DS-VLAB 源程序说明

一、技术路线

多思计算机组成原理虚拟实验系统是一种基于浏览器客户端技术的网络教学软件，该系统完全采用浏览器客户端技术，用 JAVASCRIPT+JQUERY+JQUERYUI+CSS+VML 编程实现，在 IE10 中测试通过。

系统使用 JAVASCRIPT 开发组件库，利用 CSS 绘制芯片等组件，采用 VML 绘制组件间的连接线，用 HTML5 File API 实现电路文件的导入导出。使用基于队列的单线程组件调度算法，解决众多组件之间信息传递和调度运行的问题。系统从功能上对组件进行仿真，支持电路设计，可进行复杂模型机等设计性实验。

目前，实现网络虚拟实验室的技术主要有 Java、Flash、VRML 等，与基于这些技术的虚拟实验系统相比，本系统的优点有：结构简单，无需安装插件，既能以 B/S 模式运行，也可不加修改直接以单机方式运行，能够非常方便地整合到其它网络综合实验平台中。

二、系统架构

系统架构如图 B.1 所示。客户端通过浏览器向 WEB 服务器提出页面请求，WEB 服务器响应请求，找到所请求的页面，并将此页面及其引用的 JavaScript 脚本代码和 CSS 样式表作为响应内容，发送回客户端，客户端浏览器打开发回的页面文件、解释并执行 JavaScript 代码。

客户端承担了仿真实验室运行的全部任务，包括操作界面的显示、电路的连接与绘制、电路的运行等等。由于 JavaScript 脚本是由客户端解释执行，不占用服务器资源，从而大大减轻了服务器的压力、提高了页面反应速度。

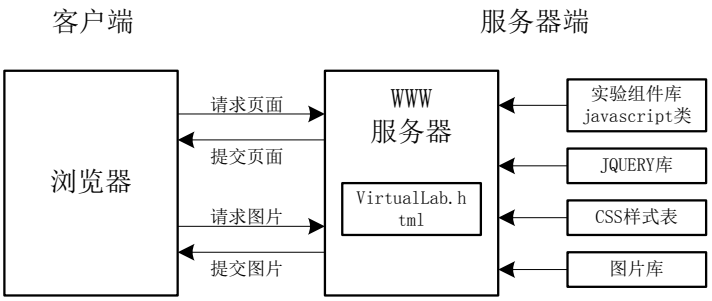

图 B.1 系统架构

三、系统主要模块

系统主要包括四大部分：组件库、电路绘制、组件调度、文件操作。

组件库是实验组件的类库，包括了所有功能器件的类函数，描述了每个器件的大小、引脚个数和名称等属性，实现了器件的功能函数。组件库供电路绘制、组件调度等模块调用。

电路绘制模块的功能是在工作区绘制芯片、门电路等组件以及组件间的连接线。当用户从实验设备工具箱中拖出组件、在组件间拉线，或导入电路文件时，系统会自动调用电路绘制模块画出相应的器件和线。

组件调度模块是所有组件的运行调度中心，负责调度和控制组件的工作顺序，在组件之间传递信息，控制信息的流动，最终得到电路运行结果。

文件操作模块主要提供实验电路的新建、导入和导出等功能。

四、源程序目录结构

源程序的目录结构如图 B.2 所示：

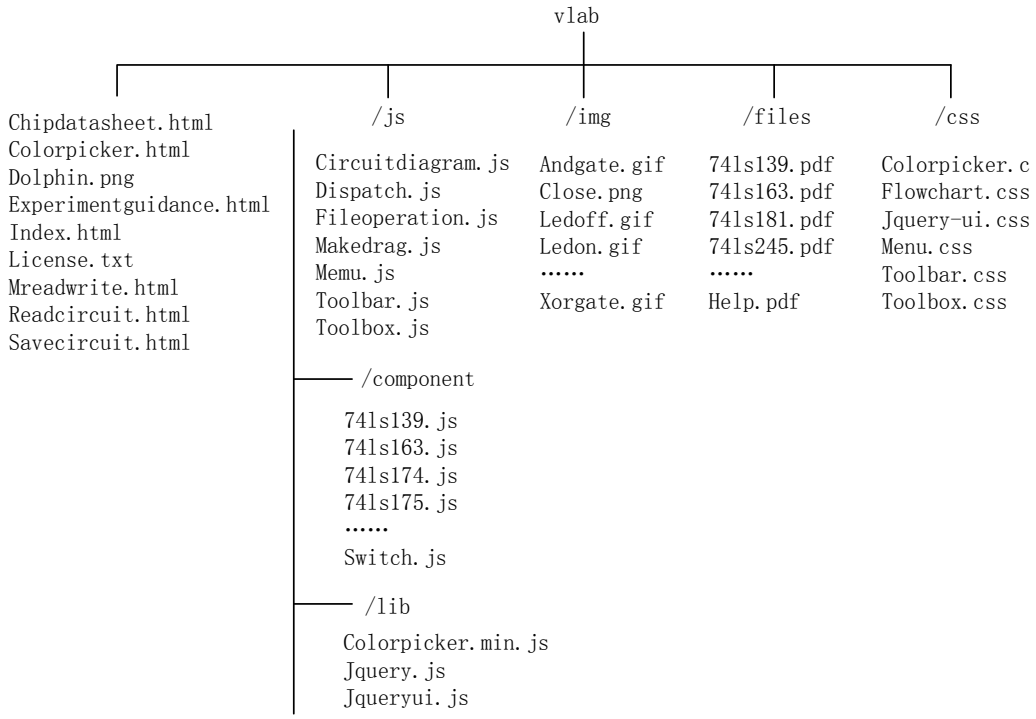

图 B.2 组件调度流程图

源程序文件夹 vlab 中一共有 9 个文件和 4 个子文件夹。9 个文件说明如下：

1） Chipdatasheet.html：芯片数据手册下载网页
2） Colorpicker.html：连接线颜色选择网页
3） Dolphin.png：网页标题图标
4） Experimentguidance.html：实验指导书下载网页
5） Index.html：网站首页，系统主界面
6） License.txt：版权许可说明
7） Mreadwrite.html：内存芯片读写网页
8） Readcircuit.html：打开文件网页
9） Savecircuit.html：保存文件网页

4个子文件夹分别为js、img、files、css。其中，js文件夹存放程序代码，img文件夹存放图片，files文件夹存放数据手册等下载资料，css文件夹存放样式表。

文件夹js是最重要的文件夹，所有javascript代码都存放在这个文件夹中，包括7个文件和2个子文件夹，7个文件说明如下：

1） Circuitdiagram.js：电路图绘制代码文件，包括组件和连接线的绘制、移动、删除、事件处理等函数代码。

2） Dispatch.js：组件调度代码文件。

3） Fileoperation.js：文件操作代码文件，包括电路图文件的导入、导出等函数。

4） Makedrag.js：拖动代码文件，包含了使对象能够被拖动的函数。

5） Memu.js：菜单代码文件，包含了菜单初始化的相关代码。

6） Toolbar.js：工具栏代码文件，包含了工具栏初始化的相关代码。

7） Toolbox.js：工具箱代码文件，包含了工具箱初始化的相关代码。

2个文件夹分别为component和lib，component文件夹是组件仓库，里面存放了所有器件的类代码，每种器件对应一个文件；lib文件夹存放的是JQUERY等JAVASCRIPT库文件。

以上文件夹和文件共同组成了源程序的整个目录结构，是源程序的全部内容。

五、系统的设计与实现

1. 组件建模

电路建模具有层次性，根据层次的不同可将元件模型分为晶体管级和行为级。行为级模型是根据元件的传递函数或者输入/输出特性来构造模型，优点是在较少牺牲精度的前提下降低了实现难度，同时保证模型实用性。本系统采用了基于行为级模型的建模方式。

实验组件可分为三种：源器件、中间器件和终端器件。源器件产生驱动整个电路运行的源数据，包括开关、单脉冲、连续脉冲等。中间器件接收输入信号，经处理后输出结果信号，包括 ALU 芯片、RAM 芯片、门电路等。终端器件用于显示结果，只有输入引脚没有输出引脚，如小灯。

组件库设计的主要内容是：组件类的设计和定义，类定义必须满足组件的绘制、运行与调度功能需要，为组件的绘制、运行与调度提供所有必须的属性和方法；源器件产生源数据的方法，主要是开关和脉冲的实现方法。

组件库由所有实验组件的 JavaScript 类组成，组件类描述组件的外形属性、电气属性和功能方法。例如，ALU 芯片 74LS181 的属性和方法定义如下：

```
function Compo74LS181() {
    this.id; //芯片唯一编号
    this.name ; //芯片名称
    this.width ; //芯片宽度
    this.height ; //芯片高度
    this.paddingLR ;  //芯片左右边距
    this.pinName; //每个引脚的名称
    this.pinWidth; //引脚宽度
```

```
        this.pinHeight; //引脚高度
        this.pinPosition; //每个引脚在芯片上的坐标
        this.pinFunction; /*每个引脚的类型  0:输入 10:必要输入 1:输出 2:地 3:电源
*/
        this.pinValue; /*每个引脚的值 0:低电平 1:高电平 2:悬空*/
        this.connection; //引脚之间的连接线
    };
    /*判断目前芯片是否已达到运算条件*/
    Compo74LS181.prototype.beReady = function () {
        for(var i=0;i<this.pinFunction.length;i++) {
            if (this.pinFunction[i] == 10 && this.pinValue[i] == 2) {
                return false;
            }
        };
        return true;
    };
    /*设置输入引脚的值，并判断芯片是否已达到运行条件*/
    Compo74LS181.prototype.input=function(pinNo,value){
        if (value == this.pinValue[pinNo]) return false;
        this.pinValue[pinNo] = Number(value);
        return this.beReady();
    };
    /*按照芯片功能，由输入引脚的值计算输出引脚的值*/
    Compo74LS181.prototype.work = function () {…};
```

开关组件有开、关两种状态，对应输出 0 和 1 两种电平信号。电源按下时，系统会自动搜寻到所有的开关组件，并按照开关状态向外输出电平信号。在实验运行过程中，用户可以通过单击开关组件使其闭合或打开，浏览器捕捉到鼠标单击事件后自动调用源器件响应函数，实现输出电平的转换和开关图标的切换。

脉冲组件使用 JavaScript 中的 Timer 定时器实现。用 setTimeout 函数设置电平跳变的时间间隔以及跳变时要调用的处理函数。脉冲组件启动之后，经过指定的时间间隔，会自动调用源器件响应函数，实现输出电平的转换，并且初始化下一个 timer、启动下一次的电平跳变。

2. **电路图绘制模块**

电路绘制模块的主要任务是根据鼠标拖放的位置，在实验电路区域显示组件和连接线，并为组件和连接线绑定鼠标事件的处理函数。

当实验者从工具箱拖拽某种组件到实验电路区域时，电路绘制模块会自动生成该组

件的一个对象，并按照类中给定的外形属性绘制其图标。当实验者将鼠标从一个引脚拖放到另一个引脚时，该模块会自动计算并绘制引脚间的连接折线，并把连接线信息保存到组件对象中。拖动组件时，该模块会修改组件的位置参数，实现组件的移动，同时重画与其相连的所有连接线。

该模块主要包括如下函数：

```
function Circuit() {
/*在指定位置(X,Y)处画出实验组件 c*/
this.addComponent=function(parentId,c,X,Y){…}
/*生成一条连接线*/
function lineCreate(paintDiv,X1,Y1,X2,Y2) {…}
/*生成连接线时，从起始引脚往目标引脚拖动的过程中，线的自动变化*/
function lineChange(line,X1,Y1,X2,Y2) {…}
/*生成连接线时，鼠标到达目标引脚后，调整定位连接线*/
Function lineAdjust(line,startPin,endPin) {…}
/*拖动组件 c 时，更新其所有连接线*/
 this.lineReplace=function(c) {…}
/*删除连接线*/
 function   lineDelete(line) {…}
/*删除组件 c 以及与组件 c 相连的所有连接线*/
 function   componentDelete(c) {…}
……
};
```

3. 组件调度模块

实验电路是由多个功能相对独立的组件组成，当一个组件运行时，该组件执行自己的功能方法，并将结果输出到下一级组件，下一级组件工作后再把结果输出到下下级组件，电平信号就这样在电路中传递和扩散，直到没有新的信号生成。设计的难点在于，怎样让众多组件协调有序地逐级工作。

此模块在实现上有 2 个选择，一是采用多线程技术，二是使用单线程方法。由于 JAVASCRIPT 是单线程的，因此本系统采用了单线程方法。

本模块使用队列实现组件调度。如图 B.3 所示，电源按钮按下时，遍历所有组件，将满足运行条件的组件入队，然后从队头取出第一个组件执行，按组件的输出结果依次修改与其相连的下级组件输入引脚的值，并判断下级组件是否达到运行条件，如果达到运行条件且不在队尾则入队，如此直到队列为空。

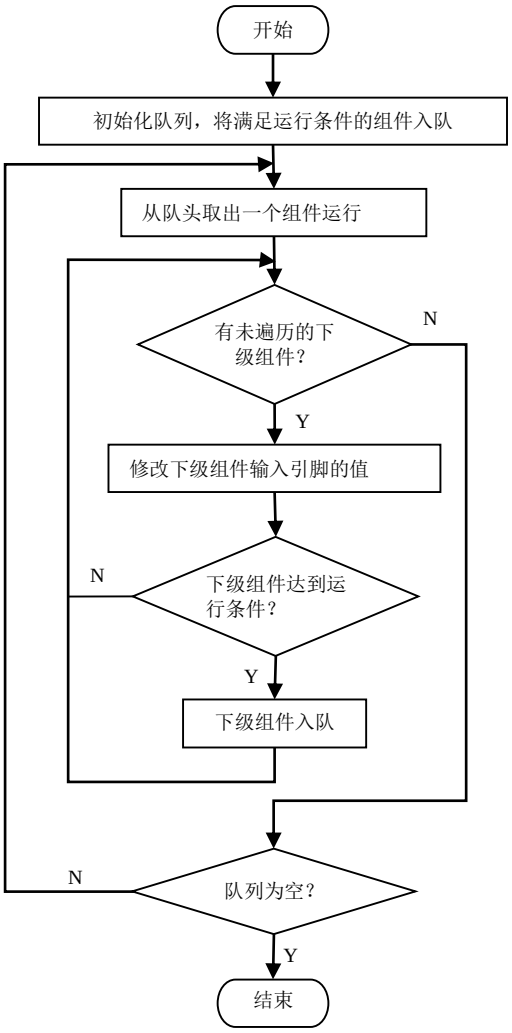

图 B.3 组件调度流程图

除了电源按钮的单击事件会启动组件调度过程以外，在实验进行中，源器件的鼠标单击事件或 Timer 事件也会触发组件调度过程。该模块的主要属性和方法定义如下：

```
function Dispatch(Circuit) {
//已达到运行条件、等待运行的组件队列
var workQueue = [];
/*初始化队列，将满足运行条件的组件入队*/
    this.initQueue = function () {…};
/*运行组件 c，修改其下级组件所有输入引脚值，并将达到运行条件的组件入队*/
    function cRun(c) {…};
/*依次运行队列中的所有组件*/
    this.runCircuit = function () {…};
/*源器件触发事件处理*/
    this.sourceTrigger = function (c) {…};
……
```

};

其中，源器件触发事件处理函数是当开关、脉冲等源器件被单击或 Timer 被触发时被系统自动调用的。此函数会更新源器件的值，并将其加入队列，然后启动 runCircuit 函数。

该算法简明有效，避免了多线程中读写共享资源的冲突，无需使用锁机制。

关于源程序的更详细信息请见源代码。