第34卷 第22期 计 算 机 工 程 2008年11月
Vol.34 No.22 Computer Engineering November 2008
·工程应用技术与实现· 文章编号：1000—3428(2008)22—0236—03 文献标识码：A 中图分类号：N945

# 主从式片上总线系统交易级的实现

黄清泉[1]，洪 沙[2]，吴垣甫[1,2]
(1. 海军驻重庆662厂军事代表室，重庆 400021；2. 重庆大学软件学院，重庆 400021)

**摘 要：**在总线的主设备上增设了实时操作系统的任务优先级分配机制，基于蒙特卡罗选择实现总线仲裁器的仲裁策略，建立不同类型的从设备存储器模型。运用SystemC在交易级实现整个总线系统模型，并对该模型进行仿真。实验结果证实了仲裁算法的有效性。
**关键词：**片上总线；优先级；仲裁策略；SystemC语言；交易级建模

# Implementation of Bus-on-chip System with Master-slave Pattern at Transaction Level

HUANG Qing-quan[1], HONG Sha[2], WU Yuan-fu[1,2]
(1. Navy Martial Representation of 662 Manufactory, Chongqing 400021; 2. College of Software, Chongqing University, Chongqing 400021)

**【Abstract】**The mechanism of task priorities assignment in Real Time Operating System(RTOS) for master devices is added, and the arbitrate policy of arbiter based on Monte Carlo methods is implemented. Moreover, various memory models for slave devices are also set up. The whole model of bus system is completed at transaction level by using SystemC. Simulation results illustrate this arbitrate algorithm is effective.
**【Key words】**bus-on-chip; priority; arbitrate policy; SystemC; Transaction Level Modeling(TLM)

运用SystemC进行交易级建模(Transaction Level Modeling, TLM)逐渐成为业界的热点[1-2]。本文主要研究如何使用SystemC在交易级实现主从式片上总线系统，并对典型的主从式片上总线系统进行交易级建模、扩充以及改进其优先级调度和仲裁算法。

## 1 主从式片上总线系统

目前，片上互连网络分为共享介质(shared-medium)网络、直接/非直接(direct and indirect)网络以及混合(hybrid)网络[3]。其中，片上共享介质结构在现有技术中体现为底板总线(backplane bus)，适合不对称网络架构，如AMBA 2.0 bus。一个较为典型的主从式片上总线系统如图1所示。

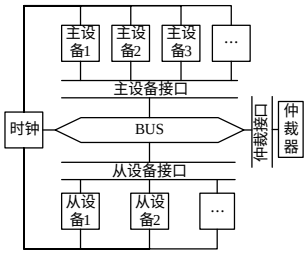


**图1 典型的主从式片上总线系统**

其工作原理为：总线在时钟上升时收集主设备请求，在时钟下降沿时将请求送入仲裁器(arbiter)，由仲裁器仲裁后决定响应哪个主设备的请求，得到响应的主设备拥有总线使用权，可以和从设备进行数据交换及通信。这里假定有3个主设备(直接型、阻塞型和非阻塞型)、2个从设备(快/慢速存储器)和1个仲裁器(基于蒙特卡罗选择的仲裁策略)。在该结构中用户可根据自己的需要开发更多的主设备和从设备，如处理器、DMA控制器、通用串口、中断控制器。在系统建模过程中系统的识别类及其关系如图2所示。

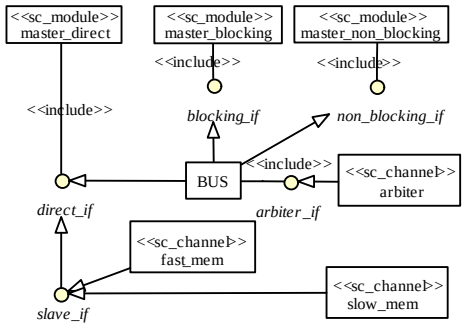


**图2 片上总线的模型类**

图2是UML类图[4]，主设备用SystemC中的模块表示，其类的版型均为sc_module。从设备、仲裁器以及总线用SystemC中的通道表示，其类的版型均为sc_channel。使用接口方法调用[5]实现整个程序架构，在各接口中定义各种类型设备的方法(对应C++的纯虚函数，如阻塞读、直接写和仲裁函数等)，在模块或通道中实现函数的具体细节。

## 2 主设备优先级调度策略

实时系统的调度算法分为以下几类：抢占式(preemptive)和非抢占式(non-preemptive)，静态(static)和动态(dynamic)，周期(periodic)和非周期(aperiodic)，详细分类可参考文献[1]。

### 2.1 EDF调度和RF调度

EDF(Earliest Deadline First)调度是动态调度算法，其基本思想是依据任务的最后期限(deadline)进行排序，离deadline越近的任务被分配的优先级越高。该调度算法能减少系统运行迟滞。RF(Rate Monotonic)调度是静态调度算法，

**作者简介：**黄清泉(1963－)，男，高级工程师、硕士，主研方向：计算机技术，软件工程；洪 沙，博士后；吴垣甫，硕士
**收稿日期：**2008-02-10 **E-mail：**wuyuanfu@qq.net

其基本思想是依据各个任务的周期分配唯一的优先级，该调度算法适合单处理器调度系统。SystemC 的仿真内核是非抢占式的，总线上的任务一旦执行则必须完成，否则后面的任务将被阻塞。

### 2.2 任务优先级分配机制

针对上述调度思想及示例中的主设备设计任务优先级分配机制，并将其嵌入 RTOS，如图 3 所示。

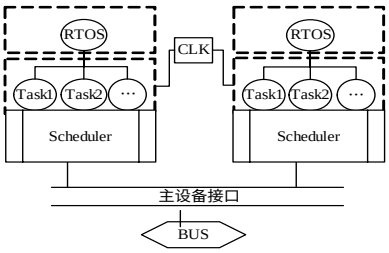


**图 3 主设备任务调度示意图**

对照图 1 和图 3 可知，主设备可能执行多个任务(Task)，RTOS 可以管理多种主设备，多种主设备也可由多个 RTOS 来管理。本例中使用 1 个 RTOS 管理 3 个阻塞和 1 个非阻塞型主设备的周期任务。基于图 3 的架构能实现多种调度算法，分配不同任务的优先级，本例采用 RF 调度算法分配 4 个任务的不同优先级。

## 3 仲裁器仲裁策略实现

不同仲裁策略实现不同的片上总线，总线性能的好坏与其仲裁策略相关。若按照第 2 节给出的优先级进行简单仲裁，即先执行优先级高的任务，则会导致某些优先级较低的请求长时间得不到满足，造成所谓的饥饿问题[6]。而基于蒙特卡罗选择(Monte Carlo)的仲裁策略较好地解决了这一问题。

### 3.1 蒙特卡罗选择

按优先级比例分配任务执行概率，称之为选择的蒙特卡罗法。以赌轮的方式在圆盘上按优先级分配各个任务所占面积，通过产生随机数并判断其范围决定最终任务的执行。设某时刻 3 个阻塞型任务的优先级分别为 2, 4, 6，非阻塞型任务优先级为 8，则轮盘面积分配如图 4 所示。

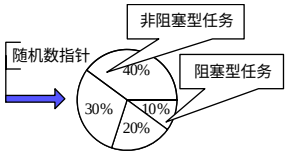


**图 4 按优先级比例分配执行概率**

此时产生随机数并判断其所属区间，所属区间表示的任务将被执行。优先级越高则面积越大，其相应的执行概率也越高。该方法和简单仲裁的差别在于即使任务优先级不是最高也有得到执行的可能。

### 3.2 仲裁算法实现

在系统执行过程中每次送给仲裁器一组任务请求，仲裁算法按蒙特卡罗选择思想根据请求的优先级分配任务执行面积，产生随机数并计算其区间，返回属于该区间的任务请求给总线。部分核心代码如下：

```
int p=0;
srand( (unsigned)time( NULL ) ); //产生随机数种子
for (i = 0; i < requests.size(); ++i)
{p+=requests[i]->priority;}   //计算总面积
int r=rand()%p;       p=0;          //计算随机数位置
for(i=0;i<requests.size(); ++i){
bus_request *request = requests[i];
  p+=requests[i]->priority;
  if(r<=p) return request;}//选择得到响应的请求
```

## 4 从设备存储器模型和直接型主设备

典型的从设备有串口、存储器和中断控制器，本例中选择快/慢速 2 种类型存储器作为从设备。其中，快速存储器(Fast_mem)实现 0 读写时间的存储器模型，这类存储器模型只关注行为，实现简单，仿真速度快。慢速存储器(Slow_mem)则需等待若干个时钟周期才能完成 1 次读/写操作，它能对类似 SDRAM 的实际存储器建模。系统仿真过程中使用这 2 种类型的存储器记录相关交易数据，通过直接型主设备读取数据，并进行观察和分析。由于快速存储器实现 0 时间读写，直接型主设备任务不需要仲裁，因此本例中的系统时钟不和它们相连，完整的系统如图 5 所示。

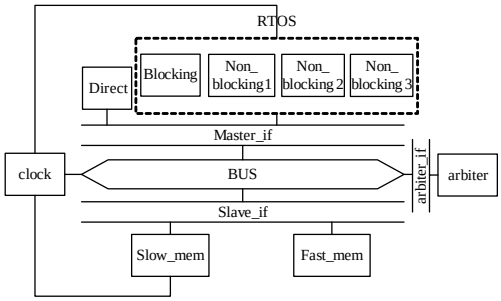


**图 5 示例中完整的片上总线系统**

## 5 系统结构探索和仿真

### 5.1 系统结构探索

设计初期通常需要快速掌握一些关键参数以判断系统各类型的设备配置是否合理。一个不好的配置虽能完成任务，但其执行时间、成本以及能耗可能难以承受。基于交易级建模的目的是在设计初期探索系统结构，通过探索得出用户需要的关键参数，从而初步优化设计架构。

这里以系统总执行时间内不同任务执行的次数为关注焦点，通过仿真验证交易级模型的行为是否满足预期要求和仲裁算法的有效性。基于本例的交易级平台还可从其他角度(如任务执行时间、总线数据流量)进行仿真，这取决于用户关心的方向。

### 5.2 系统仿真

假定快速存储器地址范围为 0x00~0x7f，慢速存储器地址范围为 0x80~0xff，等待周期为 1。非阻塞型任务 Task1 和阻塞型任务 Task2~Task4 的相关参数如表 1 所示。

**表 1 非阻塞和阻塞任务相关参数表**

| TASK | timeout | Start address | priority |
|---|---|---|---|
| Task1(non_blocking) | 20 | 0x38 | 8 |
| Task2(blocking) | 80 | 0x8c | 2 |
| Task3(blocking) | 60 | 0x1c | 4 |
| Task4(blocking) | 40 | 0x2c | 6 |

图 6 为系统在某段时间内任务的执行概况。

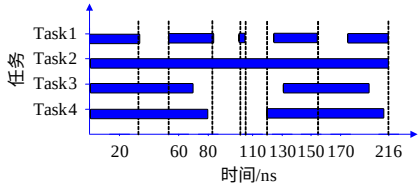


**图 6 系统某次执行示意图**

由图 6 可知在 216 ns 内，Task1 共计执行 5 次，Task3 和 Task4 执行 2 次，Task2 只执行 1 次。各个任务执行的次数不仅与其优先级相关，还和任务访问的存储器类型相关，Task2 优先级最低，且同时访问了慢速存储器，执行时间较长，可以预计在系统总执行时间内该任务执行的次数将会最少。图 7 和图 8 列举了系统某次执行过程中其中 2 个任务运行 3 000 ns 后的情况，该图由 MATLAB 自动生成。

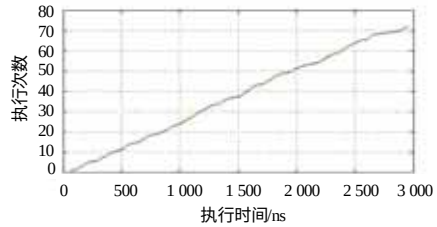


**图 7　Task1 的运行情况**

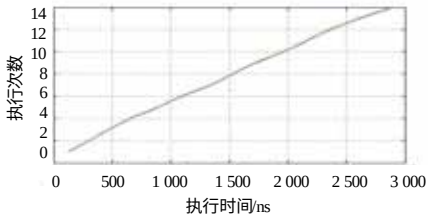


**图 8　Task2 的运行情况**

由图 7、图 8 可知，Task1 执行次数最多(72 次)，Task2 执行次数最少(14 次)，各任务的执行概率基本符合其优先级分布，从而证实了基于蒙特卡罗选择的仲裁算法是有效的，通过该图还能方便地获取任务执行时间等关键参数。

## 6　结束语

使用SystemC进行片上系统交易级建模是目前片上系统(System on a Chip, SoC)级设计中的主要方法学[7]，本文介绍SoC设计中常见的设计元素——主从式片上总线系统，通过对总线上主、从设备和仲裁器以及总线本身的交易级设计构建一个整体的交易级模型平台，对模型的仿真验证了仲裁算法，说明TLM在系统结构探索中的作用。

## 参考文献


[1] Marwedel P. Embedded System Design[M]. The Netherlands: [s. n.], 2006.

[2] 陈　琳, 刘　瑰, 徐晏琦, 等. SystemC 在 SoC 总线交易级建模的研究与应用[J]. 计算机工程, 2006, 32(16): 240-242.

[3] Jerraya A, Wolf W. Multiprocessor Systems-on-Chips[M]. San Francisco, USA: Morgan Kaufmann Publishers, 2005.

[4] 郭　宁. UML 及建模[M]. 北京: 清华大学出版社, 2007.

[5] 陈　曦, 徐宁仪. SystemC 片上系统设计[M]. 北京: 科学出版社, 2004.

[6] Kamal R. 嵌入式系统——体系结构、编程与设计[M]. 陈曙晖, 译. 北京: 清华大学出版社, 2005.

[7] Ghenassia F. Transaction Level Modeling with SystemC[M]. The Netherlands: [s. n.], 2005.


(上接第 235 页)

**表 1　扩展相容性多扫描树测试应用时间和测试功耗数据**

| Circuits | #FFs | #test | FC/(%) | *N*=2 时比率/(%) | | *N*=3 时比率/(%) | | *N*=4 时比率/(%) | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | 平均测试功耗 | 测试应用时间 | 平均测试功耗 | 测试应用时间 | 平均测试功耗 | 测试应用时间 |
| S1423 | 74 | 88 | 98.99 | 58.1 | 48.9 | 69.5 | 66.7 | 75.4 | 73.3 |
| S5378 | 179 | 162 | 99.05 | 60.8 | 49.9 | 73.5 | 67.3 | 78.9 | 74.4 |
| S9234 | 228 | 348 | 93.16 | 57.2 | 50.0 | 70.2 | 66.9 | 76.7 | 74.5 |
| S13207 | 669 | 80 | 98.32 | 57.0 | 48.8 | 70.0 | 64.7 | 76.4 | 74.4 |
| S15850 | 597 | 351 | 96.37 | 55.1 | 50.6 | 68.5 | 66.5 | 75.7 | 74.5 |
| S35932 | 1 728 | 2 879 | 88.61 | 54.3 | 52.4 | 67.9 | 66.7 | 72.6 | 76.2 |
| S38417 | 1 636 | 651 | 99.39 | 52.2 | 50.1 | 67.1 | 66.2 | 75.2 | 70.4 |
| S38584 | 1 452 | 1 056 | 95.00 | 58.1 | 50.3 | 71.5 | 66.5 | 78.6 | 75.1 |
| Av. | - | - | - | 56.6 | 50.1 | 69.8 | 66.4 | 76.2 | 74.1 |

在表 1 中，前 4 列给出了电路名称、扫描单元个数、测试向量数和故障覆盖率，后面 3 列给出了当多扫描树的扫描输入个数*N*分别为 2, 3, 4 时，本文提出的方法相对于文献[5]中平均测试功耗减少的比率和测试应用时间减少的比率。这里的测试应用时间是用扫描移位时间来度量的。对于单扫描链结构，测试应用时间$T=n\times(v+1)$；对于扫描树测试结构则有$T=l\times(v+1)$，其中，$n$为扫描单元的个数；$v$为测试向量数；$l$为扫描树的层数。在CMOS电路中，测试功耗正比于电路的时钟频率及开关跳变，因此，使用扫描单元在扫描测试过程中发生的跳变数[9]来估计测试功耗。从表 1 中还能看出，当*N*=2, 3, 4 时测试应用时间平均减少 50.1%, 66.4%, 74.1%，最高减少 52.4%, 66.9%, 76.2%；测试功耗平均减少 56.6%, 69.8%, 76.2%，最高减少 60.8%, 73.5%, 78.9%。对于*N*=2 的 S35932 电路，测试应用时间减少 52.4%，大于 50%。这是合理的，因为位于多扫描树 2 个输入端的类可能存在逻辑异或相容关系，若树上有多个层次存在这种关系，则其测试应用时间可能降低 50%以上。

## 6　结束语

本文提出一种新的多扫描树结构以减少测试应用时间和测试功耗，此结构适用于具有任意扫描输入的电路。实验数据表明，当同时存在多个扫描输入时，该方法在减少测试应用时间和测试功耗方面具有更高的效率。

## 参考文献


[1] Miyase K, Kajihara S. Optimal Scan Tree Construction with Test Vector Modification for Test Compression[C]//Proc. of IEEE Asian Test Symposium. [S. l.]: IEEE Computer Society, 2003.

[2] Miyase K, Kajihara S, Reddy S M. Multiple Scan Tree Design with Test Vector Modification[C]//Proc. of IEEE Asian Test Symposium. [S. l.]: IEEE Computer Society, 2004.

[3] Yotsuyanagi H, Kuchii T, Nishikawa S, et al. Reduce Scan Shifts Using Folding Scan Trees[C]//Proc. of IEEE Asian Test Symposium. [S. l.]: IEEE Computer Society, 2003.

[4] Bonhomme Y, Yoneda T, Fujiwara H, et al. An Efficient Scan Tree Design for Test Time Reduction[C]//Proc. of IEEE European Test Symposium. [S. l.]: IEEE Computer Society, 2004.

[5] Inoue M, Fujiwara H. Extended Compatibilities for Scan Tree Construction[C]//Proc. of IEEE European Test Symposium. [S. l.]: IEEE Press, 2006.