DOI:10.16182/j.cnki.joss.2008.09.046

基于组件的计算机组成原理虚拟实验室的设计与实现

王建新，张丽媛，盛　羽，刘丽娟
（中南大学信息科学与工程学院，湖南长沙 410083）

摘　要：提出了一种**基于组件技术的计算机组成原理虚拟实验室（PCOVL）的设计模型和实现方法。**该系统采用Java语言实现，具有良好的平台无关性。以Java Bean组件技术开发元器件，并利用Java反射技术实现了系统动态调整组件属性和行为的功能；结合数据驱动原理，多线程技术的wait、notify机制，以及锁的同步控制技术，**提出一种基于数据驱动的触发式调度机制**，有效的解决了具有复杂关系的组件之间的数据传递和调度运行的问题。PCOVL实现了可视化的定制实验流程、运行期间随机改变组件状态、以及实验结果的动态实时显示等功能，为开发计算机硬件类课程虚拟实验室提供了有力的技术支持和理论基础。

关键词：虚拟实验室；计算机组成原理；Java Bean；数据驱动；多线程
中图分类号：TP391.9　　**文献标识码：**A　　**文章编号：**1004-731X (2008) 09-2469-06

Design and Implementation of Principles of Computer Organization Virtual Lab Based on Component

WANG Jian-xin, ZHANG Li-yuan, SHENG Yu, LIU Li-juan
(School of Information Science and Engineering, Central South University, Changsha 410083, China)

Abstract: *A design model and implementing method of PCOVL (Principles of Computer Organization virtual lab) was proposed based on component techniques* .This system was implemented with Java language, which could make the virtual lab system be independent of operating system. In the PCOVL, chip instruments were developed with JavaBean component techniques, and the attribution and behavior of component could be adjusted dynamically by using Java Reflection. *A trigger-dispatching mechanism based on data-driven was presented* to effectively solve problems of data transfer and component-dispatch between components with complex relationship, which was implemented by data-driven principle, wait-notify of Java multithreading schemes, and the Lock technique to control multithreading's synchronization. In the PCOVL, functions of designing process of experiment visually, changing the status of component randomly at run time, displaying results of experiment dynamic and real-time were realized. PCOVL provides powerful tech support and theory foundation for developing virtual lab of computer hardware courses.

Key words: virtual lab; PCO; java bean; data-driven; multithreading

引　言

当前，远程教学作为一种新型的教育模式已成为各国教育界的重要研究课题[1]，基于 Internet 的虚拟实验环境是现代远程教学质量提高的关键因素。虚拟实验环境为学生进行实验提供了一种崭新的途径，学生可以通过网络在电脑上模拟各种实验，通过这种方式既可以节约经费，又可以不受时间和空间的限制[2-3]。

目前，虚拟实验室的系统设计方案从实现技术的角度可分为纯软件方式和软硬件结合的方式。以纯软件方式开发的虚拟实验室大多是采用了现今流行的 WWW 技术，包括 HTML、CGI、Java Applet、Java Servlet 等技术，它是目前计算机仿真研究中的重要工作之一，并且在很多领域中产生了很好的应用。如客户端采用 Java Applet，服务器端采用 CGI、Java Servlet 开发的远程编程虚拟实验室[4]，采用 LabView 平台开发服务组件的基于 Web 的分布式虚拟教育实验室[5]，利用 Java 语言和数字信号处理技术相结合开发的仿真系统 J-DSP[6-7]，采用 CORBA 技术实现的基于 Web 的虚拟实验室体系结构[8]，以组件技术构架、并通过 CORBA 技术实现 Java 与 Matlab 之间的无缝联接的基于 Internet 的数字图像处理仿真系统模型[9]，基于 Internet 的自动控制理论远程实验室[10]，基于组件的数字信号处理虚拟实验室系统[11]，以及基于组件的实时入侵检测虚拟实验室系统[12]等。

计算机组成原理课程是计算机专业一门重要的专业基础课，它主要学习的是计算机系统内部相应硬件间的工作原理。计算机组成原理的实验是该课程学习的一个重要环节，学习者通过实验，体会计算机每一个算术计算的全过程，了解计算机的信息存储以及逻辑运算，掌握计算机硬件组成与设计、制造、调试和运行维护等多方面的技能。但是由于硬件动作的不可见性，学生做实验时，不清楚硬件动作的执行流程和单元间的数据流动，不能检测并查看单元的选通信号

收稿日期：2007-01-24　　**修回日期：**2007-07-12
基金项目：国家自然科学基金(60673164)；湖南省杰出青年基金(06JJ10009)；新世纪优秀人才支持计划(NCET-05-0683)；长江学者和创新团队发展计划(IRT0661)。
作者简介：王建新(1969-)，男，湖南邵东人，教授，博士，研究方向为计算机网络优化算法、虚拟实验环境；**张丽媛**(1983-)，女，云南保山人，硕士生，研究方向为虚拟实验环境；**盛羽**(1977-)，男，湖南长沙人，硕士，研究方向为虚拟实验；**刘丽娟**(1981-)，女，湖南长沙人，硕士，研究方向为虚拟实验。

和读写时序，实验透明度不高，成功率低。另外，实验仪器元器件易于老化和损坏，实验成本和费用高，导致实验环境十分缺少，迫切需要一个易于操作、功能完善、交互性强、高度仿真的可视化计算机组成原理实验学习平台。

因此，本文提出了采用纯软件技术开发的基于组件的计算机组成原理虚拟实验室(PCOVL)。PCOVL 将各种常见的集成电路芯片封装成组件的形式，实验者通过选取组件、搭建实验流程、运行并查看实验结果的仿真实验过程，验证单个芯片的功能或进行组合逻辑测试，了解计算机的信息存储以及逻辑运算的原理。实验过程中用户也可以随机改变组件状态，实时的查看结果，具有很好的随机性、动态性和实时性。

1 系统架构

系统架构如图1所示。Web服务器端提供已编译好的Java类文件，并且根据用户的请求将JavaBean组件和子类传给客户端。虚拟实验中所需的元件设备以JavaBean组件的方式封装，存于服务器端的实验组件库中，服务器端的注册配置文件库用于存储JavaBean组件注册的XML文件，实验配置文件库用于存储经典实验的XML文件。

客户端主要提供虚拟实验室组件的实现、实验的运行和结果的生成。客户端采用浏览器中嵌入Java Applet的方式，用户可以通过客户端提供的菜单栏、工具条、组件注册栏、仿真实验台、组件描述栏等，动态地定制实验流程、实时地保存实验、随机地运行实验和查看图形化的实验结果.使得系统具有良好的交互性、实时性和灵活性。

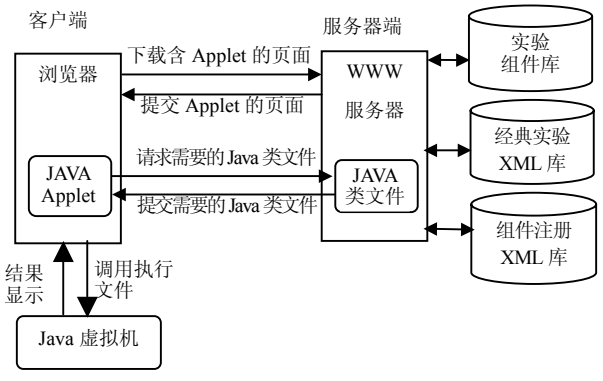


图 1 PCOVL 系统架构

2 系统的功能模块

PCOVL是一个仿真计算机硬件实验的虚拟实验平台，它将虚拟集成电路芯片通过虚拟线路组成相关的实验，具有可视化、全交互、仿真程度高等特点，其中每一个可视化的二维物体代表一个实验对象，用户通过鼠标的点击以及拖拽，完成选定对象、连线等操作，开展各种实验，观察实验的结果。同时，除了规定的实验之外，还允许用户自己设计，进行自主实验，提高用户的动手能力和创新能力。

PCOVL主要由数据生成、仿真运行和输出显示三大模块构成，系统模块如图2所示。数据生成模块提供了实验流程运行所需的数据源信号；仿真运行模块是所有组件的调度管理中心，负责调度控制组件的执行，运行用户定制的实验流程，并把产生的结果传递给输出显示模块。它是系统的核心部分，主要包括组件关联、组件间的数据传递和调度控制管理三个模块；输出显示模块将仿真结果以图形方式动态输出，实时的显示给用户。

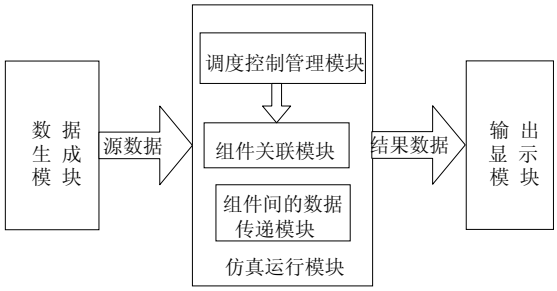


图 2 PCOVL 系统模块

整个系统的处理过程如下：用户通过设置数据源组件的属性，产生数据源信号。仿真运行模块接收数据生成模块产生的数据后，由调度控制管理模块控制组件的调度过程。它根据数据在实验流程中的传递次序，通过组件关联模块找到与数据源组件相连的一级组件，动态调用一级组件的功能方法并得到结果数据。然后通过组件间的数据传递模块和组件关联模块把结果送给一级组件相连的后继组件（即二级组件），并驱动后继组件运行。后继组件同样采用一级组件的信息交互过程，驱动三级组件，如此反复，直到输出显示模块中的终端显示组件接收结果数据并以图形的方式形象地将结果显示给用户。

3 PCOVL 数据的生成与显示

3.1 数据生成模块

数据生成模块负责产生仿真实验所需的数据源，驱动整个实验流程的运行。数据的产生主要有两种方式，一种是开关组件产生的稳态数据；另一种是单脉冲或时序脉冲组件产生的跳变数据。该模块的实现存在以下难点：用户在运行期间可以随机改变开关组件的状态，产生新的源数据，如何实时的响应开关状态的改变；脉冲组件具有瞬时性，在很短的时间内产生值的跳变，引起其他组件状态的改变，如何实现瞬时性；时序脉冲组件每隔固定的时间间隔就产生一个脉冲信号，如何周期性的响应时序脉冲信号是难点。

开关组件具有闭合和打开两种状态，闭合状态对应输出0信号，打开状态对应输出1信号。实验运行过程中，用户可以通过双击开关组件图标任意切换其闭合和打开状态，系统采用鼠标监听响应事件mouseClicked和重画实现组件的动态改变和实时响应。一旦用户产生了动作，其状态一定与原状态相反。因此，只要系统监听到了鼠标响应事件，就可以通过鼠标的选定区域找到对应的触发组件，然后利用反射技

术获取它当前的状态和方法，把状态属性设置成与其相反的状态，并把新的状态图片重画在原来的位置上，接着激活（invoke）相应的方法，产生新的输出。主要代码如下：

```
selfClass=Class.forName(this.carrier.getInstance().getClass().getName();
stopClass = selfClass.getSuperclass();
BeanInfo beanInfo = Introspector.getBeanInfo(selfClass, stopClass);
switchMethod = beanInfo.getMethodDescriptors(); //得到其方法集合
if (switchMethod[0].getMethod().getName().equals("SWITCHOFF")) {
    SwitchOff = switchMethod[0].getMethod();
    SwitchOn = switchMethod[1].getMethod(); }
else {
    SwitchOff = switchMethod[1].getMethod();
    SwitchOn = switchMethod[0].getMethod(); }
if (this.selectedCarrier.getChangeName().equals("SWITCHOFF")) {
    SwitchOff.invoke(this.carrier.getInstance(), null); / /输出为 0
}else {… …..}
```

单脉冲信号的动态瞬时产生同样采用鼠标监听响应和重画的事件过程来实现。单脉冲组件产生瞬时的输出值，默认状态为暗。当用户双击单脉冲组件时，系统监听到该事件，把组件由暗转为亮状态，对应输出1信号，并把输出结果通知给下级关联组件；接着重新变为暗的状态，对应输出0信号，再次通知下级组件。

时序脉冲组件比较复杂，它是一个周期变化、等间隔产生瞬时输出信号的过程。系统采用定时器Timer来实现，在Timer定时器产生的一个等间隔周期脉冲中，首先把组件图片显示为亮的状态，然后调用SequencePulseEvent()方法完成时序脉冲先后产生1、0输出信号的过程。该方法在产生输出前先判断时序脉冲的下级组件是否满足处理新数据的条件，如果满足，则产生输出信号，并通知下级设备；否则随机等待一段时间后再作判断，最后重画组件暗的状态。一旦在定时器中定义的脉冲个数响应完毕后，定时器将结束工作，停止产生脉冲。它的主要实现代码如下：

```
PulseTimer=new Timer(SequenceTimer.DelayTime，new ActionListener(){
    public void actionPerformed(ActionEvent actionEvent) {
        if( SequencePulseTime <=SequenceTimer.TotalTime) {
            synchronized (SequenceCarrier.getInstance()) {
            SequencePulseTime++；//记录产生的脉冲个数
            drawSequencePulse(true);//画出变红的图片，产生一个脉冲
            SequencePulseEvent();
            .drawSequencePulse(false);//画出变暗的图片表示脉冲结束
          }
      }else   PulseTimer.stop();
     }}
  PulseTimer.start();
}
```

3.2 输出显示模块

输出显示模块主要负责接收数据，并将结果映射成图形的方式直观显示出来，对数据不会进行任何加工处理。因此，数据终端显示组件只定义属性的读写方法，而没有功能方法。显示组件包括虚拟信号灯和数码显示管。

计算机组成原理实验中最终的数据都是二进制 0、1 代码，对应于显示组件虚拟信号灯的灭、亮两种状态。对于虚拟信号灯终端组件的实时响应过程，主要是在系统运行过程中根据终端组件的输入信号来动作。如果信号为 1，就重画显示亮的图片，并把状态属性设置为亮；反之，重画显示灭的图片，其实现过程与开关组件类似。数码显示管的实现不同之处在于它会根据输入端的值分别判断组件的各部分处于亮或灭状态。

4 PCOVL 中组件的运行与控制

PCO的实验流程能够分解成若干功能相对简单而集中的元件，当满足一定条件时，各个元件完成各自的职责，并启动下一个元件接替工作下去。问题的关键在于如何使原本相互之间毫无关系的一堆组件成为一个能够协调作战的有机整体。PCOVL引入多线程技术，采用基于数据驱动的触发式调度机制来控制组件线程的执行，提高了系统的并行性和实时性，解决了组件线程的协作和同步问题。

4.1 仿真运行模块

仿真运行模块的功能是采用合理的调度机制来控制组件执行顺序，把一个源组件上发生的状态变化情况通知与它关联的下级组件，使得各个组件之间的行为有一定的因果关系，控制整个实验流程功能的实现。该部分由组件关联模块、组件间的数据传递模块和调度控制管理模块三个部分组成。

(1) 组件关联模块

PCOVL的大多数组件都是多输入多输出接口，导致了组件的连接关系复杂。PCOVL中采用组件自我记录的方式，每个组件拥有一张连接关系表，定义为*ConnectTable*类，一般组件中用于记录其所有输出引脚关联的下级组件信息。由于每个组件的输出引脚个数不等，每个引脚连接的组件不确定，所以采用初始大小为5的二维可变数组表示，数组的元素是一个二元组（关联组件，引脚）。其中，表的一维代表组件的输出引脚，二维代表组件的关联信息。*ConnectTable*类定义了addElement，removeElement，nextDeviceLead三个方法，用于为指定的引脚新增连接信息，删除连接信息和获得其所有关联组件信息，使表易于扩充和维护。图5所示实验中的组件74LS181（右）的连接关系如表1所示。

表 1 组件 74LS181(右)的连接关系表

输出引脚	一维 存储位置	二维 关联信息
15	0	74LS181(左), 13
16	1	74LS245, 0
17	2	74LS245, 1
18	3	74LS245, 2
19	4	74LS245, 3

对于含有双向输入输出引脚的组件（如存储器组件），

其连接关系表稍有不同，它的引脚既可作为输入又可作为输出，组件根据条件的不同，动态的调整它的引脚属性来实现输入、输出角色的转换。因此，它的连接关系表除了记录输出引脚关联的下级组件信息外，还要记录输入引脚关联的组件信息。

(2) 组件间数据的传递模块

由于组件的连接关系复杂，耦合程度高，组件之间的数据传递不能简单的概括为并行或串行，数据传递次序的正确与否直接影响实验结果的正确性，因此，如何很好的实现组件间数据的传递是该模块的重点。

PCOVL采用直接传递方式实现组件间数据的传递，即数据由上级组件直接赋值给与它关联的下级组件，不依赖第三者存储数据。由于组件的输入引脚值是共享资源，上级组件对它进行写操作，而组件本身进行读操作。因此，数据的直接传递会导致数据的丢失，即下级组件输入引脚的旧数据还未及时处理就被新的数据覆盖。为了保证数据的同步和一致，系统采用引脚标志位判别法来实现，只有下级组件读取数据后，上级组件才能赋新值。该模块为每个输入引脚设置标志位*dataflag*，运行时，当上级组件准备生成新的输出时，先调用连接关系表的nextDeviceLead方法获得与它关联的所有下级组件，判断它们的输入引脚的标志位。当*dataflag*=0，表示旧的数据已经取走，上级组件可以执行其功能方法生成新的输出，并把输出引脚值直接赋给相连的下级组件的输入引脚上；反之旧的数据未处理，上级组件必须随机等待一段时间，再进行判断，直到下级组件执行读操作，把该标志位置为0。通过这种方式，保证了数据的有序和一致。主要的实现代码如下：

```
for (int i = this.carrier.getMethodOutNum();
    i < this.carrier.getConnectRelation().getsize(); i ++) {
    String ilead = new Integer(i - LeadINNumber).toString();
    Next = this.carrier.getConnectRelation().nextDeviceLead(ilead);
    int number = 0; //代表当前引脚所连接的下级设备数据取走的个数
    while (number < Next.length) {
        number = 0;
        for (int j = 0; j < Next.length; j++) {
            if ( (Next[j].GetDevice().getLead(Next[j].GetLead().name)
            .dataflag = = 1) {
                try {   //引脚上的数据没有取走,等待一段时间
                this.selectedCarrier.getInstance().wait(100);
                }catch (InterruptedException ex) {
                 ex.printStackTrace();}
            }else {
                number++; }
     }}
```

(3) 调度控制管理模块

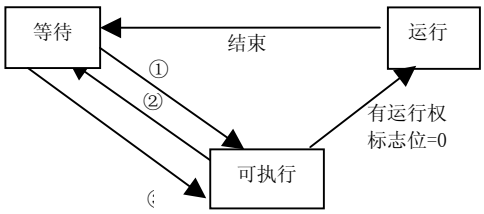
一个实验流程可以看作是由多个功能独立的虚拟元件通过数据流通道互联构成，每个组件带有若干输入口和输出口，组件的功能是将其输入口的数据进行变换后形成数据流从输出口传递给与之关联的其他组件。由于组件类型繁多，组件之间连接关系复杂，一个组件的输出数据会影响其他多个组件。另外，特殊组件（如存储器组件）的运行具有双向性，可以通过动态改变引脚的属性来完成读写功能。如何协调控制各个组件的执行次序形成有机整体，并实现系统高度的并行性和实时性是该模块设计的重点和难点。调度控制管理模块主要是采用基于数据驱动的触发式调度机制来控制组件的执行，并建立起各个独立组件之间的因果关联，使它们形成一个能够协调工作的有机整体。

数据驱动原理是数据流机实现的一种方式，只要数据不相关和资源可以利用，就可以最大限度地开发出计算的并行性[13]。基于数据驱动的触发式调度机制是把每个组件都作为一个线程，线程之间的通信采用wait()、notify()机制和锁机制来控制，线程之间的数据传递采用直接传递方式，并通过数据传递模块来控制对数据的处理。初始化时，每个组件会根据实验流程的拓扑关系，建立自己的连接关系表。在运行过程中，存储在其中的引脚数据会随着组件功能方法的执行而动态更新。组件产生的结果数据会根据连接关系表中的连接关系从输出引脚传送到与它相连的下一级组件的输入引脚上。这种调度机制有效地保证了数据源不断产生的信号都能被及时接收和处理的实时性。

具体过程如下：遍历实验流程中所有的组件，把每个组件构造成一个线程对象放到运行队列线程组中。初始状态，所有组件的引脚值都为空，标志位*dataflag*为0。当用户点击运行按钮，所有线程以等概率的机会抢占CPU得到运行权，除了数据源线程组件外，其他线程运行时都会先调用wait()方法使其处于等待状态而迅速放弃CPU的使用权，直到其上级线程调用notify()方法唤醒并驱动它们运行。按照这种调用过程，当数据源组件得到运行权后，它将根据连接关系表查找与它相连的所有下级组件，依次检查这些组件的输入引脚的标志位是否全为0。如果满足条件，它先通过JavaBean的反射技术动态的获得组件的功能方法，并invoke该方法产生输出信号，然后根据连接关系表依次把输出值赋值给与它相连的下一级等待线程组件的输入引脚上，并唤醒它们，之后自己重新进入等待状态；否则，它将随机等待一段时间，直到下次运行权的获得。下级等待线程被唤醒后便进入可执行状态，并可以再次竞争CPU从而获得运行权。下级线程组件拥有运行权后，进行与数据源组件相同的上述动作。组件的状态转换过程如图3所示。

可见，下级组件的执行是由上级组件对其输入引脚赋值来唤醒驱动的，它是被动执行的过程，当新的数据没有来到时，组件只能等待，即组件的操作执行是由“数据驱动”的。所有的线程都会一级一级地按照这种数据驱动方式触发调度，直至终端响应线程虚拟信号灯的运行。当用户点击停止按钮后，整个线程组将被中断，从而结束实验的运行，调度过程如图4所示。对于双向输入输出组件，运行期间其引脚的输入、输出角色会改变，用Direction属性表示其运行方向，

一般为正向，引脚角色互换后变为逆向。运行时，通过反射技术找到该属性，并调用getWriteMethod()和getReadMethod()方法动态的设置、获得属性，从而根据实时的属性值，进行相应的操作。



① 被赋予新的数据并唤醒；
② 有运行权，但标志位=1,Wait (n) 微秒；
③ 过了 n 微秒

图 3　组件的状态转换图

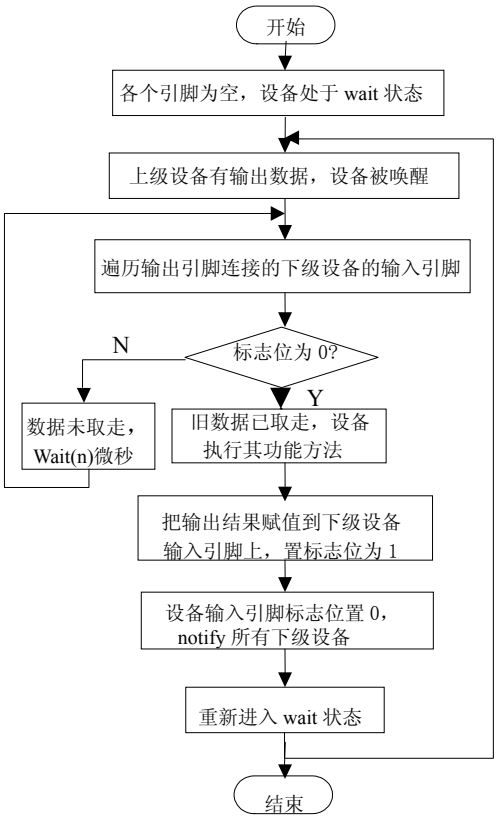


图 4　组件的调度流程图

该模块采用多线程机制大大提高了系统的并行性；采用线程组机构可以对多线程集中管理；数据驱动方式实现了选择性追踪，即在运行过程中不是对所有组件进行计算，而是只选择输入端信号有变化的组件进行计算，提高了运行效率；采用Java 语言的反射技术，能够根据设定的对象方法名和属性名调用对象方法和设置对象的属性值，使得用户能够在运行时与组件对象交互，使得仿真系统具有良好的实时性和交互性。

4.2 多线程机制在 PCOVL 中的应用

多线程机制是实现仿真运行模块的关键技术。采用多线程机制即可以实现实验流程的连续运行，又可以实时响应用户的操作。硬件实验中可以实现各个元件的并发执行，而软件模拟过程中，系统只有一个CPU，多个线程真正的“同时”执行是不可能的，线程使用单CPU必须轮流执行，但是，这发生的非常快，可以认为各个线程同时执行。其中，线程的同步和交互问题需要重点考虑。

系统运行期间，各个线程的控制流彼此独立，使得各个线程之间的代码乱序执行。当多个线程并行工作时，可能有2个以上的组件线程试图同时访问某个组件的引脚数据，系统使用线程的同步控制对这种潜在资源冲突进行预防。JAVA 的Object类提供了三个方法：wait()、notify()和notifyAll()用于协调多个线程对共享数据的存取，这些方法必须放到Synchronized同步控制块中，调用它们的线程在调用这些方法前必须“拥有”对象的锁，这就能保证如果两个线程试图在同一对象上调用这些方法时不会互相冲突。系统采用wait、notify方法对运行的多个线程进行系统级的协调和调度，实现它们的交互。一旦下级组件线程调用了wait方法，它便释放对象锁，进入等待模式，停止线程的执行，直到被上级组件调用同一个组件对象的notify方法所唤醒，使得原线程再次成为可运行的线程，与其他线程展开竞争，以争取进入该对象，从上次因调用wait()而中断的地方开始继续运行。下面给出了组件进入等待、唤醒状态的主要代码。

```
synchronized (this.carrier.getInstance()) {//获得对象的锁
    try {
        this.carrier.getInstance().wait();//组件线程进入等待状态
    }catch (InterruptedException ex) {}
}
synchronized(Next[m].GetDevice().getInstance()){//获得对象的锁
    Next[m].GetDevice().getInstance().notify();//唤醒下级组件线程
}
```

通过类锁机制synchronized (DeviceCarrier.class) 保证组件原子操作的顺利执行，避免在运行过程中被其它得到运行权的组件线程中断，修改引脚的情况。组件的原子操作是指从组件执行其功能方法生成输出到notify所有下级关联组件的整个过程不能被中断。类锁机制的原理就是让每个组件线程共用同一把类锁，谁获得了这把锁就能优先获得运行权，执行它的原子操作，没有得到锁的其它线程只能继续留在等待队列中等待。当组件的原子操作完成后释放锁，等待线程才能重新去竞争这把锁。这样使得原子操作被完全执行，保证共享数据的一致和有效。

5　系统仿真实例

PCOVL中开发了许多的组件，如基本门电路（与、或、非、异或门等），带公共时钟复位八D触发器74LS273，算术逻辑单元/函数发生器74LS181，三八译码器74LS138，八同相三态总线收发器74LS245，四位二进制同步计数器74LS161，静态随机存储器6116，可编程存储器28C16等。

用户在系统中能够完成基本的计算机组成原理仿真实验。

图5给出了计算机组成原理虚拟实验室系统的用户界面，左边是组件栏，右边是仿真实验台。以算术/逻辑运算实验为例，用户首先从左边的组件列表中选择两片“74LS181”，两片“74LS273”，一片“74LS245”，两个单脉冲，开关若干，灯泡若干，把它们拖到设计面板中，并进行连线，搭建好的实验流程图如图5所示。左右两个“74LS273”组件作为锁存器用于保存参与运算的数据B和A，两个单脉冲组件分别控制数据B和A的同步输入。左右两个“74LS181”组件分别完成数据B和A的高四位和低四位的算术逻辑运算。“74LS181”的S0-S3、M控制使能端接到开关上控制进行的算术还是逻辑运算。高位的“74LS181”组件的13号引脚CN与低位的“74LS181”的15号进位输出相连。“74LS245”组件的输出引脚接到灯泡，作为运算器结果的显示。如图所示，数据A的值为01000101，数据B的值为00000011，此时，CN=1，M=0，S0-S3为1001，根据“74LS181”的功能表可知，此时实现A加B的运算，灯泡显示结果为01001000，结果如图6所示。

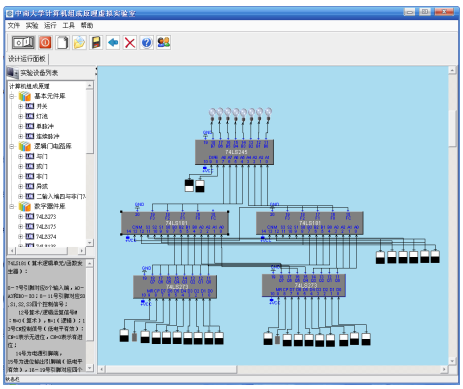


图 5 算术/逻辑运算实验的仿真实验图

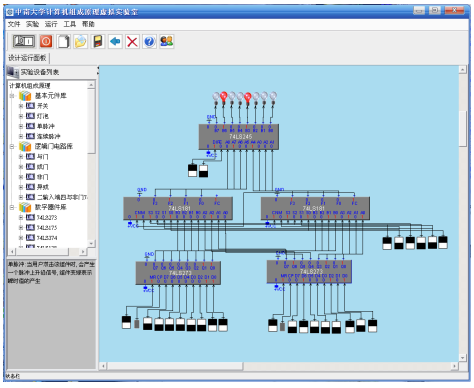


图 6 算术/逻辑运算实验的结果显示图

6 结论

本文在对计算机组成原理实验的深入分析研究的基础上，详细论述了计算机组成原理虚拟实验室的架构和实现方案。该系统以组件形式开发了大量的计算机组成原理实验设备，使得系统易于维护和扩充；基于数据驱动的触发式调度机制能够很好的仿真具有随机、动态变化、并发等特点的实验流程，大大提高了系统的并行性和运行效率；以图形化的方式实现数据的生成和结果的显示，增加了系统的可视化效果。在此系统中，用户能够进行计算机组成原理课程中的相关实验，也可以自主开发元器件扩充实验内容。该系统具有功能完整、交互性强、功能模块化清晰、实验仪器丰富且易于扩充、调度机制兼容性强等特点，能够为开发计算机硬件核心课程虚拟实验室（如数字电路，电路基础等）提供了很好的参考模型。

参考文献：

[1] Khalifa M, Lam R. Web-Based Learning: Effects on Learning Process and Outcome [J]. IEEE Transactions on Education (S0018-9359), 2002, 45(4): 350-356.

[2] Wang Jianxin, Chen Songqiao, Jia Weijia, *et al*. The Design and Implementation of Virtual Laboratory Platform in Internet [C]// Proceedings of The First International Conference on Web-based Learning, 2002. London, UK: Springer-Verlag, 2002: 169-177.

[3] 李仁发, 周祖德, 李方敏, 等. 虚拟实验室网络体系结构的研究[J]. 系统仿真学报, 2002, 14(3): 359-362, 393. (Li Renfa, Zhou Zude, Li Fangmin, *et al*. Network Architecture of Virtual Lab [J]. Journal of System Simulation, 2002, 14(3): 359-362, 393.)

[4] Cao Jiannong, Chan Alvin, Cao Weidong, *et al*. Virtual Programming Lab for Online Distance Learning LNCS 2436 [C]// First International Conference, ICWL 2002. London, UK: Springer-Verlag, 2002: 59-61.

[5] Benetazzo L, Bertocco M, Ferraris F, *et al*. A Web-Based Distributed Virtual Educational Laboratory [J]. IEEE Transaction on Instrumentation and Measurement (S0018-9456), 2000, 49(2): 349-356.

[6] Spanias Andreas, Atti Venkatraman. Interactive Online Undergraduate Laboratories using J-DSP [J]. IEEE Transactions on Education (S0018-9359), 2005, 48(4): 735-749.

[7] Spanias Andreas,Atti Venkatraman Papandreou-Suppappola, Antonia, *et al*. On-line Signal Processing using J-DSP [J]. IEEE Signal Processing Letters (S1070-9908), 2004, 11(10): 821-825.

[8] Wang Jianxin, Lu Weini, Jia Weijia. A Web-Based Environment for Virtual Laboratory with CORBA Technology [J]. International Journal of Computer Processing of Oriental Languages (S0219-4279), 2003, 16(4): 261-274.

[9] 王建新, 陆炜妮, 王伟平. 基于组件的数字图像处理仿真系统的设计与实现[J]. 系统仿真学报, 2004, 16(6): 1213-1216. (Wang Jianxin, Lu Weini, Wang Weiping. A Component-based Simulation System for Digital Image Processing [J]. Journal of System Simulation, 2004, 16(6): 1213-1216.)

[10] Marco Casini, Domenico Prattichizzo, Antonio Vicino. The AutomaticControl Telelab: A User-Friendly Interface for Distance Learning [J]. IEEE Transactions on Education (S0018-9359), 2003, 46(2): 252-257.

[11] Wang Jianxin, Liu Lijuan, Jia Weijia. The Design and Implementation of Digital Signal Processing Virtual Lab Based on Components LNCS 3583 [C]// 4th International Conference, ICWL 2005. Germany: Springer-Verlag Berlin Heidelberg, 2005: 291-301.

[12] 王建新, 安莹, 吴国政, 盛羽. 基于组件的实时入侵检测虚拟实验室的设计与实现[J]. 系统仿真学报, 2006, 18(11): 3283-3286, 3296. (Wang Jianxin, An Ying, Wu Guozheng, *et al*. Design and Implementation of Real-time IDS Virtual Lab Based on Component [J]. Journal of System Simulation, 2006, 18(11): 3283-3286, 3296.)

[13] 陈章. 基于数据驱动的构件服务软件框架研究[J]. 计算机工程与应用, 2005, 41(18): 39-41, 105.