设计成功率,往往需要对其进行建模仿真,并在仿真结果之上对其进行验证并改进,因此现代仿真技术也成为计算机应用的一个新的热点.

SystemC(SC)作为目前最常用的硬件仿真建模工具,可使设计人员实现寄存器传输级(RTL)到事

务处理级(TLM)的不同抽象级别的系统建模仿真.但随着近年的多核 CPU 技术的发展,SC 原有的串行仿真内核已无法充足利用 CPU 的处理能力.本文针对现代硬件仿真技术的特点及需求^[1-4],通过改进 SC 原来的串行仿真机制,提出了一种基于 SC 的并行内核的仿真方案,从而,提高仿真系统在多核处理器平台的仿真速度并简化系统的建模过程.

2 基于 SC 的并行仿真框架

作为片上系统描述的语言库, SystemC^[5] 允许

收稿日期: 2013-04-18; 修回日期: 2013-06-10

基金项目: 国家自然科学基金项目(61163060)

定义可执行的硬件虚拟平台原型,且已成为工业上的标准,但其串行模拟内核的固有特性已无法充分利用现有主流对多处理器(Symmetric Multi-Processor, SMP)的计算能力,针对此问题,项目组分别对多种并行化内核进行深入研究及 SC 原有事务处理机制进一步研究后,提出了一种基于并行处理内核的仿真框架方案,使新框架的仿真性能和开发效率上都有较大提升。

改进后的系统仿真框架结构如图 1 所示,框架分为硬件层、操作系统内核层、SC 并行内核层及事务处理层。新框架充分利用现有多核处理器的硬件性能以及结合恰当的层次化设计,在硬件层和操作系统内核层通过将线程直接与相应的 CUP 核心进行直接绑定,并实现各线程真正意义上的并行处理;在 SC 并行内核层实现任务调度功能,即通过调度器将相应的系统线程与所需执行的逻辑进程进行二次绑定,以确保系统的各线程高效地执行相应的逻辑进程;在事务处理层中隐藏了 SC 原有过于复杂的 TLM2.0 内存管理及其通信机制。这样可使新框架在提升系统仿真处理速度的同时,也大大简化了仿真建模过程。新方案的另一特点是将 SC 系统的主要改动限制在底层内核算法中,即 SC 并行内核层,以使改进后的仿真框架最大程度的兼容原有的 SC 仿真系统,并确保系统稳定性的同时,提高了框架的仿真效率。

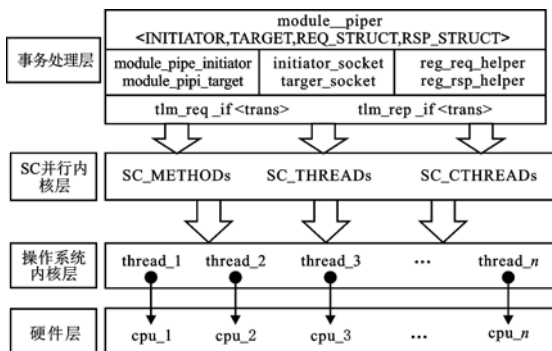


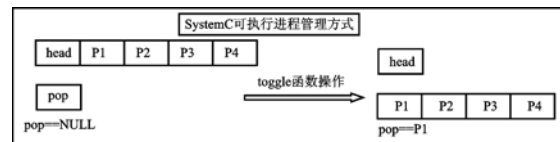
图 1 并行方案的框架结构

3 关键技术

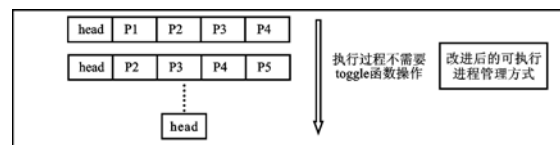
3.1 进程管理

SC 内部的可执行的方法进程和线程进程都是通过链表串联起来以方便插入和删除操作,在实际操作中以队列的方式使用进行头尾的插入删除操作。SC 原有的进程管理方式如图 2(a)所示。其过程是通过 sc_runnable 类的两个进程句柄 head 和 pop 对可执行进程进行管理,内核是通过提取以 pop 为

首的链表数据执行,进程执行过程中以时间到达触发的事件所关联的可执行进程加入到以 head 为首的队列中,当所有进程执行完后(即 pop 为 NULL 时),进行一个 toggle 的操作使得 pop 重新指向可执行进程的链表头部,如此重复。为适应并行加速内核的需求对可执行进程的管理方式进行改进,考虑到了多线程的数据同步以及减少栅障操作。改进后的管理模式如图 2b 所示,去掉了 pop 句柄只保留 head 进程句柄,句柄的插入和删除操作只通过一个链表,当有类似 notify() 发生时立即通知事件的时候,可直接将关联句柄加入链表无需任何 toggle 操作,因此内核评估阶段不需要因为 toggle 操作而进行循环同时也减少了并行内核的栅障操作,并在处理多线程数据的同步问题时加入了临界区的保护。



(a) SC 原有进程管理方式



(b) 改进后的进程管理方式

图 2 SC 原有和改进后的进程管理方式

3.2 时钟实现

SC 的原有时钟是通过两个函数进程实现的,函数分别对上升沿和下降沿事件敏感。进程互相触发对方上升沿和下降沿的事件促进整个仿真程序的时钟前进。但在事务级建模时基本不会用到方法进程,因此为了减少不同进程之间的切换,改进后的方法是将两种方法进程改为线程进程。以时钟的下降沿建立过程为例,新方法的实现过程如下:

```
void sc_clock::posedge_action() {
    while (true) {
        m_next_negedge_event.notify_internal( m_negedge_time );
        m_new_val = true; request_update();
        /* 转到别的进程继续执行 */
        simcontext() -> cor_pkg() -> yield(simcontext() -> next_cor());
    }
}

void sc_clock::negedge_action() {
    while (true) {
        m_next_posedge_event.notify_internal( m_
```

posedge_time);

```
m_new_val = false; request_update(); sim-
context() -> cor_pkg() -> yield(simcontext() -
->next_cor());
}}
```

新方案转换为线程进程后内部必须用无穷循环防止进程退出,同时通过 yield() 函数切换到其他的进程继续仿真系统的执行。

3.3 线程—进程映射

对于执行各种进程集合的操作,传统 SC 做法是采用一个线程依次提取进程句柄顺序执行,但当进程数量很大且每个进程的计算复杂度很大时其执行效率会大大降低。此外,单线程的方式处理进程时亦不能充分利用多核处理器的计算能力,浪费了大量资源和时间,因此这种传统方法在多核处理器系统上的仿真性能较差。在面对大规模 SoC 建模时候,传统的仿真系统将会运行缓慢,严重阻碍软件调试和系统验证进度。

为提高仿真效率,新方案采用线程池方式对各种逻辑进程进行调度,将多核 CPU 的每个核心分别与特定的线程进行绑定,当需要执行逻辑进程时,可同时开启池内的多个线程,并发的从链表中提取进程句柄并执行。由主对线程线程池进行管理,并在适时阻塞和释放池内线程,这样可以尽可能充分利用操作系统及多核 CPU 处理能力。

3.4 并行仿真加速算法

并行仿真加速算法如图 3 所示。

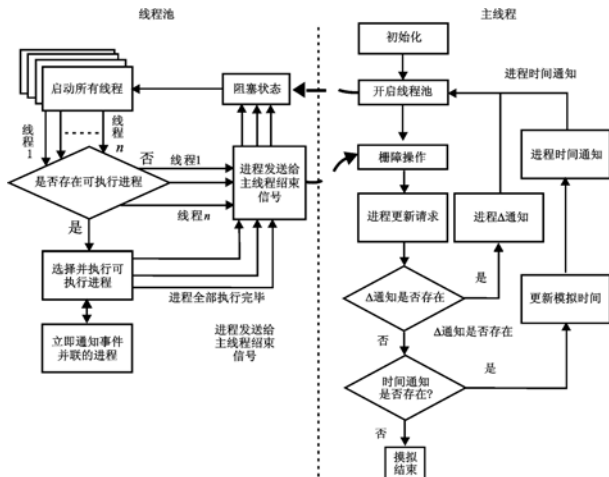


图3 并行仿真内核的调度流程图

步骤 1: 主线程初始化线程池, 创建线程以及和同步有关的事件信号;

步骤 2: 启动所有线程池中被阻塞住的线程并且把线程线程化, 然后进入栅障等待状态, 期间主要

是等待线程池的操作如下;

① 如果没有可执行的进程, 所有线程发送结束信号给主线程进入阻塞状态

② 如果存在可执行进程则依次从链表中提取进程句柄执行, 执行期间立即通知事件所关联的进程可以实时的加入到链表中从而被执行

③ 所有的进程执行完毕, 线程跳到原来的执行点然后发送结束信号进入阻塞状态。

步骤 3 线程池中所有的线程发送完信号后栅障解除, 主线程继续执行;

步骤 4 处理 delta 消息, 调用因为 request_update() 阻塞的 upate() 函数, 将相关可执行进程加入链表重新转到步骤 2 的处理;

步骤 5 如果没有 delta 通知则处理时间通知事件, 如果不存在则整个仿真过程结束;

步骤 6 如果存在时间通知事件, 则提取最小时间戳事件将关联的进程加入链表准备执行, 转入步骤 2。

内核调度器首先创建包含有与运行主机 CPU 核心数相同的线程数量的线程池。并通过利用操作系统底层函数将池中的相应线程与特定的 CPU 核心进行绑定, 在仿真执行过程中, 调度器可按特定的算法将被执行的逻辑进程与这些线程池中的线程进行二次绑定, 这样池中的线程即可通过运行主机的多个 CPU 核心实现真正意义上的并行处理, 同时又可在不增加额外系统线程的情况下, 通过调度器按需要的方式灵活地调度并执行相应的逻辑进程, 从而使系统更有效地加速仿真处理过程。在调度过程中, 并行内核使得 Δ 通知事件和时间通知事件是并行同步压入堆栈, 从而增加了进程执行顺序的不确定性。对此主线程对各个逻辑进程所产生的各种事件和数据进行实时更新维护并产生相应的事件通知队列, 并行处理中的各个线程可通过访问和修改这些队列中的系统状态信息, 实现并行线程间的互斥与同步, 以确保所需处理数据的一致性^[6]。

4 并行仿真方案验证

为验证并行仿真方案与 SC 原有串行仿真方案的相关仿真技术特性, 实验硬件平台分别选用双核 PC 机及 8 核和 16 核 CPU 的服务器进行测试; 软件环境均为 windows Server2007 操作系统和 Visual studio 2005 编译器。

在未考虑其他外部设备及网络等影响因子时, 实验结果如图 4、图 5 中所示, 从图 4 中可以看出:

在模拟的周期数一定的时,随着模拟线程数量的增加时,原有串行方案的运行时间也随之增加而迅速上升,其整个仿真过程对 CPU 核心数基本不敏感,但改进后的并行方案则随着 CPU 的核心数的增加其仿真时间基本成线性比例降低.图 5 表明在仿真线程数量相同的情况下,随着仿真周期数增加时,原有串行方案执行时间增幅远超过并行方案.综合以上实验结果可以看出改进后的并行仿真方案在多核 CPU 环境下,处理多线程仿真任务时,均比 SC 原有串行内核仿真速度及效率均有大幅提升(与 CPU 核心数呈线性相关),从而表明改进后仿真方案能够更好地利用多核处理器的处理能力来加速仿真模拟,此外,通过改进后的分层的仿真框架简化了仿真系统内部的模块连接和其间的数据传输,因此缩短了仿真系统的建模时间,从而大大提高了系统的仿真效率.

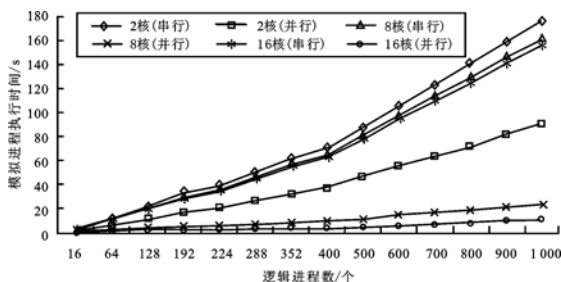


图 4 串/并行内核的执行时间

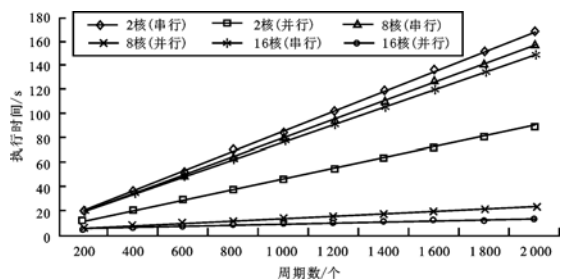


图 5 周期变化下的执行速度

5 结束语

作为常用仿真系统开发工具 SystemC(SC)^[7]提供了一个事件驱动的模拟核,使得系统的设计者能够通过使用 C++ 方便地模拟并行进程,但其原有串行仿真内核的仿真机制无法充分利用多核处理器平台强大的处理能力.在不影响 SC 原有事务处

理等机制的前提下,基于 SC 并行仿真内核方案的多层系统仿真框架通过应用线程池技术以及操作系统线程并行执行机制改进 SC 原有调度策略,对 SC 内部逻辑进程进行调度,并将其分别与多核处理器的各个处理单元进行绑定,从而实现并行仿真处理能力.实验结果表明,在多核处理器的仿真平台上,新方案可充分发挥其并行仿真处理能力,缩短了系统的仿真时间,从而提高了系统仿真的效率,并保持了对原有 SC 仿真系统的向下兼容的能力.

参考文献:

- [1] Daniel Gracia Pérez, Gilles Mouchard, Olivier Temam. FastSysC: a fast systemC engine [C]// Design, Automation & Test in Europe Conference & Exhibition (DATE). Paris, 2004.
- [2] Rafik S Guindi, Youssef N Naguib. SplitPro: a tool to overcome systemC scheduling inefficiencies [C]// Microelectronics (ICM), 2010 International Conference. Cairo, 2010: 347-350.
- [3] Nanjundappa M, Patel H D, Jose B A, et al. SCGP-Sim: a fast systemC simulator on GPUs [C]// Design Automation Conference. Taipei, 2010: 149-154.
- [4] Mello A, Maia I, Greiner A, et al. Parallel simulation of systemC TLM 2.0 compliant MPSoC on SMP workstations [C]// Design, Automation & Test in Europe Conference & Exhibition (DATE). Dresden, 2010: 606-609.
- [5] 唐玲明. 基于 SystemC 仿真器对纯功能型支持技术的研究 [D]. 保定: 华北电力大学, 2008.
- [6] 张颖星, 姚益平. 乐观策略下并行离散事件仿真动态负载划分优化算法 [J]. 计算机学报, 2010, 33(5): 813-821.
- [7] 奚杰, 陈杰, 朱玥. 利用 SystemC 实现多核系统的快速建模 [J]. 微电子学与计算机, 2007, 27(7): 214-217.

作者简介:

李德明 男, (1973—), 硕士, 讲师. 研究方向为自动化测试及网络仿真及应用等.

全盛程 男, (1987—), 硕士研究生. 研究方向为计算机网络及仿真.

叶进 女, (1971—), 博士, 教授. 研究方向为网络协议、嵌入式系统及网络应用等.