

基于 SystemC 构建多核 DSP 软件仿真平台

韦 伟

(中国电子科技集团 第五十八研究所 SOC 研究室,江苏 无锡 214035)

摘 要: 以一款多核 DSP 的设计为背景,讨论了利用 SystemC 构建多核 DSP 软件仿真平台的设计过程.利用 SystemC 建立模型,解决了传统的设计方法的设计复杂、开发周期长和不易修改的问题,该软件仿真平台实现了多核 DSP 的核间通信功能,且可以调试驱动程序、基准测试程序和其他测试程序.通过功能验证,证明利用 SystemC 模型进行软硬件协同设计有效的提高了开发并行度,缩短了开发周期,为功能验证和程序调试提供了更加方便的平台.

关键词: SystemC;多核 DSP;软件仿真

中图分类号: TP319

文献标识码: A

文章编号: 1000-7180(2016)04-0086-05

Construct Multi-core DSP Software Simulation Platform Based on SystemC

WEI Yi

(NO. 58 Research Institute, China Electronics Technology Group Corporation, Wuxi 214035, China)

Abstract: This paper takes a multi-core DSP design as the background, discusses the design process of using SystemC to construct software simulation environment of multi-core DSP. Using SystemC model solves the problem of traditional design method. The traditional design method is complex, its development cycle is long and it is not easy to modify, the software simulation platform realized the communication among the multi-core DSP and users can debug driver program, benchmark programs and other test programs. Through the functional test, the paper verifies that using SystemC model for hardware and software design effectively improves the development degree of parallelism, shorten the development cycle, provides a more convenient platform for the functional verification and program debugging.

Key words: SystemC; multi-core DSP; software simulation

1 引言

多核 DSP 的发展已经成为 DSP 发展的主流解决方案之一.在对多核 DSP 的开发过程中,如何对多核运行平台进行软件仿真成为了必须要解决的问题.传统的设计方法必须首先完成系统的 RTL 设计,才能建立 RTL 仿真验证平台进行应用程序的调试,然而多核 DSP 的架构复杂且需要具有核间通信的功能,这就导致 RTL 设计复杂且调试困难,项目开发周期较长.另外,RTL 模型建立的仿真平台不易于修改,不能满足多核 DSP 的快速设计和验证调试的要求.

采用 SystemC 构建多核 DSP 的软件仿真平台可以很好地解决上述问题,建立 SystemC 模型具有抽象层次高,设计时间短,且易于修改的特点^[1].

SystemC^[2]是基于 C++语言的. C++是一种应用广泛的面向对象的建模语言. SystemC 扩展了 C++的能力,包含了用于对硬件组件进行建模以及硬件组件之间交互的结构^[3],它在 C++的基础上增加了如并发(多个处理器同时执行)、定时事件和数据类型等概念. SystemC 还增加了一个类库,扩展了 C++的能力,该类库是一个由使用合法 C++代码编写的函数、数据类型和其他语言结构函数所组成的库.该库可以为具有硬件时序、并发和相应

行为的系统结构建模,可以提供用于描述信号、模块和端口的构造函数.借助类库中的类,使用者可以定义模块、进程以及通过端口的通信和信号.

因为 SystemC 是 C++ 的扩展,所以可以使用标准的 C++ 编程语言和开发工具,创建各种不同架构和算法的设计,并对其进行仿真、调试和研究^[4].

SystemC 给出了构建系统结构模型所必须的元素并允许使用者利用所熟悉的 C++ 语言和开发工具进行系统的设计和开发^[5].

本文讨论了利用 SystemC 建立多核之间的 NI 模块的设计过程,从而实现平台的核间通信的功能,支持使用者对应用程序和测试程序的调试,可以让使用者在硬件调试前确保程序的正确性,这样可以有效的缩短开发周期.

2 系统架构和功能概述

多核 DSP 的系统架构通过 NI 和 Router 将多个核连接起来实现核间通信. NI (Network Interface),是 DSP core 和路由网络 Router 之间的转接口,是本文构建的软件仿真平台的核心部分,该模块主要功能有:对 AHB 总线数据和 Router 的数据进行数据格式转化;实现本地 DSP core 到其他 DSP core 存储空间的数据读写;本地存储空间和其他 DSP core 存储空间之间的大批量数据交互.多核间的系统结构如图 1 所示. NI 的结构框图如图 2 所示.

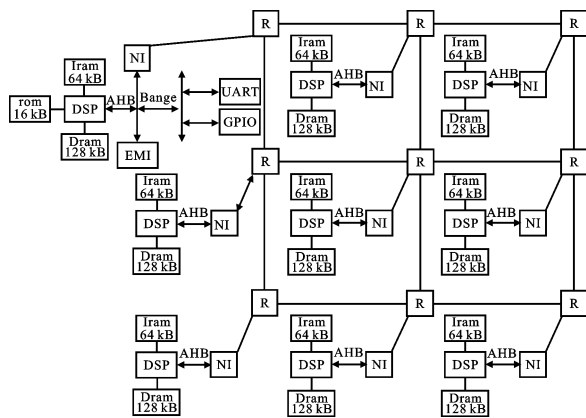


图1 多核系统整体结构框图

软件仿真平台要实现核间的通信功能,包括直接写模式、搬运模式和读模式.直接写模式:适用于少量数据的传输,由本地 DSP core 主动发起,向目的地址发送 32 bit 的数据.搬运模式用于大批量数据的传输,由本地 DSP core 主动发起,向目的地

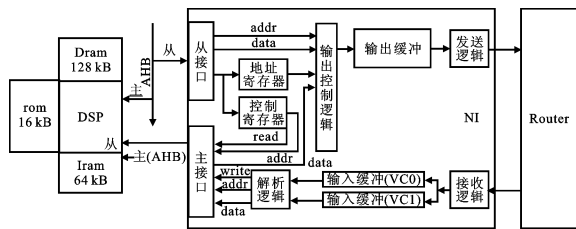


图2 NI 的结构框图

址发送最多 4 kB 的数据.读模式是搬运模式的变种,是本地 DSP core 写操作和搬运操作的结合,即通过本地 DSP core 将目标 DSP core 对应的 NI 模块配置为搬运模式.

NI 寄存器分三类:直接写模式寄存器、搬运模式寄存器和中断控制寄存器.直接写模式寄存器包括写控制寄存器和写目的地址寄存器;搬运模式寄存器包括搬运源地址寄存器、搬运目的地址寄存器和搬运控制寄存器.

3 系统设计

3.1 软件仿真平台总体架构设计

本文所构建的软件仿真平台由三部分组成:9 个 DSP core、9 个 NI、1 个 Router 和 1 个 Arbiter.其组成关系、平台的端口名称及连接关系如图 3 所示.

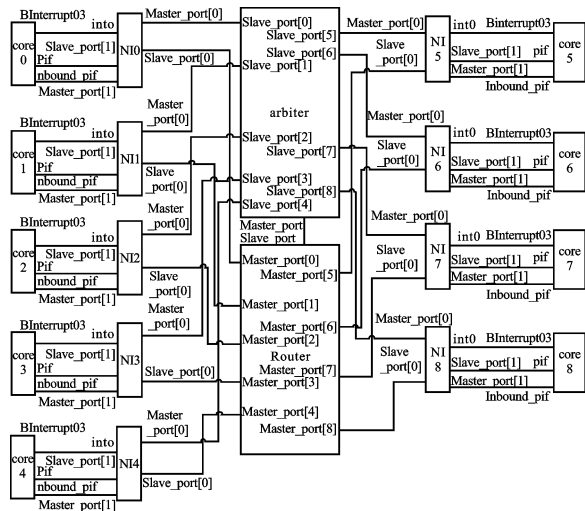


图3 多核 DSP 软件仿真平台端口名称及连接关系

每个 NI 小模块有 5 个端口,包括 2 个用于发送的 Master_port、2 个用于接收的 Slave_port 和一个中断信息发送端口 int. Arbiter 和 Router 共同协作实现硬件架构中的 Router 模块的功能,分别具有 9 个 Slave_port 和 9 个 Master_port 连接到 ni0—ni8, Arbiter 用于接收数据, Router 用于发送

数据.

3.2 NI 模块的构建

根据图 2 可知,NI 模块有两个发送端口和两个接收端口,NI 模块的处理逻辑需要把接收到的请求根据端口、类型和地址等信息放入对应的缓冲中,并从缓冲中取请求从正确的端口发出,对请求相应的反馈信息也要做出正确的处理.

3.2.1 NI 模块的构造参数定义

```
class ni_parms {
    .....
}
```

参数包括:master_port 和 slave_port 的个数,字节大小和 NI 模块的起始地址. 默认的构造参数为:1 个 master_port,1 个 slave_port,8 个字节,起始地址为 0x7007000.

3.2.2 数据的打包

在本文的软件仿真平台中,数据包以 request (请求)和 response(反馈信息)的形式传输. Master_port 可以发送 request,并接收对应的 response. slave_port 可以接收 request,并返回 response.

request 包含的信息有: ID、类型 (READ/ WRITE)、地址、数据(类型为 WRITE)等.

response 包含的信息有:对应的 request 的 ID、地址、数据(读 request 对应的 response)等.

3.2.3 缓冲的建立

为了使 NI 在同时进行接受和发送时没有冲突产生,且可以实现多个 DSP core 对一个 DSP core 的同时读写,在 NI 模块内部设计了输入和输出缓冲,用于存放请求. 代码如下:

```
deque<request>      m_outfifo_pool;
deque<request>      m_infifo_pool;
```

3.2.4 NI 模块的定义

```
SC_MODULER(NI)
{
    .....
}
```

定义了一个名为 NI 的模块类,在 NI 模块的定义中,需要给出 NI 端口、事件和方法的定义.

NI 端口定义包括:数据的输入和输出端口、中断信息的输出端口. 可以利用 SC_PORT 来定义,并依照端口的方向加以说明,有单向的输入输出端口类别,也有双向的输入输出端口类别,分别用 SC_IN、SC_OUT 和 SC_INOUT 进行说明,并且定义端口的数据类型.

NI 相关的事件包括本地 DSP core 通过 NI 和

Router 对目标 DSP core 的读写操作、其他 DSP core 通过 NI 和 Router 对本地 DSP core 的读写操作、NI 接收到了请求或反馈信息和各个缓冲区非满或非空的事件.

利用 SC_METHOD 来描述 NI 模块的功能,该函数的调用需要收到指定的量的驱动,例如如下语句定义了 NI 模块的方法进程 ni_request_process,且在 NI 接收到请求时调用该进程. 在进程 ni_request_process 中,编写了根据寄存器状态判别 NI 模块所处模式的逻辑代码,并进入相应的逻辑代码处理数据的通信. 语句 dont_initialize()可以避免启动时的执行,代码如下:

```
SC_METHOD(ni_request_process);
sensitive<<m_request_event;
dont_initialize();
```

本文的软件仿真平台在设计过程中还用到了 SystemC 的另一种进程 SC_THREAD, SC_METHOD 定义的进程不能被挂起,也不能包含无限循环. 而在本软仿平台中需要循环进行发送请求和将请求或反馈信息放入缓冲等处理. 因此,本文另外利用 SC_THREAD 定义了进程,这类进程可以被挂起,也可以在一定延迟之后或者基于某个事件的发生而重新恢复执行,代码如下:

```
SC_THREAD(send_request2dspcore_thread);
SC_THREAD(send_request2router_thread);
SC_THREAD(push_infifo_thread);
SC_THREAD(push_outfifo_thread);
SC_THREAD(pool_thread);
```

在定义上述进程的执行过程中利用了 SystemC 的动态敏感性,上述进程的运行取决于前面定义的事件的发生,利用 wait()方法可以实现. 例如:

```
void
xtsc_ni::send_request2dsp_thread(void)
{
    while(true)
    {
        .....
        wait(m_responseFdsp_event);
        .....
    }
}
```

该进程用于处理从输入缓冲取请求发送给 DSP core,需要在接收到 DSP core 对上一个请求的反馈信息后再发送下一个请求,上述代码结构就可以完成这样的操作. 其他进程定义类似,并用相应的事件触发器运行.

3.3 NI 模块的工作模式

3.3.1 写模式

由本地 DSP core 主动发起,主要过程是配置写目的地址寄存器和写控制寄存器,然后向目的地址发送 32 bit 的数据。

以 core0 向 core1 直接写数据为例:

首先,在 core0 为 ni0 的直接写目的地址寄存器和控制寄存器赋值,ni0 收到设置寄存器的写请求,设置对应寄存器的值,并返回反馈信息给 core0。

然后,core0 向 ni0 发送一条写数据的请求,ni0 收到请求后进入直接写模式,如果输出缓冲没有满,则将请求放入输出缓冲并向 core0 返回反馈信息,否则等待直至输出缓冲非满。

当 ni0 的输出缓冲非空时,ni0 则向 Router 按序发送输出缓冲中的请求,请求通过 Router 到达 ni1。

ni1 收到写请求,根据其地址信息判断其是要写到本核的请求。如果 ni1 的输入缓冲没有满,则将请求放入 ni1 的输入缓冲,否则等待直至输入缓冲非满。

当输入缓冲非空且 ni1 不处于搬运模式时,从输入缓冲中按序取出请求,修改请求中的全局地址为本地地址,发送给 core1 并返回反馈信息给 Router。

core1 收到请求后返回反馈信息给 ni1。

3.3.2 搬运模式

由本地 DSP core 主动发起,主要过程是配置搬运目的地址寄存器、搬运源地址寄存器和搬运控制寄存器使 NI 进入搬运模式。

以 core0 向 core1 搬运数据为例:

首先,在 core0 为 ni0 的搬运目的地址寄存器、搬运源地址寄存器和搬运控制寄存器赋值,ni0 收到设置寄存器的写请求,设置对应寄存器的值,并返回反馈信息给 core0。

然后,ni0 进入搬运模式,如果输出缓冲没有满,则向 core0 发送一个读请求(地址为搬运源地址),ni0 在收到 core0 的反馈信息后,获取反馈信息中的数据创建新的请求放入输出缓冲中。

当 ni0 的输出缓冲非空时,ni0 则向 Router 按序发送输出缓冲中的请求,请求通过 Router 到达 ni1。

ni1 收到写请求,根据其地址信息判断其是要

写到本核的请求。如果 ni1 的输入缓冲没有满,则将请求放入 ni1 的输入缓冲,否则等待直至输入缓冲非满。

当输入缓冲非空且 ni1 不处于搬运模式时,从输入缓冲中按序取出请求,修改请求中的全局地址为本地地址,发送给 core1 并返回反馈信息给 Router。

core1 收到请求后返回反馈信息给 ni1。

3.3.3 读模式

NI 的读模式实质上就是搬运模式,是写模式和搬运模式的结合。首先,本地 DSPcore 通过配置本地 ni 为直接写模式来配置目标 DSPcore 的 ni 的搬运寄存器,使目标核 ni 进入搬运模式。之后目标 DSPcore 的存储器被读取后通过路由发送到本地 ni,本地 ni 将数据存入本地存储器中。

3.3.4 NI 模块处理流程图

NI 模块的处理流程图如图 4 和图 5 所示,nb_request 和 nb_respond 函数用于接收请求和反馈信息,在 NI 模块接收到请求和反馈信息是被触发。

4 功能验证

本文设计的多核 DSP 软件仿真平台支持驱动代码和基准测试程序的调试运行,通过编写核间数据通信的驱动代码,并在平台上编译运行,对软件仿真平台的功能进行了验证。

在本文的功能验证过程中,对多种核间的通信方式(写模式、搬运模式和读模式)都进行了测试,数据都可以正常通信,并且由于给各个收发端口都设计了缓冲区,每个 DSP core 都可以同时实现接受和发送功能并不冲突,且每个核都可以同时接受来自其他多个核的数据不会丢失。

5 结束语

本文讨论了基于 SystemC 构建多核 DSP 软件仿真平台的过程,涉及软件仿真平台架构的设计和 NI 模块的建模过程。通过编写 SystemC 代码,实现了软件仿真平台的构建,并在平台上进行了多种数据通信方式的测试,验证了平台的正确性。利用 SystemC 进行软硬件协同设计有效缩短了芯片设计开发的周期,解决了传统的 RTL 仿真平台不能提供快速系统仿真和软件调试的问题,成为一种高效的性能分析和验证平台。

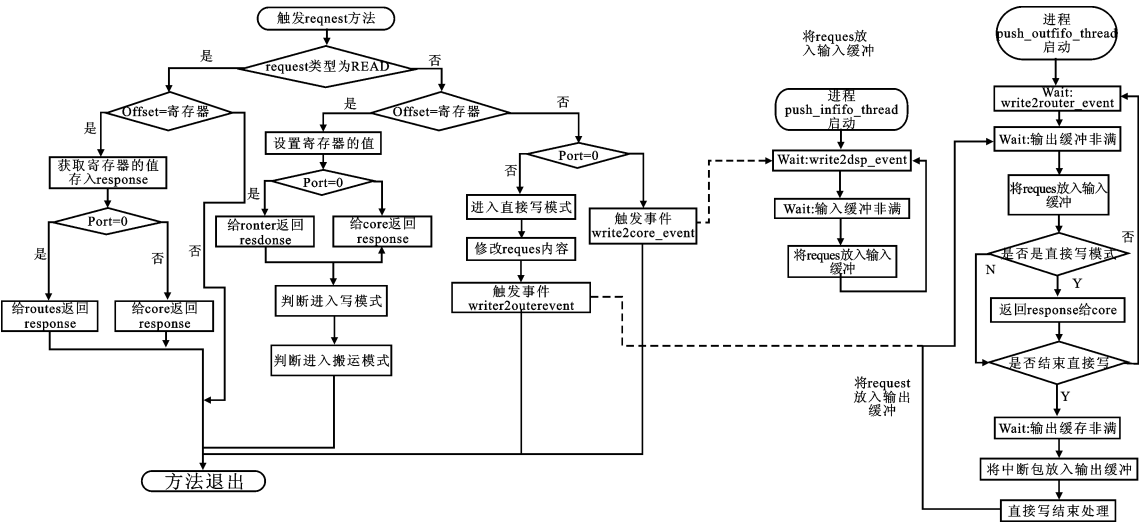


图 4 NI 模块流程图

从输入缓冲取request发给核
发送读请求给核

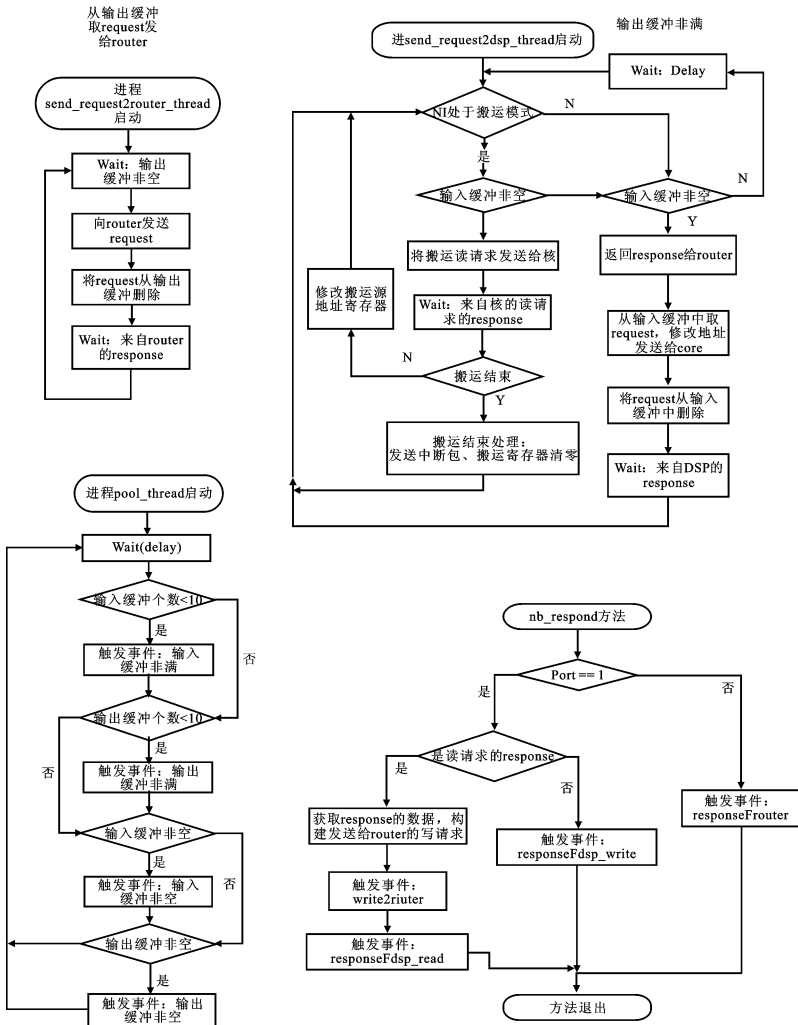


图 5 NI 模块流程图

tion, THD) 为 -94.929 9 dB, 信噪比 (Signal to Noise Ratio, SNR) 为 58.754 4 dB.

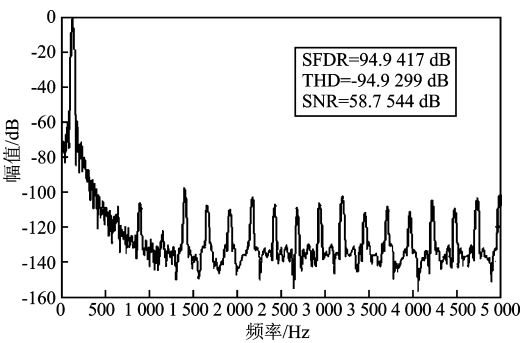


图 8 动态性能仿真

5 结束语

本文提出一种对多位每级流水线 ADC 进行 VHDL-AMS 结构式建模方法,此方法新颖、简单、仿真速度快.经 12 位流水线 ADC 静态性能和动态性能仿真,仿真结果正确无误,不会出现失码和缺码现象,验证了采用此方法建立流水线 ADC 模型的有效性.所建模型具有以下优点:

- (1)根据文中多位每级输出数字码的形式,用户可以自由修改流水级结构来满足使用需求;
- (2)模型考虑了非理想因素带来的误差,参数可调,模型符合器件实际工作性能;
- (3)基于 VHDL-AMS 的 ADC 模型在大型数模混合信号系统建模仿真时可以任意调用,移植性强.

参考文献:

[1] Lan D, Liu X D. Behavioral model based on simulink for 14-bit 200MS/s pipelined ADC[C]// International

Conference on Control Engineering and Communi- cat- ion Technology. Shenyang:IEEE, 2012: 79-82.

[2] 申振胜. 基于 VHDL-AMS 的 ADC 建模及仿真[D]. 西安:西安电子科技大学,2014.

[3] IEEE-SA Standards Board. IEEE Std 1076. 1-1999. Definition of analog and mixed signal extensions to IEEE Standard VHDL[S]. New York: The Institute of Electrical and Electronics Engineers, Inc, 1999.

[4] 孙可旭,何乐年. 基于频域特性的流水线 ADC 数字校正技术[J]. 浙江大学学报:工学版,2013, 47(8): 1393-1301.

[5] 刘蒲霞,陆铁军,王宗民. 16 位流水线 ADC 系统级建模及仿真[J]. 微电子学与计算机,2009, 26(12): 120-121.

[6] AMS Modeling Cookbook for VHDL-AMS. Mentor Graphics Corporation[R]. MentorGraphics Corpora- tion, 2010.

[7] 薛亮. 10 位高速流水线模数转换器的研究[D]. 北京:清华大学,2004:42-50.

[8] Mikael G, J Jacob Wikner, Tan N X N. CMOS data converters for communications[M]. Boston: Kluwer Academic Publishers, 2000.

[9] Xu D J, Jun P. The study of code density histogram testing method to high-speed ADC based on DSP[C]// Proc of the 8th World Congress on Intellgent Control and Automation. Jinan:IEEE, 2010: 2069-2074.

作者简介:

陈世同 男,(1980-),博士,讲师.研究方向为光纤陀螺信号检测与处理技术、光纤陀螺捷联惯性导航技术.
E-mail:gsdxzdd@sina.com.

赵振栋 男,(1991-),硕士研究生.研究方向为数模混合信号系统建模仿真技术、惯性器件与系统建模技术.

(上接第 90 页)

参考文献:

[1] Ma Guire. Issues In the development of an integrated environment for Implementation[J]. Microprocessors and Microsystems,1999, 23(4):199-206.

[2] SystemC v2.0. 1 UserGuide [EB/OL]. [2015-04-20]. <http://www.systemc.org>

[3] 王国富. 浅谈基于 SystemC 的系统设计[J]. 科技创新与应用,2014,24(16):60-60.

[4] 夏宇闻,甘伟. SystemC 入门[M]. 北京:北京航空航天大学出版社,2008.

[5] 杨光,冉峰. 单片机汇编器的设计与实现[J]. 微型计算机应用,2005,26(2):231-233.

作者简介:

韦 祎 女,(1987-),硕士研究生.研究方向为软件与理论.
E-mail:wyi811@126.com