LSE:一种处理器体系结构软件仿真器开发工具

喻之斌　金　海
(华中科技大学计算机学院　武汉 430074)

摘　要　在现代处理器体系结构设计中,利用软件仿真技术对设计结果进行验证是最重要的方面之一。然而,处理器体系结构仿真器的开发是一个非常困难的过程。主要的困难表现在三个方面:第一,目前用于处理器体系结构仿真器开发的编程语言如C或C++语言都是串行执行的语言,而处理器的各部件是可以并行运行的,使用串行编程语言编程来模拟并行执行的部件需要长时间的、仔细的程序功能与部件功能的匹配工作,并且容易出错;第二,使用串行程序来模拟并行部件的运行,模拟速度很低,并且仿真速度低是处理器体系结构软件仿真器开发领域的瓶颈问题;最后,仿真器仿真结果的可信度低也是一个关键问题。本文首先介绍了一种新的处理器体系结构软件仿真器开发工具,然后深入分析了该开发工具的优点和缺点,最后对该仿真器开发环境提出了改进方案。

关键词　处理器,体系结构,仿真技术,LSE

LSE: A Development Tool for Computer Architecture Simulator

YU Zhi-Bin　Jin-Hai
(School of Computer Science, Huazhong University of Science and Technology, Wuhan 430074)

Abstract　Software simulation is one of the most important aspects in modern processor architecture design, which is used to verify design results. However, it is very difficult to develop a processor architecture simulator. Three factors contribute to this difficulty. Firstly, the programming languages such as C or C++ used for developing processor architecture simulators are sequential while the components of a processor can run concurrently. The procedure mapping the sequential program to concurrently running components is time-consuming, difficult and error prone. Secondly, the simulation speed of simulators which are developed by sequential programming languages is very low and this is the bottle neck in processor architecture simulation field. Lastly, the high error ratio of the results of a simulator is also a key issue. In this paper, we firstly introduced a new development tool for computer architecture simulators. Then, the advantages and disadvantages of this tool are deeply analyzed. In the end, we come up with a proposal to ameliorate the development tool.

Keywords　Processor, Architecture, Simulation technology, LSE

1　引言

随着处理器体系结构复杂程度的不断提高,软件仿真技术在现代处理器体系结构研究和设计中成为越来越重要的一个方面。在工业界,处理器体系结构设计师们使用软件仿真技术来验证他们的设计;在学术界,研究人员使用软件仿真技术来评估新的思想、算法以及新的体系结构。通常,一款新的处理器在不使用软件仿真器的情况下,从其开始设计到最后测试成功一般需要耗费4～6年的时间[1]。使用处理器体系结构软件仿真器,可以极大地缩短设计时间,并大大扩展处理器的设计空间。因此,在计算机系统结构研究领域,人们十分重视体系结构软件仿真技术的研究和软件仿真器的开发。从文[2]中可以看出,体系结构软件仿真器的开发已经经历了相当长的一段时间并取得了一定的进展。然而,现代计算机体系结构变得越来越复杂,特别是多核处理器技术的出现,软件仿真器的开发越来越困难。一方面,人们对处理器性能和功能要求越来越高,处理器厂商必须更快地制造出更新的、更多不同种类的处理器来满足人们的要求。另一方面,利用软件仿真器来验证更新、更多不同种类的处理器设计是一项非常费时费力的工作。更糟糕的是,在现有的技术条件下,不仅软件仿真器本身的开发非常困难,而且已有的软件仿真器存在着多方面的缺陷。主要表现在以下几个方面:

1)软件仿真器的开发周期非常长;

2)软件仿真器的仿真速度非常慢;

3)仿真结果的错误率较高,不能正确地指导处理器的设计。

本文首先介绍了一种新的处理器体系结构软件仿真器开发方法和软件仿真器开发工具。然后对该方法和工具的特点、优点和缺点进行了分析,最后对该开发工具的进一步完善提出了建议方案。

2　LSE介绍

LSE的全称是Liberty Simulation Environment,它是由普林斯顿大学计算机系开发的一套用于处理器建模或仿真的开发工具。该开发环境旨在对处理器的并行结构化部件进行建模,然后自动生成处理器仿真器。在仿真器的开发过程中

喻之斌　博士研究生,主要从事多核处理器体系结构、体系结构软件仿真技术研究;金　海　教授,博士生导师,主要从事计算机体系结构、并行分布式处理、集群与网格计算等方面的研究。

最大限度地重用已有的组件，从而降低仿真器开发的成本和难度。利用 LSE 的仿真器开发过程如图 1[3] 所示。

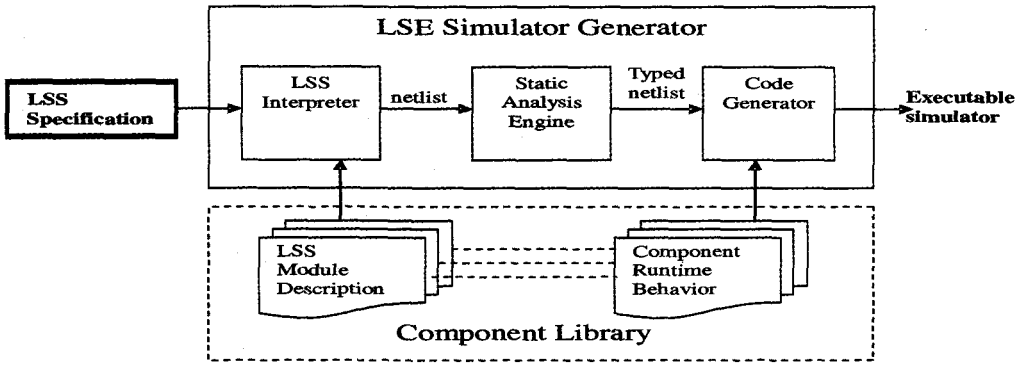

图 1　利用 LSE 的仿真器生成过程

在图 1 中，利用 LSE 开发处理器体系结构软件仿真器的第一步是使用 LSS 语言对拟仿真的处理器结构进行描述。LSS(Liberty Structural Specification Language)是 LSE 中定义的一种硬件描述语言，主要对处理器中的并行结构化部件，如处理器中的算术逻辑部件(ALU)、通用寄存器、状态寄存器、高速缓存及地质转换监视缓冲器(TLB)等进行静态描述[4]。该描述以 *.lss 文件保存，一个 lss 文件就是一个某种抽象级别的处理器部件或处理器。获得了 lss 文件以后，需要使用 LSE 提供的编译器对其进行解释。解释的过程中，将调用 LSE 组件库中的模块描述库(LSS Module Description)。LSE 中的模块描述库有两种：一种是 LSE 提供的核心模块库[5]，另一种是仿真器开发者利用 LSE 提供的模块定义机制自己开发的模块库。经过 LSE 编译器解释的 LSS 文件只刻画了处理器各部件纯粹意义上的连接，各连接上的约束还没有加上，也还没有对这些约束进行合法性、一致性检查。特别地，还没对同一连接的两个部件端口上数据类型的一致性进行检查。因此，下一步就是利用 LSE 提供的静态结构分析引擎(Static Analysis Engine)给连接加上约束并检查这些连接和约束的合法性、一致性。如果 lss 文件成功通过了静态结构分析引擎的分析，就得到了具有正确连接约束的处理器部件或处理器模型，并被称为具有连接类型约束的模型。最后，LSE 代码生成器(Code Generator)通过调用组件运行时行为库(Component Runtime Behavior)生成可执行的仿真器。

从图 1 中可以看出，LSE 由三部分组成：(1)Liberty 结构化硬件描述语言 LSS；(2)仿真器生成器；(3)核心模块库和运行时库。其中 LSS 是利用 LSE 进行处理器仿真器开发的基础，它不仅是对处理器结构进行描述的工具，而且是仿真器开发人员创建扩展库的工具。核心模块库提供了用于创建处理器模型的基本元素，如选择器、转换器、路由器等。LSE 定义了两种类型的模块：一种称为叶子模块，另一种称为组合模块。叶子模块是一种简单模块，它不能再被拆分为更简单的模块。在申明叶子模块时，需要指出叶子模块的参数、信息接口以及与该模块对应的行为代码文件的位置。图 2 中的代码示例了一个叶子模块的申明。模块的行为代码就是模块的定义，它用一种类似 C 语言的编程语言 BSL(Behavior Specification Language)来实现。组合模块是由已经存在的模块组合而成的模块，参与组合的模块既可以是叶子模块，也可以是组合模块。和叶子模块一样，申明组合模块必须指出其参数、信息接口。与叶子模块不同的是，它不使用 BSL 语言来定义它的行为，而是将内部子模块实例化并将子模块实例连接起来，如图 3 所示。

```
module delay{
    parameter initial_state = 0:int;

    inport in:int;
    outport out:int;

    tar_file = "corelib/delay.tar";

    //BSL 参数
};
```

图 2　LSS 叶子模块申明

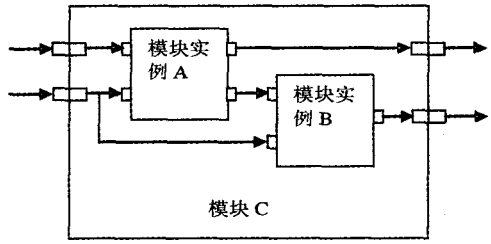

图 3 组合模块示例

3　LSE 的主要特点

LSE 是目前最优秀的处理器体系结构开发工具之一。它的特点主要体现在以下几个方面：(1)提供了并行结构化建模机制；(2)提供了基于模块的重用机制；(3)静态模型分析；(4)LSE 可以作为一个处理器通用仿真器构建框架。

3.1　并行结构化建模

通常并行结构化系统是由可并行执行的部件组成的。每个部件都有一个或多个输入及输出端口，部件之间通过端口互相连接，构成一个网状模型。运行时，部件间通过在预定义的网状模型上发送与接收数据进行通信。计算机处理器系统是一个同步数字设备，其状态与一个统一的时钟同步，内部部件可以并行执行，因此，计算机处理器系统是一个典型的并行结构化模型。LSE 通过核心模块库提供了数字设备所必需的基本元素，如转换器、选择器等。另外，在 LSE 中还可利用 LSS 语言根据基本部件元素或已经定义的部件定义更复杂的部件。因此，LSE 可以直接对处理器系统进行并行结构化建模，并使它们并行执行。如每个处理器部件对应一个 LSE 模块实例。当然，根据抽象级别的不同，一个 LSE 模块实例也可以对应一组处理器部件，但 LSE 模块实例依然是可并行执行的。

3.2 基于模块的重用

LSE提供了和面向对象方法中的类及软件工程中的组件类似的重用机制。LSE的重用机制主要体现在以下几个方面:(1)参数化部件;(2)多态;(3)静态模型分析;(4)数据收集。

3.2.1 参数化部件

参数化部件是指LSE允许通过参数的形式对模块实例的属性进行定制。例如,高速缓存部件的置换策略可以通过参数在一个预先定义的枚举中选择。LSE的参数化模块定制分为两种:结构化定制和算法定制。结构化定制通过参数来规定一个组合部件实例中参与运行的子部件的实例。即组合部件的实例结构和该组合部件在定义时定义的静态结构可能不一样,但组合部件实例结构中的子部件集合是其静态结构中子部件集合的子集。算法定制使用参数化的方式来继承和增强已有部件的功能和行为。例如,我们可以通过参数来定制总线仲裁器的内部仲裁逻辑。

3.2.2 多态

为了更好地支持模块重用(实际上是仿真器开发中的代码重用),LSE支持两种形式的多态:参数化多态和组件重载。参数化多态用于创建和使用独立于数据类型的组件模型。这里虽然使用了多态这个词,但它却与面向对象系统中的多态不一样。LSE的参数化多态更像C++语言中的模板。组件重载是指LSE根据数据类型的不同而自动选择同一模块声明的不同模块实现。面向对象系统中,函数名重载根据不同的参数类型来选择不同的函数实现。而在LSE中,根据模块的端口和连接类型选择不同的模块实现。

3.3 静态模型分析

静态模型分析是为仿真器的优化和仿真器开发者使用方便而提供的一种机制,主要用于具有多态端口的具体数据类型推理。即仿真器开发者在构建处理器模型时,某些部件端口的数据类型可能一时难以确定,这时可以使用LSE提供的通用数据类型,在处理器模型开发完成以后,启动静态模型分析功能,LSE便可根据模型中的部分端口的具体数据类型推理出所有端口的具体数据类型。

3.4 处理器通用仿真构建框架

LSE是一种通用的处理器仿真器开发工具,它可以将多种不同的仿真器、处理器仿真部件、仿真器开发技术以及指令集仿真器集成在一起。首先,LSE本身提供了一套核心部件,如仲裁器、选择器等。其次,分支预测、高速缓存等的仿真可以作为独立的仿真器开发,它们既可独立使用,也可集成在一个处理器系统中使用。另外,不同指令集的模拟器(Emulator)如IA-64、IA-32、PowerPC和MIPS等也可使用LSE作为独立的模拟器开发出来并集成使用。更为重要的是,LSE可以将多种不同厂商开发的处理器仿真器如SimpleScalar[7]、Simics[6]和Smart[8]等集成在一起。最后,在LSE中可以集成多种仿真器加速技术,如采样技术、并行调度技术等。因此,处理器仿真器开发人员利用LSE可以根据自己对目标处理器的需要而裁剪仿真器的功能、性能以及行为。

4 LSE分析

快速开发一个仿真结果可靠、运行速度快、配置灵活和修改方便的处理器仿真器,一直是计算机系统结构设计师的理想之一。

LSE最主要的贡献是通过它提供的一整套开发工具和开发机制,使快速开发、灵活配置和方便修改的处理器仿真器成为了可能。学术界使用得最多的仿真器SimpleScalar耗费了其作者两年半的时间开发其第一个版本[7]。使用LSE开发一个与SimpleScalar相同功能的仿真器却只要几个月的时间。应该说LSE成功解决了处理器体系结构软件仿真领域三个主要困难中的第一个困难,即大大缩短了仿真器本身的开发周期。LSE的这个成功主要来源于它采用了并行结构化的建模方法。传统的处理器仿真器开发主要使用C或C++等编程语言,这种类型的编程语言是串行执行的,对某个处理器部件的仿真通过函数调用的方式来实现。整个仿真器的运行通过main函数开始执行,依次串行地调用各个功能不同的函数,从而实现对一个处理器的仿真。这种串行执行的仿真方式与所要仿真的处理器实际运行的方式存在着显著的差异。因为通常情况下处理器的部件可以并发地运行。仿真器执行方式和处理器实际运行方式的不同带来了以下几个方面的困难:第一,将串行执行的函数的功能准确无误地对应到具有并行运行结构的处理器部件上非常耗费开发人员的精力和时间,并且容易出错。第二,仿真方式违背了仿真的原则:尽可能地还原物理部件的结构和运行。仿真方式不匹配直接的后果就是仿真结果不可靠。第三,串行仿真并行执行的处理器部件,成倍降低了仿真器的执行速度。仿真速度是目前处理器体系结构仿真领域的一个瓶颈问题,特别是在多核处理器的环境下。正是因为观察到了传统仿真器开发的这个缺陷,LSE提出了并行结构化的处理器结构建模方法,利用其提供的结构化硬件描述语言LSS,就可将处理器的结构描述出来,省去了传统开发中需要开发人员将函数映射到处理器部件上的过程,大大提高了仿真器的开发速度,并且仿真器结构可准确无误地反映处理器的结构。使用LSS描述出来的处理器模型就是一个虚拟的处理器,程序可以在上面执行,像真实的处理器一样,虚拟的部件也可以并行运行。

第二,LSE提供的重用机制方便和加速了新仿真器的开发以及对已开发的仿真器的修改,降低了开发成本,可满足人们对更新、更多种不同处理器进行仿真的需要。在面向对象方法中,重用主要是通过类的继承和复合实现的。在类的继承方式中,派生类的实例会包含基类的所有数据成员,而有些基类的数据成员(如私有数据成员)在派生类中是不可使用的,这样就导致了派生类实例的额外内存开销。类的复合也有同样的问题,即复合类的实例不论它使不使用子类,它都得包含子类的实例(如C++)或子类实例的引用(如Java,C#),但无论如何也会有额外的内存开销。LSE提供的参数化部件重用机制避免了这些问题,达到了重用而无额外内存开销的效果。LSE没用采用类的继承机制,而在类的复合机制上做了改进。和类的复合一样,LSE的组合部件在定义时规定了一组子部件以及这组子部件间的连接,我们称之为该组合部件的静态结构,但不同的是这些子部件在实例化时并不一定都参与运行,LSE通过参数在静态结构中选择参与运行的子部件,从而避免了额外的开销。

第三,多态和静态模型分析为仿真器维护,功能扩展提供了方便。LSE的多态类似于C++语言中的类模板机制。当仿真器开发者需要为仿真器增加一种新的数据类型时,只需将这种数据类型作为参数传给相应的模块即可。如LSE中的队列(queue),它并没有规定具体的数据类型,而是可以接受所有的LSE数据类型,包括用户自定义类型。在为仿真器增加新的部件时,需要指出该部件端口的数据类型。LSE允

许用户只使用一个类似 void * 的通用数据类型，然后由 LSE 静态模型分析引擎自动分析部件端口的数据类型。这种机制也会节约复杂处理器仿真器的开发时间。

虽然 LSE 成功解决了处理器仿真领域的第一个问题，即大大缩短了仿真器的开发时间，但它并未成功解决仿真器的速度慢和仿真结果出错率高这两个问题。使用 LSE 开发出来的仿真器的仿真速度甚至比用传统方法开发出的仿真器的速度还慢[9]。即使 Penry 博士使用调度的方法对 LSE 生成的仿真器进行了加速，如单处理器仿真器加速了 2.08 倍、四核处理器仿真加快了 7.56 倍[9]，但仿真速度还是难以满足目前对仿真器的速度要求，特别是多核处理器的仿真速度要求。另外，由于仿真器的执行速度慢，目前的 CPU 测试程序包如 SPEC CPU2006 或 SPEC CPU2000 不可能在处理器仿真器上在人们可以接受的时间内运行完。因此，只能选择部分测试程序指令在仿真器上执行，从而模拟实际处理器的性能[2]。这又带来了另一个问题，即仿真结果出错率高，不能对处理器设计进行指导。

造成 LSE 生成的仿真器慢的原因主要有两个方面：部件重用的副作用和静态分析模型的类型推理。第一，虽然 LSE 通过参数化部件的方法减少了内存的额外开销，但它并没有克服面向对象方法的另一个局限：重用粒度的选择问题。在 LSE 中，重用粒度指组件模块的大小。组件模块粒度过大，一方面仿真器不能详细刻画仿真对象的结构、行为及特征，导致仿真结果不可靠，另一方面导致简单功能的额外开销。因为某些部件非常简单，但没有粒度更细的仿真部件使用，从而造成不必要的资源浪费。如果组件模块粒度过小，又会引起仿真器结构中连线过于复杂、仿真部件间信号量大的问题。大量信号的处理将直接降低仿真器的运行速度。另外，LSE 的组件模块粒度也影响着仿真器的抽象级别问题。在处理器体系结构仿真领域，仿真抽象级别也是一个非常难于决定的问题。仿真速度和仿真结果的正确性这一对矛盾也在这个问题中得到了体现。抽象级别高，仿真器结构简单，运行速度快，但只能得到一些简单信息，仿真结果用处不大。抽象级别低，仿真器构造复杂，运行速度慢，可以得到更为详细的仿真信息，但往往在仿真信息有用时需要非常长的仿真时间。第二，LSE 提供的静态分析模型也大大影响了仿真器的运行速度。虽然在仿真器开发时，我们可以利用 LSE 定义它的静态结构，但运行时，LSE 允许根据参数选择参与运行的组件部件。因此，每次实例化参数化的组件部件时，都需要重新推理组件部件的端口的数据类型。而处理器模型性中组件部件端口的数据类型推理是一个 NP 难度问题[1]，所以这个过程也将在很大程度上降低仿真器的执行速度。

5 改进方案

根据以上分析，LSE 的主要问题集中在两个方面：生成的仿真器仿真速度和仿真结果的出错率。因此，在不牺牲仿真结果精度的情况下提高仿真器的执行速度是主要的努力方向。我们提出如下方案，从多方面对 LSE 生成的仿真器进行加速：

1）在 LSE 框架下加入系统采样技术，从处理器测试程序中抽出部分能代表整个程序的指令，加速 LSE 仿真器的执行。将系统采样技术开发成一个独立的模块，使其紧跟在 LSE 仿真器初始化之后运行，进行测试程序动态指令的选择。

2）使用并行技术优化 LSE 仿真器所使用的计算模型 HSR（Heterogeneous Synchronous Reactive）。

3）建立仿真器模块粒度优化模型，对模块的力度进行优化，从而加快仿真器的执行速度。

4）从 LSE 中去除对模块端口的数据类型自动推理。因为在设计处理器部件时，该部件与其他部件通讯的数据类型不难由设计者手动确定下来。

总结 在处理器研究、设计和制造领域，体系结构软件仿真器是一个必不可少的工具。目前主流的仿真器多是采用串行编程语言进行开发，并且其仿真执行也是串行的。因此这种开发模式有三个主要弊端难以回避：仿真器开发周期非常长，开发难度非常大；仿真器的执行速度非常低；仿真结果的可信度低。LSE 提出的并行结构化开发模式及其提供的工具可以有效地解决第一个问题。然而，目前利用 LSE 构建的仿真器的速度和仿真结果可信度依然不理想，难以满足处理器研究、设计和制造领域的需要。通过对 LSE 深入、系统的分析，我们发现 LSE 目前的缺陷并不是并行结构化建模方法的缺陷，LSE 还有很大的改进空间。经过我们的改进，LSE 将是一套更先进的处理器体系结构软件仿真器开发工具。

参 考 文 献

1 Blome J, Vachharajani M, et al. The Liberty Simulation Environment. ASPLOS XI Tutorial, 2004.10

2 喻之斌，金海. 多核处理器体系结构软件仿真技术：研究综述. 计算机科学，2007(10)

3 Vachharajani M, Vachharajani N, et al. The Liberty Simulation Environment: A Deliberate Approach to High-level System Modeling. ACM Transactions on Computer Systems, 2006, 24(3): 211～249

4 Vachharajani M, Vachharajani N, et al. The Liberty Structural Specification Language: A High-level Modeling Language for Component Reuse. In: ACM SIGPLAN 2004 Conference on Programming Language Design and Implementation, 2004

5 The Liberty Research Group. Liberty Simulation Environment Core Module Library Reference Manual, 2003.12

6 Magnusson P, Christensson M. Simics: A Full System Simulation Platform. Computer, 2002, 35(2)

7 Todd M A. A User's and Hacker's Guide to the SimpleScalar Architectural Research Tool Set. Intel MicroComputer Research Labs, 1997

8 Roland E W, Thomas F W, et al. SAMRTS: Accelerating Microarchitecture Simulation via Rigorous Statistical Sampling. In: Proceedings of the 30th Annual International Symposium on Computer Architecture (ISCA'03), 2003

9 Penry D A. The Acceleration of Structural Micro-architectural Simulation via Scheduling: [Ph D dissertation]. The Department of Computer Science. Princeton University, 2006