第一章 绪论

模拟器是体系结构研究和微处理器设计的重要基础支撑平台。本文将构建一

种 RISC 处理器的指令集模拟器，为 RISC 处理器体系结构的研究打下坚实的基础。

本章首先介绍论文的研究背景和意义；接着介绍国内外研究现状；最后介绍

论文的主要研究内容和结构。

1.1 研究背景与研究意义

随着 CPU 设计规模的不断增大，处理器设计正面临着一系列严峻的挑战。同

时由于深亚微米工艺带来的设计困难，CPU 设计的复杂度大大提高。在集成度越

来越高的今天，面对数千万乃至上亿晶体管的规模，那种“设计硬件原型-实现-评

估-改进-再实现”的模式已经无法满足现代设计应用的需求[

1]；而数学模型在 Cache、

分支预测、乱序执行等技术被广泛应用的情况下并不能精确地反映系统的实际情

况[

2]。研究表明，如果在 CPU 的方案论证和设计阶段，没有及时发现问题与瓶颈，

将会使后续的工作变得更为困难且代价更加高昂[

3]。因此，在开发一个新的体系结

构处理器时，为了确保处理器功能特性和性能参数达到设计的预期目标，对体系

结构进行验证是一个必不可少的步骤[

4]。

体系结构模拟器是在宿主机（运行模拟器的机器）上运行并能模拟目标机（需

要模拟运行的机器）行为功能的软件[

5]，因此又称之为软件模拟器。体系结构模拟

器具有运行速度快、模拟精度高、灵活性和可扩展性好的优点，因此受到了体系

结构研究者们的格外青睐。更有甚者，对大部分体系结构初学者来说，软件模拟

器是他们进行体系结构研究的唯一手段。国际上绝大部分的体系结构研究成果都

是基于模拟器平台做出的[

6][7][8][9]，国外各大处理器设计公司也都有自己专用的模

拟器平台[

10][11]，因此，软件模拟器是体系结构研究和微处理器设计的重要基础支

撑平台。

体系结构模拟器根据模拟的详细程度可以分为功能模拟器（指令集模拟器 ISS，

Instruction Set Simulator）和性能模拟器（时钟周期精确模拟器）两类。前者只模

拟目标系统的指令集体系结构，比如寄存器状态、指令语义、存储器状态等功能

特性；后者除了模拟功能特性外，还模拟目标系统的微体系结构，比如流水线、

分支预测、Cache 层次等[

13]。功能模拟器能够验证目标系统的正确性，辅助体系结

构前期方案的确定；性能模拟器能得到目标系统的性能参数，辅助体系结构设计

方案评估。

体系结构模拟器的一个重要指标是模拟速度。体系结构研究时有两个重要需求，第一是设计方案的快速评估，以提高设计空间的探索速度和“设计-评估-修正”

周期的迭代速度。第二是使用多种类型的真实应用作为测试程序，以全面准确地

评估体系结构设计在各种不同应用场景下的性能。这两种需求都要求模拟器具有

非常高的性能，以在受限的时间预算内找到最佳的设计方案。然而，虽然经过了

多年的发展，现有模拟器的模拟器速度仍然非常低，因此，提高模拟速度是模拟

器开发的重中之重。

Gem5[

12]是一个多核全系统模拟器平台，由著名的多核模拟器 GEMS 和 M5 合

并而来，其集成了多种指令集体系结构（包括 x86、ARM、MIPS、ALPHA 和 Power）

和 CPU 模型，并支持从微体系结构（Micro-architecture）到系统级（System-level）

多种结构的模拟。Gem5 继承了 GEMS 的优点，能够细致而灵活的模拟存储系统

结构，包括支持多种不同的 Cache 一致性协议和互连模型。由于它采用面向对象

的程序语言（C++和 Python）编写，因此具有良好的模块化结构，结构非常灵活。

目前项目组正在进行一种 RISC 指令集处理器的研制，尚没有可用的指令集模

拟器进行原型验证和功能验证，因此设计一款目标处理器的指令集模拟器对项目

组后续工作的展开具有重要意义。

1.2 国内外研究现状

到目前为止，体系结构模拟器的研究与开发已经有了几十年的历史，开发者

们基于不同的目的开发出了各种各样的模拟器，包括 Liberty、SimpleScalar、SimOS、

SimOS-Goodson、GEMS、SS\_SMT、ARMulator、Bochs、SkyEye、Gem5、ACSim

等。

LSE（Liberty Simulation Environment）是由普林斯顿大学开发的用于计算机系

统建模的集成开发环境[

15]，该开发环境主要用来对计算机系统的并行结构化部件

进行建模，然后自动生成模拟器，在模拟器的开发过程中可以最大限度地重用已

有的组件，从而达到降低开发成本和难度的目的。利用 LSE 开发模拟器的过程如

图 1.1 所示。

SimpleScalar 工具集最初是由 Wisconsin 大学的 Todd Austin 在其攻读博士期间

开发出来的[

16]。由于其开源且具有良好的可移植性和可扩展性，能满足研究者们

对模拟器的不同需求，曾一度成为业界主流的模拟器。SimpleScalar 工具集还提供

了详细的开发技术文档，并提供构造模拟器所需的全部资源：编译器、汇编器、

链接器、模拟器和调试器等，使得在其基础上开发处理器模拟器较为简单。

SimpleScalar 工具集是一系列的模拟器的集合，其体系结构如图 1.2 所示，它

包含有两个功能模拟器 Sim-fast 和 Sim-safe 以及一个性能模拟器 Sim-outorder[

16]。

这些模拟器模拟的侧重点各有不同，用户可以根据自身的要求来进行选择。Sim-fast

速度最快，但模拟精度很低，它不对指令进行错误检查，没有计时机制，也没有

引入指令级的并行；Sim-safe 在 Sim-fast 的基础上，对指令是否对齐、存储器访问

是否符合相应的权限等进行检查，在考虑模拟速度的前提下提高了模拟精度。

SimpleScalar 工具集中模拟精度最高的是 Sim-outorder，它是一款超标量模拟器，

支持乱序发射和执行，并且可以对程序执行期间流水线的状态进行记录。

由斯坦福大学 DASH/FLASH 研究小组于 1992 年开发的 SimOS 是发布最早的

多处理器全系统模拟器[

14]，它不仅可以对运行在某种体系结构上的操作系统进行

研究，也可以对某种体系结构本身进行研究，其总体结构如图 1.3 所示。SimOS

最早应用于对基于 MIPS 指令集的多核处理器进行建模，它实现了对包括网卡（NIC，

Net Integer Circuit）、RAM、Disk、终端控制（Console）、CPU、MMU 和存储系

统等在内的计算机系统的各个部件的模拟，并采用顺序单发射的 CPU 模型，可以运行完整的操作系统和各类应用程序。由于其支持全系统模拟，可以运行大量的

系统服务应用程序，为工业界和学术界广泛采用。但由于 SimOS 采用执行驱动方

式，宿主机的硬件结构对目标机的模拟具有一定的影响，因此造成可移植性上的

缺陷。

基于龙芯 CPU 的多核全系统模拟器 SimOS-Goodson 是中国科学院计算技术研

究所研制开发的基于龙芯 CPU 的执行驱动模拟器[

17]，其结构如图 1.4 所示，它对

计算机系统中包括 CPU、存储系统、硬盘、网卡、终端控制等在内的所有组件进

行了建模，使得在其上可以运行完整的操作系统，从而使体系结构研究者可以在

真实的操作系统环境下模拟目标处理器对各类应用程序的执行。

SPEC 标准测试集的测试结果表明，SimOS-Goodson 具有较高的模拟精度，其

与真实硬件系统的误差控制在 15%以内。在具有较高模拟精度的同时，

SimOS-Goodson 的模拟速度也较为可观，进行全系统模拟时，其 IPS（Instruction Per

Second）达到了 300K，仅比应用程序级的模拟速度慢约 30%[

17]。

GEMS （General Execution Model Simulator）是 Wisconsin 大学在商业模拟器

Simics 的基础上开发出来的全系统模拟器，它通过加载时序模型来驱动功能模拟

器 Simics 的执行[

18]。它能够细致而灵活的模拟存储系统的结构，包括支持多种

Cache 一致性协议和互连模型。GEMS 的主要优点在于它的灵活和简洁，用户可以

专注于时序模型的设计，而全系统环境下复杂的功能模拟则交由商业模拟器 Simics

来实现，但也正是由于功能模型受时序驱动，每条指令提交时都必须在时序模型

和功能模型之间校验结果，使得时序模拟和功能模拟的部分工作重叠冗余，从而

影响模拟执行的速度。

SS\_SMT[

19]是基于 SimpleScalar 开发的同时多线程（Same Multiple Thread，

SMT）模拟器，其定义的存储层次很复杂，在 Cache 中包括了对 Module 和 Bank

结构的模拟。SS\_SMT 中增加了可配置的译码宽度（每个时钟周期从指令队列中取

出的指令数目）。但是，由于 SS\_SMT 采用一种特殊的 Cache 模型，这种 Cache

模型无法客观地实现对典型超标量体系结构的建模，使得其模拟精度较低；而且

其过多的继承了 SimpleScalar 解释型模拟器的结构，需要对指令进行逐条翻译，所

以其模拟速度也不高。

ARMulator[

20]是 ARM 公司推出的集成开发环境 ADS（ARM Developer Suite）

中提供的指令集模拟器，它模拟的是 ARM 微处理器体系结构和指令集，与运行在

宿主机（通常是 x86 体系结构）上的调试器相连接，提供了开发和调试 ARM 程序

的软件模拟环境。ARMulator 不仅可以模拟 ARM 处理器的指令集体系结构和微体

系结构，而且对存储器和处理器外围设备进行建模，以实现对全系统的支持。

Bochs[

21]是一种开源且高度可移植的全系统模拟器，它完全模拟了 x86 平台的

所有硬件，包括 CPU、Disk、Memory、声卡和其它的周边设备。Bochs 的典型应

用在于提供完整的 x86 体系结构模拟，它目前还支持对 AMD64 CPU 的模拟，在

它上面可以运行 Linux 操作系统。由于它模拟了所有的硬件设备，因此可移植性很

高；但也正是因为它模拟了所有的硬件设备，使得其运行速度非常缓慢。

SkyEye[

22]（天目）是国内一个开源的自由软件项目，它的目标是在通用的 Linux

和 Windows 平台上实现一个纯软件的、基于 ARM 处理器的嵌入式模拟集成开发

环境，目前 SkyEye 主要用来模拟基于 ARM 内核的处理器。SkyEye 可支持多种指

令集体系结构，可以实现多种嵌入式开发板的模拟。由于 SkyEye 是一个全系统模

拟器，除了模拟 CPU 内核，它还模拟了存储器、存储管理单元、Cache、串口、网卡、时钟等。由于 SkyEye 的目标不是对硬件逻辑进行验证，而是协助系统软件

的开发和调试，所以其模拟精度不是很高。

Gem5 模拟器结合了 M5 和 GEMS 中最优秀的部分，是一款高度可配置、集成

多种 ISA 和多种 CPU 模型的体系结构模拟器[

12]。目前，Gem5 已经能够支持多种

商用 ISA，包括 X86、ARM、ALPHA、MIPS、Power、SPARC 等，并且能够在

X86、ARM、ALPHA 上加载操作系统。Gem5 是目前业界主流的多核模拟器，因

此本文实现的模拟器基于 Gem5 进行设计和开发。

ACSim 是中国科学院计算技术研究所基于 Gem5 开发的 ARMv8 性能模拟器

[23]。该模拟器支持多种处理器执行模式，包括简单的功能级模拟、简单时序模拟、

顺序和乱序执行模拟；支持多种内存子系统的模拟；可以和 Mc PAT 连接以模拟处

理器功耗。

此外，比较著名的研究项目还有法国的嵌入式多体系结构模拟器 Qemu[

24]、上

海同济大学研制的 PEMU[

25]以及浙江大学开发的 WuKong 系统

[26]等。

尽管上述模拟器种类繁多，但大致可以分为两类：针对某种体系结构的专用

模拟器和通用模拟平台。现有专用模拟器的模拟性能很好但由于其只针对特定体

系结构，使得其使用范围单一。通用模拟平台尽管使用广泛，但一般都是建立在

解释型模拟器的基础上的，这就使得其需要在译码阶段耗费大量的时间，从而使

得模拟速度不尽如人意。因此，设计实现一款通用、灵活、可配置而又模拟速度

可观的模拟器对体系结构的研究有着极其重要的作用。

1.3 主要研究内容

本文将设计实现目标处理器的指令集模拟器，主要包括三部分工作：指令模

拟、存储模拟以及模拟器测试。

对于指令模拟，本文将在 Gem5 原有的指令集基础上进行扩展，实现目标处

理器指令集，包括 Load/Store 指令、数据运算指令、分支指令、异常生成指令以

及系统指令。并在原有的解释型指令集模拟器的基础上，对其指令处理流程进行

改进，使其在保持灵活性的同时提升模拟速度，以满足人们对高速模拟的需求。

对于存储模拟，本文采用地址映射的方式将目标机的地址空间映射到宿主机

的内存空间，从而以一个连续的虚存空间来实现对目标机存储器的建模。为了提

高地址转换效率，本文拟设计实现一个全局软 TLB 表来提升虚实地址的转换效率，

以提升模拟器的模拟速度。

对于模拟器测试，本文将分为两部分：功能测试和性能测试，功能测试主要

验证模拟器功能的正确性，而性能测试则主评测模拟器执行速度及我们所提优化国防科学技术大学研究生院工程硕士学位论文

第 8 页

第二章 体系结构模拟器理论基础

模拟，Robert E Shannon（信息论创始人）将其定义为设计一个真实系统的模

型并在这个模型上进行试验以理解系统的行为或者在各种限定条件下评估系统操

作的各种策略[

27]。而模拟器是一种用来模拟其它真实硬件系统的工具，它可以分

为两大类：一类称为硬件仿真器（Emulator），一类是软件模拟器（Simulator）。

前者将软硬件相结合，通过硬件来运行程序，可以很好的提高模拟速度；通过软

件来提供一个界面，以供用户实时了解程序的运行情况。后者根据模拟的详细程

度又可以分为指令集模拟器（功能模拟器）和时钟周期精确模拟器（性能模拟器），

这类模拟器以纯软件的形式在一台现有的计算机（称为宿主机）上模拟另一台计

算机（称为目标机）程序的执行过程。与硬件仿真器相比，软件模拟器的优点在于

它不需要目标机以及任何配套硬件的支持。

2.1 模拟技术理论基础

模拟技术理论最早起源于图灵机。英国数学家艾伦·图灵在著名的《论可计算

的数，以及其在判定问题中的应用》[

28]一文中提出了图灵机的概念，奠定了现代

计算机理论的基础。

图灵还定义了模拟的概念：在不考虑硬件和速度的情况下，不同体系结构的

机器 A 和 B，在理论上可以用指令实现互相模拟（即图灵机）[

29]。图灵机概念的

提出为计算机软件模拟技术的发展奠定了基础。

计算机软件模拟技术实质上就是用计算机软件来模拟实际硬件系统的行为，

以获得该系统的行为数据，方便对其进行分析与研究。模拟开发的基本流程如图

2.1 所示。

进行模拟，首先通过对实际硬件系统建模来将之具体化。建模与具体化的过

程中必须保证所建模型的结构与实际硬件系统相近或一致，以确保所建模型的精

确性，只有精确度较高的模型才能真实的模拟出硬件系统的行为，最终获得正确

的结果。在对硬件系统的建模过程中，需要考虑所选择的算法是否合适。评价一

种算法是否合适的准则在于是否符合模拟的要求和硬件系统的特征。为了保证最

终的模拟精度，必须确保所选择算法的精度够高，稳定性够好。选定合适的算法

后，进行程序设计，即用程序语言将模型描述出来。待确定程序模型的正确以后，

就可以用这个模型来进行模拟实验，得到相应的结果。最后分析模拟结果，结果

分析既可以是针对模型本身的数据，对模型本身进行评价或研究；也可以是对模

拟的目标系统性能作出评价。 SPEC 标准测试集的测试结果表明，SimOS-Goodson 具有较高的模拟精度，其

与真实硬件系统的误差控制在 15%以内。在具有较高模拟精度的同时，

SimOS-Goodson 的模拟速度也较为可观，进行全系统模拟时，其 IPS（Instruction Per

Second）达到了 300K，仅比应用程序级的模拟速度慢约 30%[

17]。

GEMS （General Execution Model Simulator）是 Wisconsin 大学在商业模拟器

Simics 的基础上开发出来的全系统模拟器，它通过加载时序模型来驱动功能模拟

器 Simics 的执行[

18]。它能够细致而灵活的模拟存储系统的结构，包括支持多种

Cache 一致性协议和互连模型。GEMS 的主要优点在于它的灵活和简洁，用户可以

专注于时序模型的设计，而全系统环境下复杂的功能模拟则交由商业模拟器 Simics

来实现，但也正是由于功能模型受时序驱动，每条指令提交时都必须在时序模型

和功能模型之间校验结果，使得时序模拟和功能模拟的部分工作重叠冗余，从而

影响模拟执行的速度。

SS\_SMT[

19]是基于 SimpleScalar 开发的同时多线程（Same Multiple Thread，

SMT）模拟器，其定义的存储层次很复杂，在 Cache 中包括了对 Module 和 Bank

结构的模拟。SS\_SMT 中增加了可配置的译码宽度（每个时钟周期从指令队列中取

出的指令数目）。但是，由于 SS\_SMT 采用一种特殊的 Cache 模型，这种 Cache

模型无法客观地实现对典型超标量体系结构的建模，使得其模拟精度较低；而且

其过多的继承了 SimpleScalar 解释型模拟器的结构，需要对指令进行逐条翻译，所

以其模拟速度也不高。

ARMulator[

20]是 ARM 公司推出的集成开发环境 ADS（ARM Developer Suite）

中提供的指令集模拟器，它模拟的是 ARM 微处理器体系结构和指令集，与运行在

宿主机（通常是 x86 体系结构）上的调试器相连接，提供了开发和调试 ARM 程序

的软件模拟环境。ARMulator 不仅可以模拟 ARM 处理器的指令集体系结构和微体

系结构，而且对存储器和处理器外围设备进行建模，以实现对全系统的支持。

Bochs[

21]是一种开源且高度可移植的全系统模拟器，它完全模拟了 x86 平台的

所有硬件，包括 CPU、Disk、Memory、声卡和其它的周边设备。Bochs 的典型应

用在于提供完整的 x86 体系结构模拟，它目前还支持对 AMD64 CPU 的模拟，在

它上面可以运行 Linux 操作系统。由于它模拟了所有的硬件设备，因此可移植性很

高；但也正是因为它模拟了所有的硬件设备，使得其运行速度非常缓慢。

SkyEye[

22]（天目）是国内一个开源的自由软件项目，它的目标是在通用的 Linux

和 Windows 平台上实现一个纯软件的、基于 ARM 处理器的嵌入式模拟集成开发

环境，目前 SkyEye 主要用来模拟基于 ARM 内核的处理器。SkyEye 可支持多种指

令集体系结构，可以实现多种嵌入式开发板的模拟。由于 SkyEye 是一个全系统模

拟器，除了模拟 CPU 内核，它还模拟了存储器、存储管理单元、Cache、串口、国防科学技术大学研究生院工程硕士学位论文

第 7 页

技术的效果。

希望本次课题的研究能设计实现一个项目组可用的指令集模拟器，以用来辅

助目标处理器进行原型验证和功能验证等。

1.4 论文结构

本论文旨在介绍一种 RISC 处理器指令集模拟器的设计与实现过程，全文共分

为六个章节，具体安排如下：

第一章为绪论。首先介绍本课题的选题背景和意义，其次对指令集模拟器的

研究现状进行描述，最后介绍本文的主要研究内容和论文结构。

第二章为体系结构模拟器理论基础。列举并简要描述体系结构模拟器设计和

实现所涉及到的关键理论、驱动方式以及模拟策略。对于模拟驱动方式，分别介

绍跟踪驱动模拟和执行驱动模拟；对于模拟策略，介绍基于解释型和基于编译型

两种指令集模拟器，包括其工作流程和优缺点等。

第三章为模拟器总体设计。首先对本模拟器的需求进行分析，包括功能需求

和非功能需求等；其次目标处理器的指令集体系结构进行分析，包括指令分类、

寄存器类型以及系统调用等；最后介绍本模拟器的总体框架和总体设计流程。

第四章为模拟器的详细设计与实现。首先对本模拟器的基础平台——Gem5 进

行介绍；接下来结合 Gem5 功能模拟部分代码生成原理，对模拟器的各个模块进

行详细设计与实现，为了提高模拟速度，在指令集模拟过程中，对指令翻译进行

优化，即动态的选择译码信息存入 Cache 中，以达到译码信息复用的目的，从而

缩短译码时间；接下来对存储系统进行建模，并设计实现一个全局的软 TLB 表，

来提高地址转换效率，以提高模拟速度。

第五章为模拟器的测试。本章通过运行 SPEC CPU2000 测试集来对模拟器的

功能和模拟速度进行测试，最后得出结论。

第六章为总结与展望。本章对全文的工作进行总结，并对需要进一步研究的

工作进行分析和展望。

第二章 体系结构模拟器理论基础

模拟，Robert E Shannon（信息论创始人）将其定义为设计一个真实系统的模

型并在这个模型上进行试验以理解系统的行为或者在各种限定条件下评估系统操

作的各种策略[

27]。而模拟器是一种用来模拟其它真实硬件系统的工具，它可以分

为两大类：一类称为硬件仿真器（Emulator），一类是软件模拟器（Simulator）。

前者将软硬件相结合，通过硬件来运行程序，可以很好的提高模拟速度；通过软

件来提供一个界面，以供用户实时了解程序的运行情况。后者根据模拟的详细程

度又可以分为指令集模拟器（功能模拟器）和时钟周期精确模拟器（性能模拟器），

这类模拟器以纯软件的形式在一台现有的计算机（称为宿主机）上模拟另一台计

算机（称为目标机）程序的执行过程。与硬件仿真器相比，软件模拟器的优点在于

它不需要目标机以及任何配套硬件的支持。

2.1 模拟技术理论基础

模拟技术理论最早起源于图灵机。英国数学家艾伦·图灵在著名的《论可计算

的数，以及其在判定问题中的应用》[

28]一文中提出了图灵机的概念，奠定了现代

计算机理论的基础。

图灵还定义了模拟的概念：在不考虑硬件和速度的情况下，不同体系结构的

机器 A 和 B，在理论上可以用指令实现互相模拟（即图灵机）[

29]。图灵机概念的

提出为计算机软件模拟技术的发展奠定了基础。

计算机软件模拟技术实质上就是用计算机软件来模拟实际硬件系统的行为，

以获得该系统的行为数据，方便对其进行分析与研究。模拟开发的基本流程如图

2.1 所示。

进行模拟，首先通过对实际硬件系统建模来将之具体化。建模与具体化的过

程中必须保证所建模型的结构与实际硬件系统相近或一致，以确保所建模型的精

确性，只有精确度较高的模型才能真实的模拟出硬件系统的行为，最终获得正确

的结果。在对硬件系统的建模过程中，需要考虑所选择的算法是否合适。评价一

种算法是否合适的准则在于是否符合模拟的要求和硬件系统的特征。为了保证最

终的模拟精度，必须确保所选择算法的精度够高，稳定性够好。选定合适的算法

后，进行程序设计，即用程序语言将模型描述出来。待确定程序模型的正确以后，

就可以用这个模型来进行模拟实验，得到相应的结果。最后分析模拟结果，结果

分析既可以是针对模型本身的数据，对模型本身进行评价或研究；也可以是对模

拟的目标系统性能作出评价。

2.2 模拟驱动方式

体系结构模拟器的驱动方式主要分为两种：踪迹驱动和执行驱动（又叫程序驱动）。

2.2.1 踪迹驱动模拟

踪迹驱动模拟将每条指令顺序执行所产生的所有信息作为模拟器的输入，从

而模拟某种体系结构处理器的功能和性能[

30]。其模拟过程如图 2.2 所示。

假设要模拟的目标系统为 T，目标系统上执行的应用程序为 w，则模拟 T 中 w

执行的问题可以分解为两个部分：生成踪迹信息以及输入踪迹信息进行模拟。如

果把生成踪迹信息的系统命名为 G，那么，踪迹信息就是 G 执行 w 时产生的所有

信息。接下来将踪迹信息作为模拟器的输入，在目标系统 T 中模拟执行 w。踪迹

驱动的优点在于比较简单，而且可以不关心其它无关部分而只对局部进行详细的

模拟。但是这种方式要求生成踪迹的系统 G与模拟的目标系统 T的体系结构相似，

否则由 G 生成的踪迹信息不能正确反映 w 在 T 上的实际运行情况，从而导致获得

错误的模拟结果。踪迹驱动的另外一个缺点在于模拟器的输入信息是静态的，不

能对目标系统的动态特征进行研究。

2.2.2 执行驱动模拟

对于执行驱动模拟，存在两种不同的观点：一种认为执行驱动模拟器输入的

是程序的二进制文件（可执行指令），而不是踪迹信息，因此，模拟器的输入集

大小只与程序的静态指令数相关，而与动态指令数无关。如图 2.3(a)所示，由模拟

器来执行程序的所有可执行指令。还有一种观点认为，执行驱动模拟器的输入信

息（程序的可执行指令）分成两部分：一部分由宿主机代为执行，另一部分则由

模拟器执行。如图 2.3(b)所示，这类模拟器只执行与所研究问题相关的指令，其它

指令则由宿主机代替执行。这样做的优点在于模拟速度较快，但是这种模拟方式

要求宿主机与目标机的指令集体系结构一致，这就限制了它的应用范围。

与踪迹驱动模拟器相比，执行驱动模拟器将用高级语言编写的应用程序经编

译和链接后得到的二进制文件作为输入信息，在模拟过程中会模拟系统的动态特

征，如动态指令的生成和分支预测等，因此更加接近目标系统的真实情况，精确

度更高。然而，也正是由于需要模拟目标系统的动态特征，使得执行驱动模拟器

的模拟速度较踪迹驱动模拟器更慢。

2.3 指令集模拟策略

指令集模拟策略决定了模拟器的实现方法，是整个模拟器设计的基础。模拟

策略的设计直接影响到模拟器的性能，因此在设计实现模拟器之前，首先需要确

定所采用的模拟策略。模拟策略可分为两种：基于解释型和基于编译型。

2.3.1 基于解释型指令集模拟器

长期以来，由于工作原理简单和模拟精度高，解释型指令集模拟器受到了广

泛的关注（如 Gem5、SimpleScalar 等）。解释型 ISS 最大的特点在于直接将硬件

行为映射到软件[

31]，从而模拟出真实的硬件环境。由于其对指令进行逐条翻译，

使得指令的执行可以很好的被控制。解释型指令集模拟器的工作流程很简单，通

常是取指（fetch）－译码（decode）－执行（execute）的循环，如图 2.4 所示。

1、取指：从模拟器的内存空间中取出一条指令；

2、译码：对取出的指令进行翻译，得到指令的操作码、操作数、执行函数等

信息；

3、执行：执行指令所对应的执行函数，并修改相应寄存器的值，包括 PC 寄

存器；

解释执行的工作流程使得解释型 ISS 设计原理比较简单，易于设计和实现，

且灵活性较好，模拟精度高。由于其对指令进行逐条翻译然后执行，使得其可以

很容易的实现调试机制。但是也正是由于需要对指令进行逐条翻译，导致模拟器

需要在译码阶段花费大量的时间，所以解释型指令集模拟器的模拟速度一般不是

很高。

2.3.2 基于编译型指令集模拟器

随着体系结构的研究越来越深入，其设计的复杂度越来越高，这给模拟器的

性能带来了一定的负面影响。为了提高模拟性能，研究者们对模拟过程进行了改

进，引入了编译的思想——采用一次翻译多次执行的方法来缩短模拟过程中指令

译码的时间。译码就是将目标指令翻译成宿主机可识别的指令，宿主机通过这些

指令来完成对目标机状态的操作。根据译码过程处于编译还是运行时，编译型指

令集模拟器又可分为静态编译型指令集模拟器（Static Compiled ISS）和动态编译

型指令集模拟器（Dynamic Compiled ISS）。

由 Zhu and Gajski[

32]给出的静态编译型指令集模拟器将本处于运行时的指令译

码过程转移至编译时，如图 2.5 所示。目标机二进制代码经编译器编译，之后由代

码生成器优化生成宿主机的二进制代码，并最终运行于宿主机。

该技术有一个缺点，那就是要求程序代码为静态的，这就使得静态编译技术

只能为一小部分数字信号处理器所使用。与典型的 DSP（数字信号处理器）应用

不同的是，微处理器通常面向的是完整的操作系统，而操作系统有一个重要的特

点——程序代码在运行时是动态的，这就局限了静态编译型模拟器的广泛使用。

动态编译型指令集模拟器的典型代表为 Embra[

33]及 Shade

[34]，其工作流程如图

2.6 所示：

其基本思想是复用译码信息，即在内存中取出一条指令后，判断该条指令是

否是第一次执行，若是，那么对其进行译码，并在译码完成后将译码信息保存到

Cache 中，然后执行；若不是，则直接在 Cache 中调用该指令的译码信息执行。由

于该技术在程序运行时进行指令译码，因此很难进行代码优化。

2.4 本章小结

本章首先简要叙述了模拟技术的理论基础以及一般的模拟过程。其次介绍了

两种模拟驱动方式，与跟踪驱动模拟器相比，执行驱动模拟器的模拟精度更高，

更加接近于计算机硬件运行程序的真实情况。最后介绍了两种模拟策略，以及基

于两种模拟策略构建的模拟器的优缺点。由于基于解释型模拟策略的模拟器需要

对指令进行逐条翻译，导致在译码阶段耗费了大量的时间；基于编译型的模拟器则对指令进行一次翻译多次执行，动态编译型模拟器更是重复利用译码信息，这

虽然大大缩短了译码时间，但由于其在运行时进行指令翻译，使得代码很难进行

优化。