廣東工業大學

本科毕业设计（论文）

数字逻辑仿真实验系统

学　　院　计算机学院

专　　业　网络工程

（网络系统开发与管理)

年级班别　2012 级（1）班

学　　号　3112006332

学生姓名　李嘉豪

指导教师　张静

2016 年 6 月

数字逻辑仿真实验系统

李嘉豪

计算机学院

摘要

开发数字电路芯片是一个专业门槛很高的技术活，不仅需要学习大量的专业课程知识，还要熟练操作各种课程实验。这令不少初学者力不从心，尤其在课程实验的操作上更是如此。由于实验机箱和跳线的某些部分会因时间老化和过往使用者的严重错误操作而变得不能正常工作，初学者往往会浪费大量时间在排错上而不是验证课程实验上。为此，一个能够仿真课程实验的虚拟系统是必要的。通过这个虚拟系统，初学者就可以无视物理设备和跳线的问题而专心于实验中。

本文首先对数字逻辑课程实验机箱的各种实验元素进行功能划分和分析，并抽象出了门电路芯片模块、门电路端口模块、门电路连线模块、拓扑导入和导出模块、用户与拓扑交互模块和拓扑数据可视化模块这五个功能模块。然后论文讨论了如何使用Java 语言和一定的算法来编写这些模块以模拟实验机箱的功能元素。为了对门电路拓扑进行仿真，系统引入了逻辑表达式和端口真值信息分发机制来模拟芯片的计算和端口真值的传播。同时为了避免连线环路问题，系统还引入了端口连接信息分发机制和端口连接信息撤销机制来维护拓扑连接信息的一致性。

根据论文所设计的方案，该数字逻辑仿真实验系统被成功开发出来，并且很好地对数字逻辑课程实验进行了仿真。

关键词：仿真实验系统，数字逻辑，Java

Abstract

Development of digital circuit chip is an activity with a very high technical threshold because of a lot of professional courses knowledge and proficiency courses experiments, which makes lots of beginners powerless, especially in the operation of the experiment on the course. Since some parts of the experiment box and jumpers do not work because of aging and the serious error operation of previous users, beginners will always waste a lot of time on troubleshooting instead of on the experiment. Thus, a virtual system with simulation is necessary. Through this virtual system, beginners can ignore the trouble of physical device and jumpers and concentrate on experiments.

This paper will first divide and analyze all kind of experiments of digital logic course experiment box， and abstracted the five functional modules which are respectively gate circuit chip module, gate port module, gate circuit connection module, topology import and export module, users and topological interaction module and topology data visualization module. Then，this paper discusses how to use the Java language and some algorithms to write these modules in order to simulate the functional elements of the experiment box. In order to simulate gate circuit topology, system uses logical expression and port truth value information distribution mechanism to simulate the calculation and propagation of port true values. At the same time, in order to avoid the connection loop problem, the system also introduces the port connection information distribution mechanism and the port connection information revocation mechanism to maintain the consistency of topology information.

According to the design of the paper, the digital logic simulation experiment system was successfully developed, and the simulation of digital logic course was carried out.

Key words：Simulation experiment system, digital logic, Java

目　　录

1 绪论

1.1 课题研究背景与意义

《数字逻辑与系统设计》课程是计算机、通信、自动控制及信息等专业的重要专业基础课。核心内容包括数字逻辑及系统概述、组合逻辑电路分析、时序逻辑电路分析、数字系统设计方法学。本课程实验主要部分让学生通过将课程理论与实践两者紧密结合来掌握设计数字逻辑电路必需的理论基础和基本方法。本课程实验研究内容是电路的输入与输出之间的逻辑关系，并探索通过逻辑门电路的组合来实现这些逻辑关系[1]。可见实现基本逻辑运算和复合逻辑运算，可用单元门电路来进行搭建。

本课程配套实验概念性和实践性都很强，学生需要通过实验巩固数字逻辑及系统理论课的内容，训练学生对数字逻辑及系统的综合分析技巧。通过本实验课程，应能准确完整理解数字电路及系统的基本概念及分类；掌握常见数字电路类型及结构。

目前数字逻辑实验设备为数字逻辑实验箱，能够让学生掌握基本门电路特性及其测试条件与方法、常用组合逻辑电路设计的一般方法及组合逻辑电路的测试方法、常用时序逻辑电路如锁存器、触发器、计数器等的基本工作原理及设计应用。

但学生在做实验时往往由于复杂接线和某些线路故障、接触不良等原因而频出各种错误。一旦不清楚问题出处，就会导致时间无故花在检查线路及故障上而无暇顾及实验的主要目标，即熟悉逻辑电路及器件的功能、熟练掌握电路设计的方法。

本项目设计一个虚拟仿真实验系统，利用计算机仿真等技术对实验教学内容进行模拟，从而达到降低实验教学成本，提高教学质量的目的。这里的仿真就是让计算机根据一定的算法和仿真库来对电子电路设计进行模拟以验证设计并排除错误[2]。引入虚拟实验教学能够有效解决设备的消耗、易损和学生人数日益增加之间的矛盾，使实验不受时间空间的限制，在提高教育质量方面起着重要作用。

将器件虚拟化，将平台模拟化，将实验教学仿真化，无论是从人力还是物力上都是一种极大的节约，从而调动学生实践能力，发挥自主创新精神。虚拟仿真技术正是创新性计算机人才培养改革的产物。在仿真坏境下，学生既可以避免因实验器件本身的损坏所造成的实验障碍，同时也可以更主动地、自由地进行综合性实验，极大地拓宽了学生的创新思维空间。

1.2 课题研究的国内外发展情况

1.2.1 国外发展情况

目前国外有很多数字电路模拟仿真软件，各类开源闭源的仿真系统有很多，比如multiSIM7、EWB、Tina、Proteus[3]。 但是这些仿真系统过于专业性，对初学者来说用于课程实验明显易用性不够。这些专业仿真系统的功能是面向专业工作者的，因此它们都没有提供相应的基本支持让初学者来做课程上的配套实验。这明显不符合初学者情况的。而且专业仿真系统画布功能有很多限制，比如没有端口吸附功能，即不能让用户用鼠标自由拖拽端口到门电路芯片的某条边上。另外这些专业系统的画布没有连线随端口真值变色的功能，明显与用户的可视化交互工作做得不够多。

1.2.2 国内发展情况

由于国内的大量电子板卡都使用了外国芯片作为元器件，所以国内为了和国际标准接轨，使用的大多是从外国引入的专业仿真系统工具。即使国内有这方面的自研产品，其标准和专业功能也是向国外看齐的。因此对初学者来说也是易用性不够的。

1.3 课题研究的主要内容和创新点

1.3.1 课题研究的主要内容

本课题主要是面向数字电路教学实验用途的，因此这个课题的主要面向对象明显是针对学生和初学者的。该课题的存在意义在于替补实验室中的数字电路实验机箱来完成教学实验。之所以要替换实验中的机箱并取而代之为仿真系统，是因为据我所知实验中的机箱大多会因为时间老化原因或者学生的严重接线错误而变得不能使用，比如机箱接线插口生锈而接触不良或者学生在接线时造成短路均会导致机箱某些部分不能用。除此之外，实验中所用到的接线也会因学生的有意无意的过度拉伸卷折而变得接触不良或完全不能用。鉴于这些能够令实验失败或者无限期延迟的因素，虚拟化的仿真系统是十分必要的。然而要把这些实验搬到虚拟的仿真系统进行仿真，需要理清实验机箱能够完成实验的各种要素。首先机箱需要能够执行逻辑运算的门电路芯片。其次就是能够允许使用电线对这些门电路芯片的输入输出端口进行一定的连接操作以构成一个能够执行更复杂逻辑运算的电路拓扑。在使用电线进行连接拓扑时，正确地

对门电路芯片的输入和输出端口进行连接是完成实验的关键工作。然后机箱需要一些 LED 灯来观察验证实验的输出结果。 最后学生对最终正确的电路拓扑和实验结果进行记录。通过对实验机箱的上述分析，这个课题的研究内容主要有五点：

1. 能够对各种门电路芯片的输入输出逻辑关系进行仿真；
2. 能够对各种门电路芯片的端口进行相应连接并能对拓扑进行仿真；
3. 能够对拓扑连接中的错误进行检测识别并自动处理或交至用户处理；
4. 能够提供一些外部接口以和一些可视化组件交换数据；
5. 能够提供一些组件来可视化电路拓扑的输入输出状态；
6. 能够把使用者的拓扑结构和参数进行导入和导出操作。

1.3.2 课题的特色和创新点

本课题的特色主要体现在如下几个方面：

1. 采用了最流行的 Java 语言来编写这个仿真系统， 也就是说这个仿真系统可以跨平台执行，只要有 JRE 环境即可，不需要针对不同平台重新编写或编译；
2. 使用 MyEclipse2015 作为系统开发的 IDE 和项目管理工具，通过它可使系统代码组织结构利于管理，且在代码层面上也容易与其他企业级应用结合；
3. 用户可以对门电路拓扑进行导入导出操作以在其他机器上运行拓扑；
4. 用户可以任意布置门电路芯片中端口应该停靠在芯片的哪一条边上；
5. 仿真系统的画布能够让连线随端口真值变化而变色，方便用户观测仿真过程；
6. 添加了一些编程接口供系统和外部组件进行信息交换以实现信息的共享。

本课题的创新点主要体现在如下几个方面：

1. 用户可以使用已经提供的门电路芯片，也可以自己定制门电路芯片；
2. 定义门电路芯片逻辑的真值表达式可以使用“$” 特殊符号来控制芯片端口的真值变化是否激活芯片直接进入下一轮计算周期；
3. 改进了逆波兰算法以用于实现逻辑表达式的“逻辑截断”特性；
4. 使用了门电路连接信息分发/撤销机制来防止一个输入端口通过与其直接连接或间接连接的输入端口连到了多个输出端口，并用以防止一个输入端口通过与其直接连接或间接连接的输入端口对同一个输入端口进行多次连接。

1.4 课题研究的难点

本课题研究中虽然使用了最流行的 Java 语言来编写仿真系统， 可以使用 JDK 中包括图形库的很多库函数，然而在实际的研究开发过程中还是出现了很多难题需要克服和解决，其中以下几个难点最为关键：

1. 如何定义相应的类来描述门电路芯片及其端口，因为门电路芯片和端口都需要相应的类数据项来描述它们的内部属性，这些内部属性直接或间接决定了这些门电路芯片和端口在仿真时的静态参数和动态参数。若没有事先定义好这些类，必定会对后续代码编写造成极大影响。一旦类定义改变，就有可能牵扯到整个项目代码的结构和调用。因此如何对门电路芯片和端口进行恰当的类定义是至关重要的。

2. 如何协调好向画板添置门电路芯片的流程和端口间的连线流程是两件难事，因为这些流程的某些关键环节会对拓扑的某些要素进行修改，如果处理不好这些关键环节不仅有可能会导致拓扑数据的不一致性甚至会有可能导致仿真时的逻辑演算问题。因此处理好向画板添置门电路芯片的流程和端口间的连线流程是两个关键问题，仿真系统必须能够应付这些流程中的各项操作和变化情况。

3. 如何防止一个输入端口通过与其直接连接或间接连接的输入端口连到了多个输出端口，如何防止一个输入端口通过与其直接连接或间接连接的输入端口对同一个输入端口进行多次连接。由于这些问题的解决需要知道一个端口是否通过一个远程端口来连接了一个输出端口，一个端口是否通过一个远程端口形成了一个连接环路。因此为了解决这个问题，系统需要在其中利用一些连接信息共享的机制来彼此通告并同步彼此的连接状态。但是这个信息共享机制的建立是一个很有挑战性的技术活。

4. 如何处理有时序功能的电路芯片的逻辑演算是个问题，因为大部分有时序功能的门电路芯片其内部结构都是存在输出与输入进行连接的环节。正是这个原因，这些门电路芯片的一个逻辑演算周期并不等同于一次逻辑表达式的计算，而是若干次逻辑表达式的计算。怎么让门电路芯片在若干次逻辑计算后自行停下，是问题关键所在。

5. 如何和外部组件进行信息的同步共享是一个难题。因为系统底层框架是根据自己的仿真演算结果来改变拓扑中的端口动态参数，这些改变本来只有框架底层知道并处理的，怎么把这些变更通告给外部组件，该以什么数据形式来描述这种变更并发送至外部组件，该定义多少种变更通告，这些都是需要仔细的规划。因此系统底层框架和外部组件的变更信息共享和通告是一个难点。

2 关键技术

2.1 Java 技术

作为一种可以编写跨平台应用程序的纯面向对象程序设计语言， Java 最初被 Sun 公司定义为 Java 程序设计语言和 Java 平台的总称。它由 James Gosling 和同事们共同研发并于 1995 年正式推出。Java 更早期的名称为 Oak ，用于设计消费类电子产品的嵌入式芯片用途。 在 1995 年更名为 Java 并重新设计用以开发 Internet 应用程序[4] 。其中 Java 实现的 HotJava 浏览器显示了 Java 的魅力：跨平台、动态 Web、Internet 计算。从此 Java 被广泛接受并推动 Web 的迅速发展，常用浏览器均支持 Java applet。另一方面 Java 技术也不断推陈出新。Java 自面世后就非常流行，发展迅速，对 C++ 语言形成有力的冲击。在全球云计算和移动互联网的产业环境下，Java 更具备了显著优势和广阔前景。2010 年 Oracle 公司收购 Sun 公司，由其负责对 JVM、JDK、JRE 等的技术支持。Java 技术具有卓越的通用性、高效性、平台移植性和安全性，广泛应用于 PC 、数据中心、游戏控制台、科学超级计算机、移动电话和互联网，同时拥有全球最大的开发者专业社群。

2.2 Swing 技术

Swing 是一个用于开发 Java 图形界面应用程序的开发工具包， 它以抽象窗口工具包 AWT 为基础，使跨平台应用程序可以使用任何可插拔的外观风格。Swing 开发人员通过使用少量代码，就可以利用 Swing 包中丰富、灵活的功能和模块化组件类来开发出令人满意的用户界面。

Swing 工具包的基础 AWT 包拥有很多组件类，这些组件类是开发图形程序的基础。而 Swing 则在 AWT 的基础上对这些组件进行了修改和升级等。AWT 被 Swing 取代，主要是因为 AWT 包中组件类已不能满足日益增长的客户要求，且容易发生与特定平台相关的故障[5]。 Swing 组件更少地依赖于目标平台且更少地使用自己的 GUI 资源。因此尽量使用 Swing 组件来编程[5]。 而为了区分新的 Swing 组件类和与之对应的 AWT 组件类，Swing GUI 组件类都以字母 J 为前缀命名[5]。

2.3 MyEclipse 技术

MyEclipse 企业工作平台，是 MyEclipse Enterprise Workbench 的汉语名称，简称 MyEclipse。它极大扩展了 Eclipse IDE，能够让我们在数据库和 J2EE 上开发、发布。而在应用程序服务器的整合方面，它也极大提高了工作效率。 MyEclipse 是功能丰富的 J2EE 集成开发环境，包括了完备的编码、调试、测试和发布功能，并且完整支持 HTML，Struts，JSP，JSF，CSS，Javascript，SQL，Hibernate，Spring。 在体系结构上，MyEclipse 特征可被分为 7 类，分别为 Java EE 模型、Web 开发工具、EJB 开发工具、 应用程序服务器的连接器、 Java EE 项目部署服务、数据库服务和 MyEclipse 整合帮助。最新版 MyEclipse2015 有以下优点： 优化 Java EE 开发、保持厂商中立、RESTful Web 服务开发、项目工作流中保持 Maven、 使用构架等提升你的 Java Spring 项目、 PhoneGap 助你迅速开启移动策略、快速简单的企业交付、一体化 IDE 能满足你的需求、广泛的技术累积、多种应用服务器上轻松测试、移动和云开发就绪、在统一堆栈下为 Java EE 团队提供技术、集中管理 MyEclipse、非 IBM WebSphere IDE[6]。

3 系统分析

3.1 系统设计原则

3.1.1 系统实用性原则

一般的数字电路仿真系统在结构上和界面上都是十分专业复杂的，而且一般都是使用硬件语言来自动搭建电路拓扑。所以这些数字电路仿真系统并没有对画板中手动搭建数字电路拓扑提供足够的可视化交互支持和错误操作检查。初学者在没有学过硬件语言和工具使用手册的情况下是很难在这些系统上开展课程实验的。为了让学生能够方便研究电路的输入与输出之间的逻辑关系，探索通过逻辑门电路的组合来实现这些逻辑关系的方案，并利用基本逻辑门电路来实现基本逻辑运算和复合逻辑运算[1]，本系统在界面布局上尽量做到简单明了。在连接拓扑时也尽量减少用户操作步骤和距离，并使用一定的算法和预检测技术来减少用户的误操作，使系统实用性大大增强。

3.1.2 可维护性和可扩展性

由于使用纯面向对象编程语言 Java 作为系统开发语言， 所以开发过程中可以充分利用 Java 所提供的内置类特性来对系统各个底层功能进行模块化组件化以实行高内聚低耦合的软件工程原则。从而减少组件模块间的代码依赖性，便于系统代码的可重用性和扩展性，也便于对组件或模块进行代码层面的升级维护和系统性能调优。

另外系统内核中也增加了必要的数据信息交互接口，便于外部的一些可视化组件和系统底层之间的数据共享，实时把系统内核在仿真时的变化情况通过外部可视化组件显示到用户眼前。所以很容易开发其他外部组件并附加到整个系统框架之上。

3.1.3 系统的健壮性

该仿真系统在开发过程中，应该充分考虑初学者在搭建数字电路拓扑中的各种错误，比如把两个输出端口连到同一个输入端口就是初学者在搭建数字电路拓扑中可能会出现的错误。仿真系统如果不引入一种错误检测和错误处理机制的话，很容易造成仿真流程发生各种难以预料的错误甚至导致系统的崩溃假死。为此，仿真系统引入了一种检错和出错处理机制来维持仿真拓扑的完整性和一致性，以保持仿真流程的顺利进行。同时用户在系统中搭建数字电路拓扑过程中也会出现一些删除操作牵扯到多个端口的连接，比如删除芯片或者端口都会牵涉到与之连接端口的连接信息的修改。系

统在应用这些删除操作时应自动级联修改与之连接端口的连接信息，以保证拓扑的逻辑一致性。可见系统具备足够的措施处理各种异常错误以保证仿真系统的健壮性。

3.1.4 系统界面的整洁性

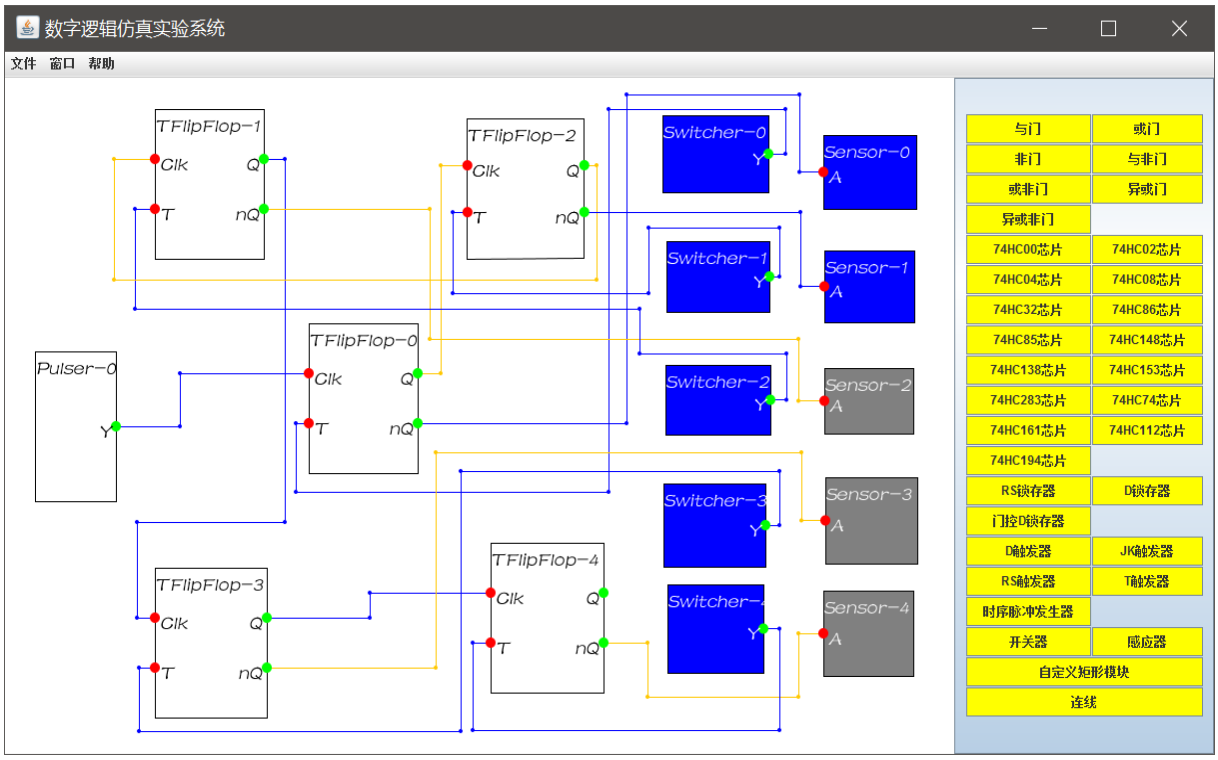


图 3.1 系统主界面图

考虑到系统的主要用户是来自学生的初学者，系统尽量使用简洁的界面作为系统主界面。如图 3.1 所示， 系统主界面被划分为一个搭建拓扑用的画板窗口、一个门电路芯片素材窗口和一个能调出各种可视化组件窗口的工具栏。初学者可从门电路芯片素材窗口中选择相应芯片并把其放置于画板中的任意位置来完成添加芯片的操作。

3.1.5 系统操作简便

为了和系统界面的整洁性搭配，系统在搭建拓扑时也尽量使用最少步骤完成用户的操作。在这里以 GNS3 操作流程为参考[7] 。用户首先需要在门电路芯片素材界面中选择相应芯片并放置到画板中任意位置。在连线过程中，用户首先需要在芯片素材窗口中切换系统模式为连线模式，然后把鼠标指针移动到其中一个芯片上方并左击以弹出该芯片端口列表，接着用户选择其中一个端口，此时该端口和鼠标指针间会有一条直线随鼠标指针游走，用户随之把鼠标指针移动到另一个芯片上方并左击以弹出该芯片端口列表，用户最后只需再选择一个端口即可完成一个完整连线操作流程。

3.2 系统需求分析

3.2.1 系统使用者

1. 学生用户

学生为系统的主要使用者，学生可以不需要实验室的机箱便可以通过这款仿真系统来完成课堂上的仿真实验。学生可以根据教师发放的拓扑文件来完成仿真实验，也可以根据教师的要求直接搭建数字电路拓扑并完成实验。当然学生也可以根据自己的兴趣要求来自行完成更难的实验而不需要事先申请机箱。

2. 教师用户

教师主要会通过这款仿真系统来定制实验内容，然后把这些实验内容分发给学生来完成。同时也减少了为学生布置实验机箱的麻烦。教师只要保证学生电脑能够运行 JRE 环境和拓扑文件的完整性即可做到上述内容。

3.2.2 系统应用场景

这款仿真系统是为了弥补机箱实验时存在机箱老化损坏或接线的接触不良等的缺点而开发的。这款系统会帮助学生免除以上这样的忧虑以更快地完成实验。通过使用这款仿真系统，学生可以不用机箱即可完成课堂布置的实验。学生只要有一台能运行 JRE 环境的电脑即可。学生完全可以随时随地完成这些仿真需要，极大提高了实验的灵活度也增加了学生的实验热情。当然为学生布置课程实验的教师也可以使用这款系统定制实验内容。教师既可以事先做好数字电路拓扑然后再发放给学生来完成模拟仿真实验，也可以直接发放实验要求让学生根据要求完成数字电路拓扑并保存提交。

3.2.3 系统用例图及其描述

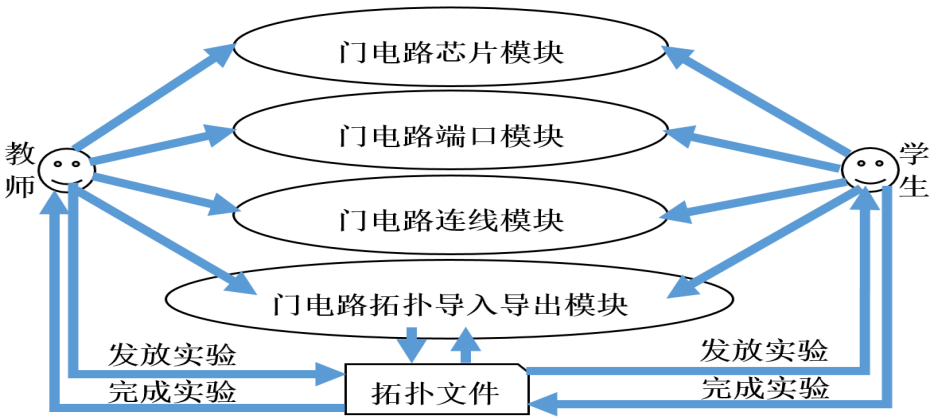


图 3.2 系统用例图

如图 3.2 所示， 教师和学生在使用该数字电路仿真系统时并没有功能上的区别。教师和学生都可能要对数字电路拓扑中的门电路芯片、门电路端口和门电路连线进行增删操作和修改操作并对最终成型的数字电路拓扑进行保存。唯一不同的是两者在使用门电路拓扑导入导出功能的目的不相同。学生导入拓扑可能是为了完成仿真实验，而导出拓扑可能是为了提交实验结果。反观教师，导入拓扑可能是为了检查学生实验完成程度或者完成待完成实验内容，导出拓扑则是为了发放实验内容给学生。

3.3 系统模块功能分析

3.3.1 门电路芯片模块

门电路芯片模块主要用于组织一个门电路芯片在仿真时的各项静态或动态参数并提供相应的函数方法以图形的形式绘制芯片内有关可视化的参数。这些参数至少包含了位置参数、长宽参数、角度参数、名字信息、端口列表和一些限制参数，这些限制包括是否允许修改名字、是否允许修改长宽参数、是否允许修改旋转角度信息、是否允许增删端口。门电路芯片模块本身应提供一组抽象函数把来自鼠标指针的事件信息转换为对门电路芯片内部参数的变更。这些内部参数必须被转换为适合可视化的格式来存储，以提高系统绘制拓扑元素的效率。除此之外，门电路芯片模块还必须控制门电路芯片的逻辑计算周期和逻辑表达式的编译和解释工作，以实现对芯片逻辑功能的仿真过程。总而言之，门电路芯片模块主要应有以下几个作用：

1. 门电路芯片静态或动态参数的获取和修改；
2. 门电路芯片可视化参数的图形化绘制工作；
3. 门电路芯片逻辑计算周期的控制工作；
4. 门电路芯片逻辑表达式的编译和解释。

3.3.2 门电路端口模块

门电路端口模块用于组织一个门电路端口的各项静态或动态参数并提供相应的函数方法来修改或获取这些参数。这些参数包含端口当前连接情况和端口在画板中实际位置。另外，门电路端口模块能够接受对方端口的真值修改操作和从对方端口发来的连接信息公告。总之，门电路端口模块主要应有以下几个作用：

1. 门电路端口静态或动态参数的获取和修改;
2. 返回门电路端口在拓扑画板中的实际位置；
3. 接受连接对端端口的真值修改操作和连接信息公告。

3.3.3 门电路连线模块

门电路连线模块用于对拓扑画板中的门电路芯片中的门电路端口进行连线操作。门电路连线模块通过调用门电路端口模块的函数方法修改连接双方端口的连接信息来达到连接要求。在连接时，门电路连线模块会对比双方端口的连接情况来判断是否已经直接或间接连接过彼此端口。

3.3.4 拓扑导入和导出模块

拓扑导入和导出模块用于拓扑文件静态数据和内存拓扑动态数据之间的转换。拓扑文件静态数据通过拓扑导入和导出模块可以重建门电路拓扑在内存中的表示形式。反之内存拓扑动态数据通过该模块也可以保存为文件形式上的静态拓扑数据。

3.3.5 用户与拓扑交互模块

用户与拓扑交互模块用于接收用户对拓扑元素变更操作并转换为对门电路芯片模块、门电路端口模块和门电路连线模块的函数方法的调用。用户与拓扑交互模块负责收集一组用户操作以识别用户操作意图并落实到底层调用。

3.3.6 拓扑数据可视化模块

拓扑数据可视化模块用于收集拓扑数据并对这些数据加以整理并绘制到拓扑画板中。拓扑数据可视化模块不仅可主动与底层拓扑数据同步，也可在仿真过程中接收来自底层的重绘消息并刷新画板，以保证用户操作结果能及时展现在用户眼前。

3.4 系统可行性分析

3.4.1 经济可行性分析

该虚拟仿真系统是基于 Java 语言开发的软件系统，因此该系统的部署环境是能够运行 Java 程序的环境，也就是说运行该系统的机器必须要有 JRE 环境才可以部署该虚拟仿真系统。 而 JRE 环境的部署是不需要花费任何授权费或购买费即可从网上

下载得到，因此运行该虚拟仿真系统并不需要增加额外的运行维护费用。其次，该虚拟仿真系统所用到的 Java 开发工具包 JDK（Java Development Kit）也能免费获得。因此，只需要在机器上安装 JDK 后即可完成搭载 Java 开发环境的工作。总而言之，该系统的研发经费仅会用在开发人员的配置和至少一台机器的购置上。

3.4.2 技术可行性分析

该虚拟仿真系统由于其自身结构功能的需要必须使用模块化的设计理念来开发。而利用 Java 这种面向对象语言可以让虚拟仿真系统各个功能块以类的形式来划分。这不仅让系统代码的结构性更好，也可以通过分配人员负责不同的功能块的开发工作来提高系统的开发效率。其次 Java 的标准库中有 AWT 和 Swing 两个图形库，前者负责基本的简单的图形开发，后者负责更高级的图形开发工作。在开发该虚拟仿真系统时可以根据实际的图形开发工作需要来充分发挥它们的长处和特点。因此完全可利用 Java 的算法库图形库来对电子电路设计进行模拟并依此来验证设计并排除错误[2]。

3.4.3 法律可行性分析

该虚拟仿真系统的代码编写全部使用 JDK 自带的标准代码库，完全没有使用到第三方的代码库。而 JDK 自带的标准代码库是开源的免费的，因此该虚拟仿真系统的代码是完全没有侵犯到其他代码著作权的，保证了其代码的合法性。其次，该虚拟仿真系统仅用于对门电路拓扑的仿真用途，没有任何侵犯用户权利的可能性。

3.4.4 操作可行性分析

该虚拟仿真系统在设计初期就考虑到了其操作易用性。在借鉴有类似功能的软件系统时，充分吸收了这些软件系统的用户操作特性，最大限度发挥用户的流水线操作来达到操作步骤最少操作跨度最短的效果。同时，该虚拟仿真系统的操作流程将遵循用户日常使用电脑的操作习惯来设计，让用户以最短时间了解该系统的操作流程。

4 系统设计

4.1 系统架构设计

由于本系统是一个数字电路仿真实验系统，因此该仿真系统首先需要一个仿真内核层来模拟仿真整个数字电路拓扑的仿真变化过程。仿真内核层需要提供仿真时所需的各种方法函数，不仅用于仿真内核层中各个模块的相互调用和数据消息通信，也用于供构建在仿真内核层之上的操作交互层和数据可视化层调用以满足它们对仿真内核层参数的获取和修改。操作交互层是仿真内核层和系统用户的过渡层，通过它，用户的具体操作请求会被转换为相应的操作去执行，以实现用户对拓扑的交互工作。数据可视化层用来把存放于仿真内核层的拓扑以图的形式绘制到用户眼前。

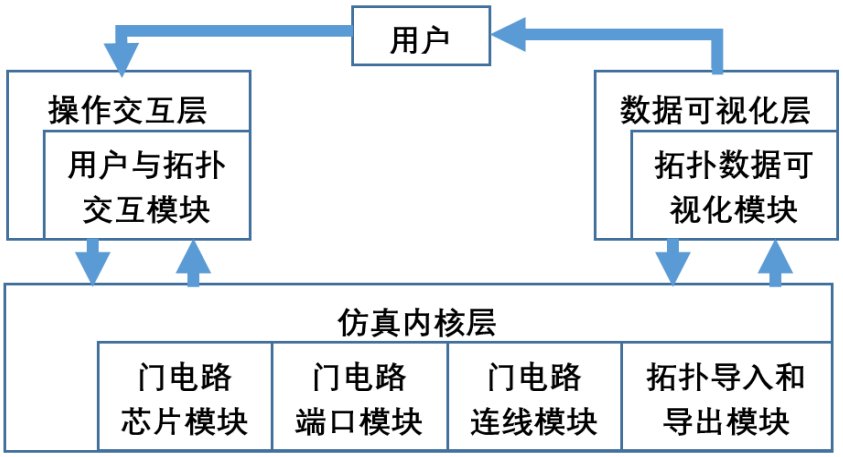


图 4.1 系统架构图

4.1.1 仿真内核层的角色功能

如图 4.1 所示，仿真内核层是这款仿真实验系统的关键所在。它由门电路芯片模块、门电路端口模块、门电路连线模块、拓扑导入和导出模块所组成。仿真内核层通过门电路芯片模块来定义和影响门电路芯片的各项功能参数，通过门电路端口模块来控制和修改门电路端口的各项仿真参数，通过门电路连线模块来协调连线流程的各项操作并间接调用门电路端口模块来修改连线双方端口的连接状态，通过拓扑导入和导出模块把仿真内核层中的门电路拓扑在内存动态数据和文件静态数据之间进行转换。

仿真内核层主要有两方面的功能。一方面对内部的门电路芯片模块、门电路端口模块、门电路连线模块和拓扑导入和导出模块的信息交流工作进行维护，以保证这些模块的工作能够按照一定的顺序来执行，从而正确完成上层发来的请求任务。一方面

对上层的操作交互层和数据可视化层进行打交道，要么响应操作交互层的操作请求，要么响应数据可视化层的数据查询请求。

仿真内核层控制着整个仿真流程的正常进行。仿真过程中的各种数值修改和连带操作都是通过仿真内核层中的 API 来完成的。这些 API 操作可能导致拓扑数据的变更而变得不一致，因此仿真内核层还维护着拓扑数据的逻辑性一致性。删除门电路芯片或者端口都会激活仿真内核层对与之连接端口的连接信息的修改以维持拓扑的逻辑一致性。由于仿真内核层掌管了拓扑所有数据，因此仿真内核层还需要响应用户拓扑导入和导出请求，要么从指定拓扑文件中提取拓扑数据并在仿真内核层中重构电路拓扑，要么以一定数据格式整合仿真内核层的拓扑数据并输出到用户指定的文件中。

4.1.2 操作交互层的角色功能

操作交互层是构建在仿真内核层之上的一个抽象层，它包含了用户与拓扑交互模块，负责在用户和仿真内核层间建立一个交互桥梁，起到了用户操作翻译官的功能。若用户需要更改仿真内核层中的门电路拓扑要素，则需要事先通过操作交互层的处理并转换为一组对仿真内核层的 API 调用。 也即是说操作交互层实际上就是把用户一系列操作如鼠标的移动、鼠标左右键点击等鼠标操作转换为对仿真内核层的一组 API 调用。操作交互层隐藏了仿真内核层的各种繁琐底层操作，因此有了操作交互层这样一个用户操作解释器，用户可以只要根据自身需要操纵鼠标即可透过操作交互层来间接控制仿真内核层的各种拓扑数据和控制仿真流程等操作。

4.1.3 数据可视化层的角色功能

数据可视化层是构建在仿真内核层之上的另一个抽象层，它包含了拓扑数据可视化模块，负责在用户眼前以图形的形式描述仿真内核层中拓扑数据和仿真结果，起到了拓扑数据翻译官的功能。数据可视化层会在拓扑数据发生变化的时候绘制仿真内核层中的拓扑数据，这些数据包括拓扑中的门电路芯片、门电路端口和门电路连线的数据。数据可视化层会根据门电路芯片中的端口性质进行不同的颜色标识，比如输入端口可以用红色标识，输出端口可以用绿色标识，当前被选中端口可以用靛蓝色标识。除了绘制拓扑数据外。数据可视化层还会对拓扑的仿真结果进行可视化，比如会根据输出端口的真值变化更改与之相连的连线的颜色。

4.2 系统功能结构

根据系统需求分析和系统架构设计，可设计出下面的系统功能结构图，如图 4.2。

图 4.2 系统逻辑模块结构图

4.2.1 仿真内核层的功能构成

1. 门电路芯片模块

门电路芯片模块负责对门电路芯片的相关事务，其中包括门电路芯片的创建、门电路芯片的删除、门电路芯片的命名、门电路芯片的移动、门电路芯片的扩缩、门电路芯片的旋转、门电路芯片的计算、门电路芯片的绘制这几项主要事务。门电路芯片的创建工作主要负责对门电路芯片各项参数的初始化并将其添加到门电路拓扑中；门

电路芯片的删除工作主要负责对门电路芯片的连接状态的清理工作并将其从门电路拓扑中移除；门电路芯片的命名工作主要负责修改门电路芯片的文字标识并向那些使用文字标识的组件通告修改后的文字标识；门电路芯片的移动工作主要负责同步修改芯片主对角线上两点的位置参数；门电路芯片的扩缩工作主要负责修改芯片主对角线上两点中某一点的位置参数的横轴或纵轴坐标；门电路芯片的旋转工作主要负责修改门电路芯片的方向向量并生成新方向仿射矩阵；门电路芯片的计算工作主要负责对逻辑表达式进行动态编译并放置到栈中执行；门电路芯片绘制工作主要负责对门电路芯片的位置参数和旋转参数进行混合计算并绘制门电路芯片的实际轮廓。

2. 门电路端口模块

门电路端口模块负责对门电路端口的相关事务，其中包括门电路端口的添加、门电路端口的删除、门电路端口的命名、门电路端口的吸附、门电路端口的绘制、解析端口实际位置、通告自身连接情况和接受对方连接公告。门电路端口的添加工作主要负责门电路端口各项参数的初始化并将其添加到所属的门电路芯片上；门电路端口的删除工作主要负责对门电路端口的连接状态的清理工作并将其从门电路芯片中移除；门电路端口的命名工作主要负责修改门电路端口的文字标识并向那些使用文字标识的组件通告修改后的文字标识；门电路端口的吸附工作主要负责对门电路端口在门电路芯片上的附着位置进行调整；门电路端口的绘制工作主要负责对门电路端口所附着门电路芯片的位置和该门电路芯片的旋转参数进行混合计算并在实际的坐标下绘制门电路端口；解析端口实际位置工作主要负责计算门电路端口的实际位置，正如上述绘制工作一样，这需要对门电路端口所附着门电路芯片位置和芯片的旋转参数进行混合计算；通告自身连接情况工作主要负责向某个直连端口发送除该端口外其余直连端口的所有连接信息；接受对方连接公告工作主要负责记录对方端口所发送的连接信息，并激活所属门电路端口把接受到的连接公告往其余直连端口发送，以保持端口连通图内部连接信息的一致性。

3. 门电路连线模块

门电路连线模块负责对门电路连线的相关事务，其中包括门电路端口间连线、解除端口间的连线、门电路连线的绘制和判断端口的重连线。门电路端口间连线工作主要负责对连线双方端口添加彼此的连接信息并激活双方端口的连接信息公告流程；解除端口间的连线工作主要负责从双方端口删除彼此的连接信息并激活双方端口的连接

信息公告流程；门电路连线的绘制工作主要负责获取连线双方端口的实际位置坐标并描绘连线；判断端口的重连线工作主要负责判断连线双方端口是否存有对方的连接信息，若存有对方端口的连接信息则表明这个连接是非法的，门电路端口间连线操作将被制止，从而避免端口的重连线。

4. 拓扑导入和导出模块

拓扑导入和导出模块负责对拓扑数据的转换工作，其中包括门电路拓扑的导入和门电路拓扑的导出。前者属于文件形式到内存形式的转换，而后者属于内存形式到文件形式的转换。门电路拓扑的导入过程其实是在模仿用户的操作，即先创建门电路芯片，然后往这些芯片添加门电路端口，最后再连接端口。这个导入过程需要上述三个模块的相互协助。门电路的导出过程则先导出门电路芯片信息，然后再导出门电路端口信息，最后才导出门电路连线的连接信息。这个顺序是为了方便拓扑的导入工作。

4.2.2 操作交互层的功能构成

操作交互层主要围绕用户与拓扑交互模块来起作用的。用户与拓扑交互模块负责用户与拓扑的交互工作，即把用户对拓扑的操作诉求转换为对拓扑的底层数据操作。其中包括收集用户操作意图、转换用户操作意图和调用内核层的方法。收集用户操作意图的主要工作是负责监控用户一组鼠标操作流程以获取用户对拓扑的操作诉求；转换用户操作意图的主要工作是负责将上阶段所获取到的用户诉求转换为一组有序的相应的仿真内核层方法；调用内核层方法的工作负责对上阶段转换来的一组仿真内核层方法进行依次调用。

4.2.3 数据可视化层的功能构成

数据可视化层主要围绕拓扑数据可视化模块来起作用的。该模块负责把底层拓扑的数据转换为一组可视化图形来显示在用户的眼前。其中包括绘制门电路拓扑和接收底层重绘消息。绘制门电路拓扑的主要工作是负责提供一个虚拟画布给门电路芯片模块、门电路端口模块和门电路连线模块并调用相应函数让其把各自的可视化参数绘制到这块虚拟画布上，最后虚拟画布会被显示到用户面前；接收底层重绘消息的工作主要负责监听来自仿真内核层中的拓扑数据变化消息，当这些消息与可视化数据的变更有关时，数据可视化层会重新绘制拓扑，以及时把拓扑实时状态显示到用户眼前。

4.3 系统操作流程

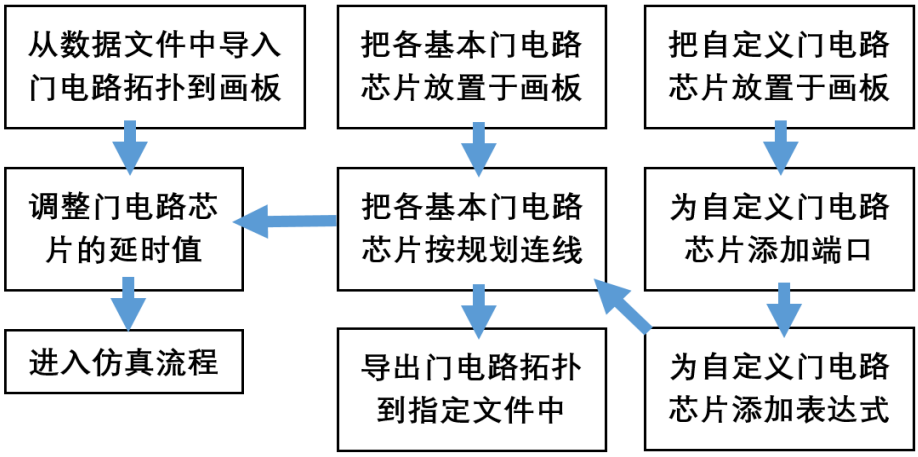


图 4.3 系统操作流程图

如图 4.3 所示，系统允许用户使用事先制定的基本门电路芯片或者用户自己制定的门电路芯片来搭建门电路拓扑。由于基本门电路芯片的类型和内部参数是互相对应的，因此用户不能够对其进行逻辑表达式修改操作和端口增删改名操作。而用户自定义的门电路芯片则没有上述限制，用户可以任意进行逻辑表达式修改操作和端口增删改名操作。用户在搭建门电路拓扑时，需要先往画板添加门电路芯片，若为自定义芯片则需要添加端口和逻辑表达式，然后再根据用户实际需要进行端口间的连线操作。在仿真时，如果有时序脉冲发生器参与到拓扑仿真，可以对相关的门电路芯片的延时值进行适当调整，以满足仿真的实际需要。

5 系统模块功能设计与实现

5.1 门电路芯片模块

5.1.1 芯片逻辑表达式具体形式的设计与实现

门电路芯片作为一个特定的运算单元，应根据输入端口的真值变化，重新计算其输出端口的真值状态，以保持门电路芯片输出和输入之间的逻辑关系。这里的逻辑关系直接体现在芯片逻辑表达式的设定上。芯片逻辑表达式支持赋值运算“=”、或运算“|”、与运算“&”、异或运算“^”、非运算“!”和用于嵌套的左右括号“()”， 另外还提供用于分割语句的分号“;”和特定修饰符“$”。除“$”特定修饰符外， 其余运算符的优先级及结合方式与 C 语言的逻辑运算符类似[8]，具体见表 5.1：

表 5.1 芯片逻辑表达式运算符优先级

优先级	运算符	名称或含义	使用形式	结合方向	说明
1	$	修饰符	$端口名	右到左	单目运算符
	()	圆括号	(表达式)	左到右	--
2	!	非运算符	!表达式	右到左	单目运算符
3	|	或运算	表达式|表达式	左到右	双目运算符
	&	与运算	表达式&表达式	左到右	双目运算符
	^	异或运算	表达式^表达式	左到右	双目运算符
4	=	赋值运算符	端口名=表达式	右到左	双目运算符

逻辑表达式的一般形式如公式 5.1 和 5.2 所示:

$$\text{Expr} \rightarrow \text{Word} \mid \text{Word;} \mid \text{Word; Word} \tag{5.1}$$

$$\text{Word} \rightarrow \text{Word pop Word} \mid \text{sop Word} \mid (\text{Word}) \mid \text{Pn} \tag{5.2}$$

其中 pop 和 sop 分别为双目和单目运算符，Word 代表中间表达式，Pn 代表端口名字。如与门表达式 “$Y=A&B”，Y 为输出端口的名字，A 和 B 为输入端口名字。

在这里要说明一下修饰符“$” 的存在意义，由于某些门电路芯片其逻辑结构本身就存在内循环，比如触发器就会把上个输出状态纳入到下个输入状态中，即触发器的某个输出端口和某个输入端口进行了连接，因此基于逻辑表达式是顺序执行的这样的考虑下，门电路芯片上个稳定状态到下个稳定状态之间不能只限制在一次逻辑表达式的顺序执行中，而应由芯片是否达到稳定状态来决定逻辑表达式的顺序执行次数，

即逻辑表达式的执行次数取决于芯片的端口真值什么时候不再发生变化，所以为了实现这一功能，端口的真值变更会默认触发门电路芯片的再运算，如图 5.1。 然而这样有可能会面临死循环的风险，因为若不是触发器最终再次达到了一个平衡状态，按照常理是会陷入计算逻辑表达式的无限循环中，无限循环可用于产生时序信号。因此修饰符“$”用于禁止端口真值变更引起门电路芯片的再运算，以防止死循环的发生。

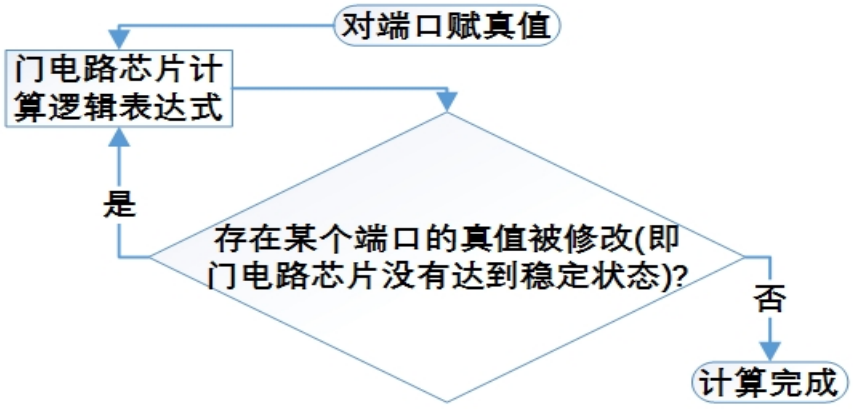


图 5.1 默认的芯片逻辑计算流程图

5.1.2 芯片逻辑表达式运算机制的设计与实现

为了方便对逻辑表达式的解释运算，门电路芯片会事先把逻辑表达式转换为逆波兰形式，然后才把逆波兰形式的表达式按顺序放入栈中运行。由于逆波兰形式的表达式作为一种后缀表达式，它把运算符的优先级和结合方式以及左右括号的嵌套都转化为逆波兰表达式的出栈顺序上，因此门电路芯片只要把栈中逆波兰式按出栈顺序解释运行即可实现原始表达式所表达的逻辑[9]。以异或门中缀表达式举例，见公式 5.3：

异或门中缀表达式：$Y=(!A&B)|(A&!B) (5.3)

异或门后缀表达式：$Y, A, !, B, &, A, B, !, &, |, = (5.4)

改进的后缀表达式：B, !, A, &, B, A, !, &, |, $Y, = (5.5)

这里要说明一下改进的后缀表达式。若按照四则运算中缀表达式转后缀表达式，异或门后缀表达式 5.3， 应该转换为公式 5.4， 即“$Y, A, !, B, &, A, B, !, &, |, =”。可以看出实际上只要把双目运算符的两个参数表达式的处理顺序颠倒一下就可以得到公式 5.5 的后缀表达式形式，换句话就是把中缀表达式第一个参数表达式的后缀形式放到比第二个参数表达式的后缀形式更靠近出栈位置， 即把“Word_A op Word_B”的中缀形式转换为“Word_B, Word_A, op”而不是“Word_A, Word_B, op”。之所以要这样转换，完全是基于某些运算的逻辑截断特性来考虑的，因为某些运算，比如或运算

的第一个参数表达式为真时就不需要再对第二个参数表达式进行运算，与运算的第一个参数表达式为假时就不需要再对第二参数表达式进行运算[8]。为了满足这些逻辑截断特性，转换出来的后缀表达式应该能让双目运算符的第一个参数表达式的后缀形式最先被出栈处理，处理出来的结果被用以判别随后的第二个参数表达式的后缀形式是否需要被进一步处理运算。一旦在某个双目运算符下其第一个参数表达式的真值运算结果符合该双目运算符的逻辑截断特性，那么该双目运算符的第二个参数表达式的运算过程就可以被省略以减少运算量。

在这里还要说明一下特定修饰符“$”和赋值运算 “=”是如何联合参与运算的。首先赋值运算符“=”会把其右边表达式的运算结果与左边端口名所对应的端口真值进行比较。如果赋给端口的新值和端口旧值相同，则该赋值运算不执行，此时与端口名是否被“$” 修饰无关。若赋给端口的新值和其旧值不相同，则该赋值运算起效并把待赋真值应用到端口上。然后根据端口名是否被“$” 修饰来决定是否触发该端口所对应门电路芯片的再运算，此时若端口名被“$” 修饰则直接跳过触发门电路芯片再运算，否则若端口名没有被“$” 修饰则触发端口所在门电路芯片计算逻辑表达式，该计算过程会不断地执行直到芯片再次达到稳定状态。具体流程见图 5.2。

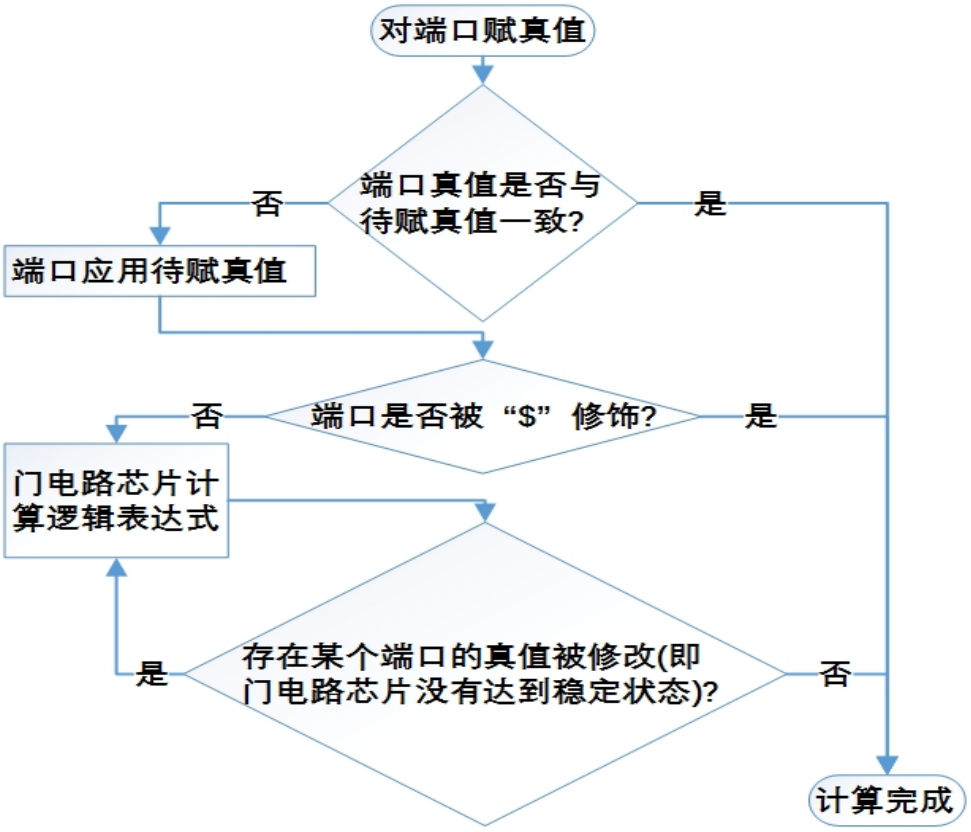


图 5.2 有修饰符“$”和赋值运算符“=”参与下的芯片逻辑计算流程图

5.2 门电路端口模块

5.2.1 端口连接信息分发机制的设计与实现

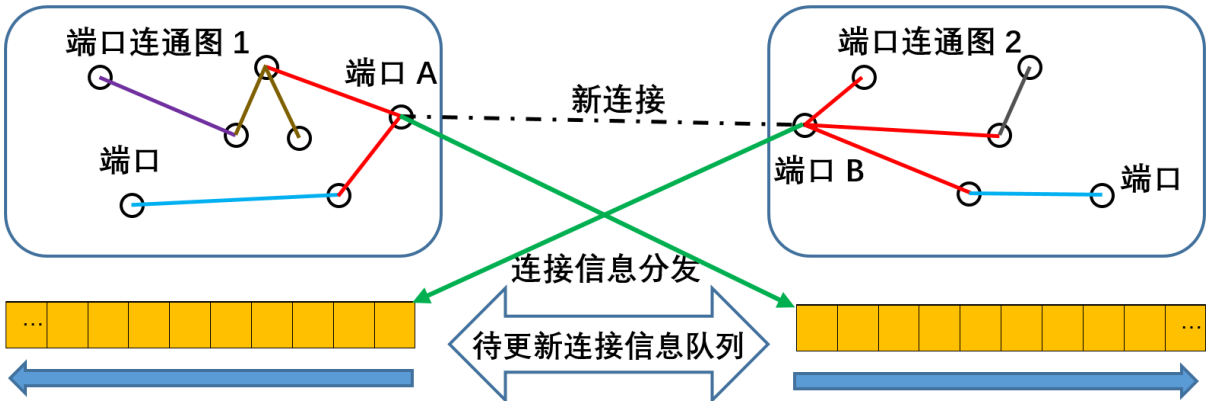


图 5.3 端口连接信息分发机制示意图

为了防止端口之间的连接出现环路现象，每个端口必须知道自己所在的端口连通图的连接状态，为此需要一种机制在端口连通图的连接状态发生变化时分发连通图的连接变更信息。在这里本文借鉴了 Rip 路由协议[10, 11]部分工作原理。 如图 5.3 所示，当两个互不连接的端口连通图中各取一个端口 A 和 B 进行连接时， 将会导致两个连通图的融合并激发双方的连接信息同步过程。该过程正是这个端口连接信息分发机制来维护管理的。当端口 A 和 B 尝试进行连接时，会首先互相确认对方是否存在于自己所在的连通图中。如果对方不存在于自己所在的连通图中，将会继续检测连接后的连通图是否存在两个输出端口。若连接后的连通图存在两个输出端口，则连接是非法的并拒绝该连接请求。如果上述两次检测都通过，则会激活端口连接信息分发机制以同步两个连通图的连接信息。端口连接信息分发机制会在两个方向上进行连接信息的同步工作。以图 5.3 为例，一个方向上，端口 A 会将自己背后的端口连接信息和自身信息发往端口 B 的待更新连接信息队列并等待端口 B 的线程处理。另一个方向上，端口 B 会将自己背后的连通端口信息和自身信息发往端口 A 的待更新连接信息队列并等待端口 A 的线程处理。当端口 A 和 B 从它们的待更新连接信息队列中更新连接信息后，会将这些信息继续发往跟它们直接连接的端口的待更新连接信息队列中并等待处理。这些信息会一直蔓延到端口连通图的各个端口处以完成端口连通图内所有端口的连接信息同步。当同步完成后，两个连通图中的每个端口都会拥有两个连通图的连接信息，也就是说完成了合并成为一个新的完整连通图的所有必须的连接信息共享过程，使新连通图内任一端口都知晓除自身外连通图内所有端口的存在。

5.2.2 端口连接信息撤销机制的设计与实现

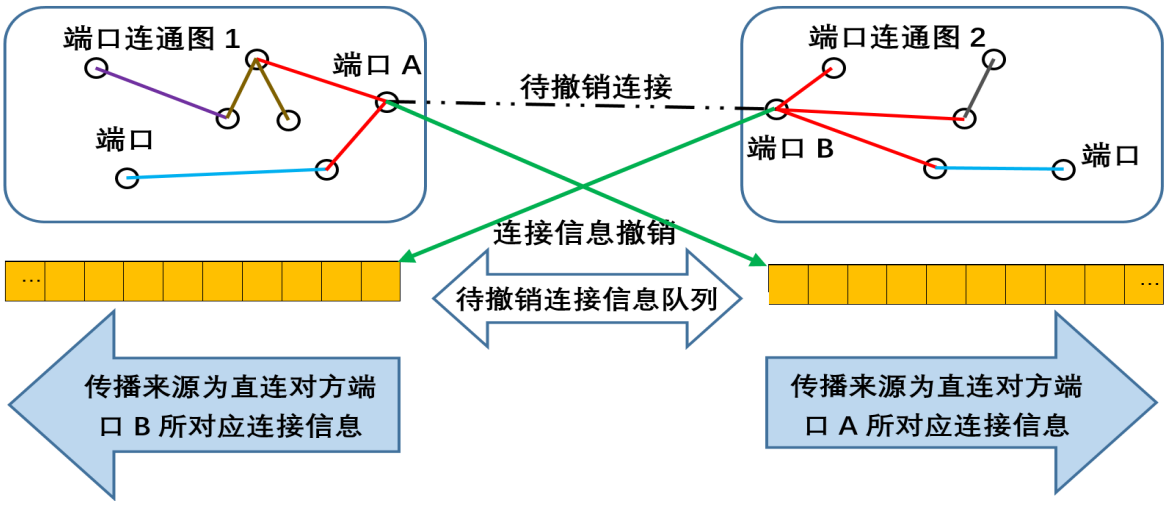


图 5.4 端口连接信息撤销机制示意图

端口连接信息撤销机制与端口连接信息分发机制的处理过程是类似的，就是把待撤销连接信息从某端口发送到跟该端口周围直连端口的待撤销连接信息队列中并等待处理，然后这些直连端口会在处理的时候会以自身端口作为信息公告来源继续发送这些待撤销连接信息到这些直连端口周围的直连端口，并从自己的待撤销连接信息队列中删除这些待撤销连接信息。待撤销连接信息会一直蔓延直至连通图边沿端口。端口连接信息撤销机制和端口连接信息分发机制一样遵循“水平分割”原则，即从某端口发往到其周围直连端口的连接信息公告，不会再从其周围直连端口发往到该端口。

端口连接信息撤销机制与端口连接信息分发机制相同地方还体现在机制触发后两个方向上的传播路径上。端口连接信息分发机制两个方向上都是从待连线端口到连线前对方端口的连通图，而端口连接信息撤销机制两个方向上则都是从待撤销连线端口到撤销连线后对方端口的连通图。以图 5.3 和图 5.4 来说，端口连接信息分发机制两个方向上分别为端口 A 到端口连通图 2 和从端口 B 到端口连通图 1， 端口连接信息撤销机制两个方向上也分别为从端口 A 到端口连通图 2 和从端口 B 到端口连通图 1。

基于这些相同之处，在代码实现上，端口连接信息撤销机制和端口连接信息分发机制作为一套统一机制使用同一个缓冲区。之所以这两个机制能够统一起来，与它们在公用的队列上使用统一的数据结构来封装连接信息离不开的。每一个进入队列的连接信息公告都包含了三类标识，分别为信息公告来源、待处理连接信息和处理方式。其中，信息公告来源代表发送这条信息的端口，待处理信息代表要处理的连接信息，处理方式则决定待处理信息是用于分发还是撤销的。这样就完美统合了两个机制。

5.2.3 端口真值信息分发机制的设计与实现

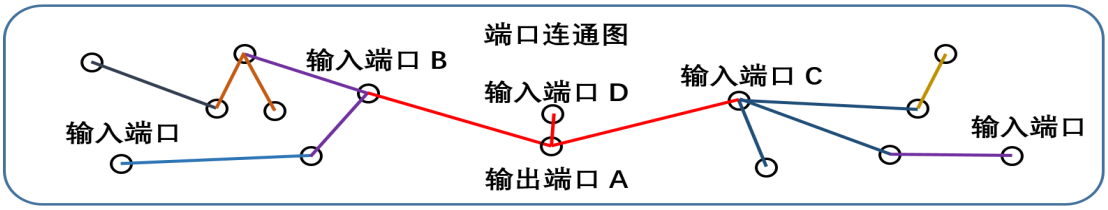


图 5.5 端口真值信息分发机制示意图

在对门电路拓扑进行仿真时，有些输出端口的真值会随着仿真过程的进行而发生变化，而这些变化亦会导致输出端口所在端口连通图的输入端口的真值的变化。这些变化在端口连通图中的扩散工作是由端口真值信息分发机制来处理执行的。 以图 5.5 为例，若输出端口 A 的真值发生变化， 则端口 A 会向最近的三个输入端口 B、C、D 发送真值变更请求。 当输入端口 B、C、D 的线程被唤醒并接收来自输出端口 A 的真值变更请求时，输入端口 B、 C、 D 的真值将会应变更请求并同步为请求中所携带的真值。当输入端口 B、 C、 D 的真值同步后，若同步前后的真值相异，输入端口 B、C、D 还会向与自己直接连接的端口发送真值变更请求，并等待目的端口的处理。这个过程遵循“水平分割”原则，亦即一个端口的真值变更请求导致了另一个端口的真值变化，则后者不会再向前者发送真值变更请求。这里有个需要注意的地方就是，真值变更请求并不是存放于队列中，这跟端口连接信息分发机制和端口连接信息撤销机制的处理是不同的。实际上真值变更请求也可以像端口连接信息分发机制那样存放于一个待处理队列中，但是当真值变更请求的增长速度远大于处理请求的速度时，使用队列存放变更请求这样的一种处理将不能满足实时仿真的要求，因为必须要处理旧变更请求后才能处理新的变更请求。这是我不愿意看到的，为此我只提供了能容纳一个真值变更请求的空间来存放真值变更请求。这样当空间中的真值变更请求在没有被处理的时候，一个新的真值变更请求进来时，就可以直接覆盖掉旧的真值变更请求。当端口的进程被唤醒时，就可以直接处理最新的真值变更请求，而不必再处理那些更旧的真值变更请求，处理效率得到了根本性的提高，满足了实时仿真的要求。

在端口连接或断开时，端口真值的变更不再由端口真值信息分发机制来处理，而是作为一种“捎带”操作由端口连接信息分发/撤销机制来处理的。因为输入端口一旦接收到的端口连接公告存在某个输出端口信息，就可根据公告的处理方式将输入端口的真值设置为输出端口真值或默认真值而不用再通过端口真值信息分发机制来处理。

5.2.4 端口连接信息分发/撤销机制与端口真值分发机制在实现时的异同

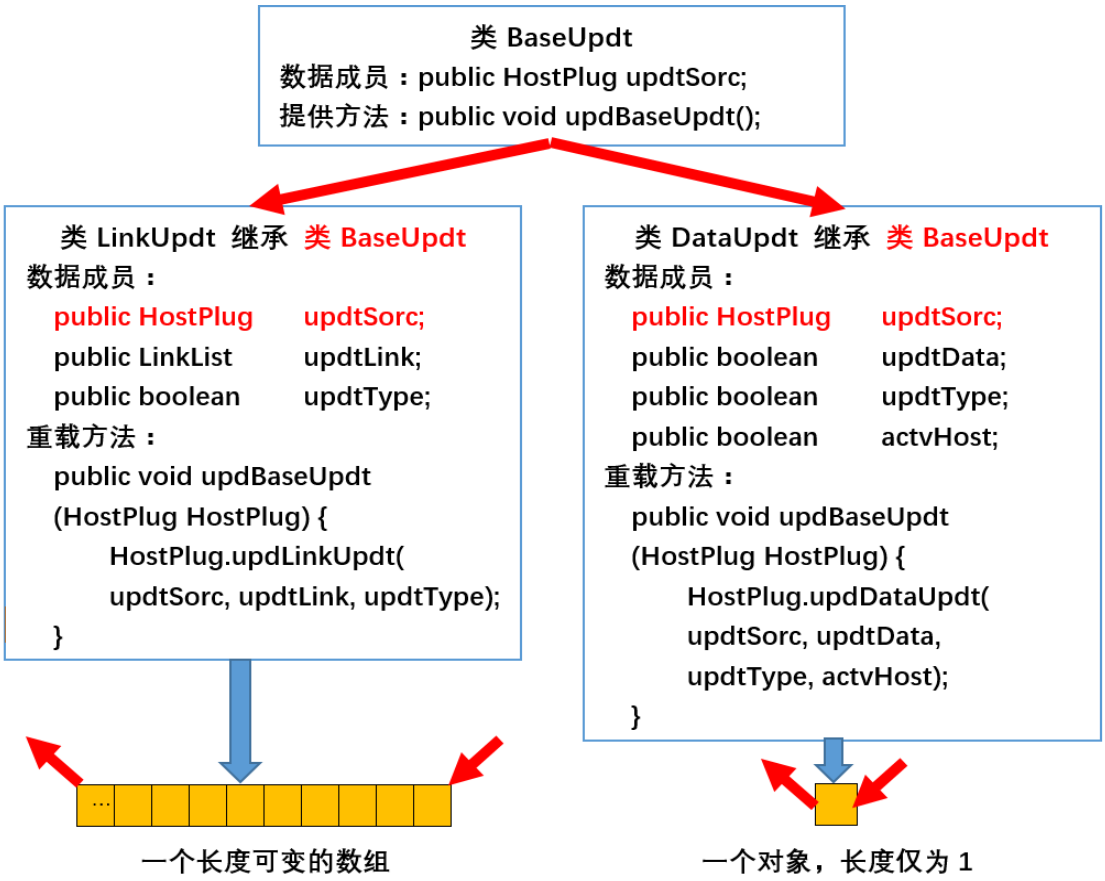


图 5.6 端口连接信息分发/撤销机制与端口真值分发机制的类结构图

如图 5.6 所示，端口连接信息分发/撤销机制与端口真值分发机制所用到的类分别为 LinkUpdt 类和 DataUpdt 类，使用这些类所生成的对象作为各自的信息公告用于同步信息。这两个类都继承了 BaseUpdt 类，其中继承得到的数据成员 updtsorc 用于标识信息公告的来源，重载的 updBaseUpdt 函数用于处理信息公告。 updBaseUpdt 函数会根据信息公告的类型来调用相应的方法，若信息公告属于 LinkUpdt 类对象则对端口调用 updLinkUpdt 方法， 否则对端口调用 updDataUpdt 方法。类 LinkUpdt 的非继承得到的数据成员 updtLink 代表该信息公告的待处理连接信息， updtType 代表这个信息公告类型，true 时则把待处理连接信息添加到端口连接记录表， 见图 5.8， 否则从端口连接记录表删除这些待处理连接信息。 类 DataUpdt 的非继承得到的数据成员 updtData 代表信息公告所要求端口的真值；updtType 代表该信息公告是否不再传播，true 时表示该端口处理完信息后继续向它的直连端口传播公告， 否则只处理不传播；actvHost 代表端口处理完信息后是否激活端口所属门电路芯片执行逻辑表达式计算。

5.3 门电路连线模块

5.3.1 门电路连线执行过程的设计与实现

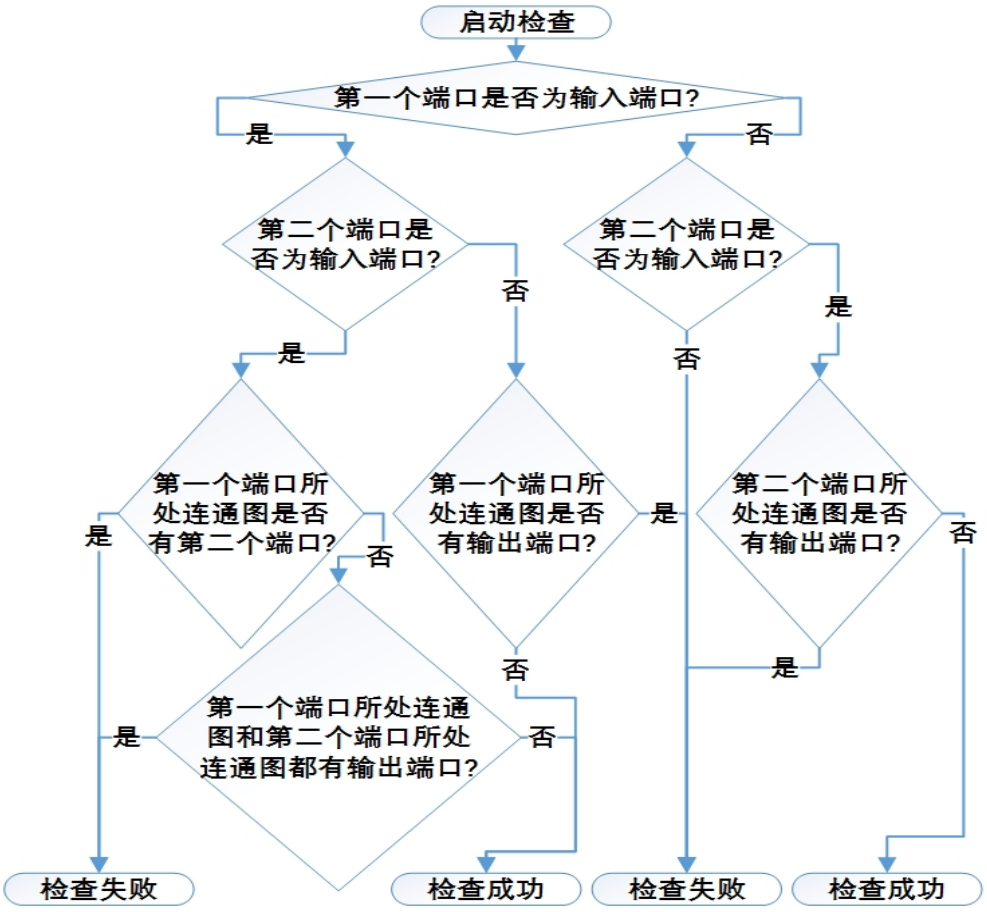


图 5.7 门电路连线检查过程流程图

门电路连线执行过程主要分为两个阶段。第一阶段是检查阶段，检查待连接的两个端口其输入输出属性和所处环境是否符合连接条件。如果符合连接条件则该次连接生效，否则连接请求被忽略掉。正如刚才所说的，是否符合端口连接条件需要考虑双方端口类型属性及其所处环境这两个因素，这里的所处环境指的是端口所处连通图的当前连接情况。综合这两个因素考虑，检查阶段首先会检查参与此次连接的第一个端口是否为输入端口。如果第一个端口为输入端口，则若第二个端口为输入端口但第一个端口所处连通图有第二个端口或第一个端口所处连通图和第二个端口所处连通图都有输出端口，或第二个端口虽为输出端口但第一个端口所处连通图已连接有一个输出端口，那么该连接不符合条件。如果第一个端口为输出端口，则若第二个端口为输出端口或虽为输入端口但其所处连通图已连接有一个输出端口，那么该连接也不符合条件。除以上情况外，连接请求都可成功被执行的。具体过程如图 5.7 所示。

正如图 5.7 所示， 门电路连线过程中所使用的检查流程是优化过的。这主要体现在待连接的两个端口只有一方为输出端口时，只需要检查输入端口的另一方所处连通图是否有输出端口就可以知道是否符合连接的要求，而不需要再检查一方是否存在于另一方所处连通图中。如果输入端口的另一方所处连通图有输出端口，那么这个输出端口要么为待连接输出端口的一方，即一方存在于另一方所处连通图中，要么为另一个输出端口，此时由于连通图中只允许一个输出端口的存在，因此一方不存在于另一方所处连通图中，但这两种情况都是不允许执行连接请求的。而如果输入端口的另一方所处连通图没有输出端口，那么待连接输出端口的一方肯定不在输入端口的另一方所处连通图中，这种情况下连接请求肯定会被成功执行的。也就是说，在待连接双方端口中只有一方为输出端口时，检查一方是否存在于另一方所处连通图中是多余的。

门电路连线执行过程的第二个阶段是同步阶段，主要是交换待连接的两个端口所在连通图的端口连接情况，使之成为一个完整的新连通图。在这个阶段下，系统会先建立两个端口的连接，然后再调用之前说过的端口连接信息分发机制来进行连接信息的同步。之所以是这样的顺序，是为了在建立连接时分配好空间用以接收连接信息。接受到的连接信息会以以下结构来分类组织成为端口连接记录表，见图 5.8。

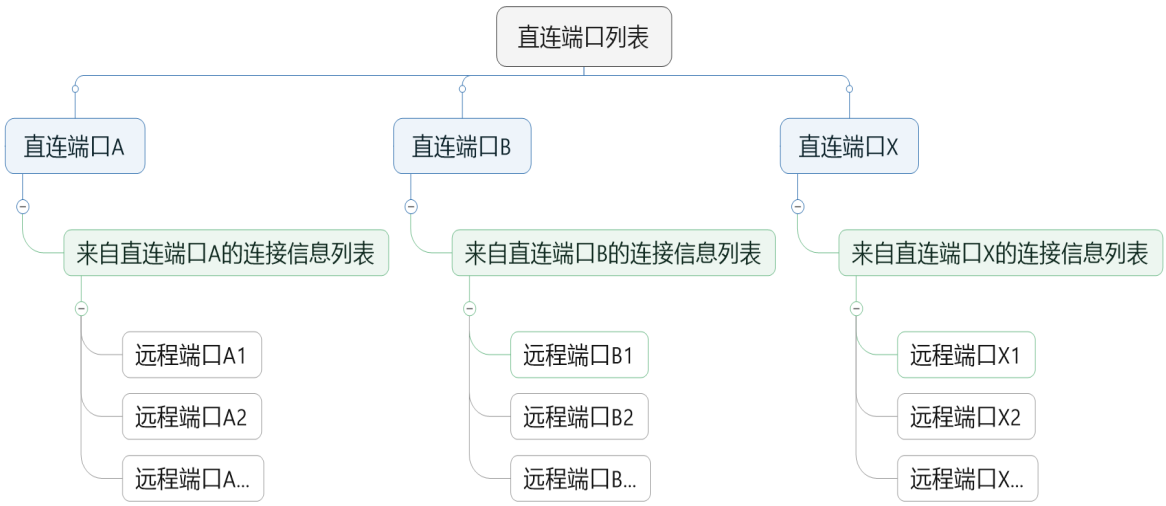


图 5.8 端口连接记录表结构图

说明一下，这里的直连端口指的是通过连线直接相连的端口，远程端口则包含了直连端口和通过直连端口间接连接的端口。如果以“跳数”来说，直连端口“跳数”为 1，远程端口的“跳数”大于等于 1。 使用这样的结构来分类组织连接信息，有它的好处，就是可以区分端口所接收到的连接信息的传播来源，这样在撤销连线时，就可以直接把传播来源为连线对方端口的对应连接信息往其余直连端口方向传播即可。

5.3.2 门电路连线撤销过程的设计与实现

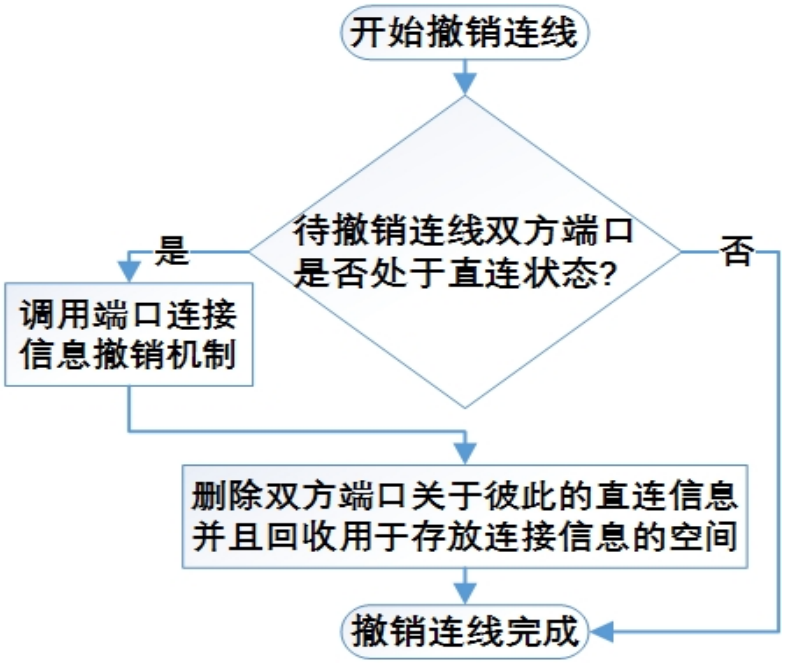


图 5.9 门电路连线撤销流程图

门电路连线撤销过程主要分为两个阶段。第一个阶段是检查阶段，主要检查待连接的两个端口是否处于直连状态。如果是则该次连线撤销请求生效，否则此次连线撤销请求被忽略掉，因为对不存在的连接执行撤销操作是没有意义的。第二个阶段是连线撤销阶段，在这个阶段下，系统会先调用之前说过的端口连接信息撤销机制，以待撤销连线的双方端口为起点来撤销彼此对方连通图中相关联的连接信息，使一个原本完整的连通图被划分为两个互为分离的两个连通图。然后再从待撤销连线的双方端口删除彼此的直连信息并回收用于存放连接信息的空间，具体过程见图 5.9。

在这里要说一下，当某个端口通过端口连接信息撤销机制把要撤销的连接信息封装到连接信息公告并发往另一个端口后，另一个端口是如何根据连接信息公告来删除自身某些连接信息。如果某个端口从自己的连接信息公告队列中通过检测公告的第三类信息——处理方式后，发现为一个撤销连接的信息公告，那么就会提取公告的第一类信息——信息公告来源和第二类信息——待处理连接信息，然后遍历该端口的端口连接记录表，端口连接记录表结构图见图 5.8， 如果直连端口为信息公告来源，那么从该直连端口的连接信息列表中删除与待处理连接信息相同的条目，否则以当前端口为信息公告来源向直连端口发送封装了这些待处理连接信息的新撤销连接信息公告。按照这样的删除流程，可以从一个连通图内删除相应的连接信息。

5.4 门电路拓扑导入和导出模块

5.4.1 门电路拓扑导入过程的设计与实现

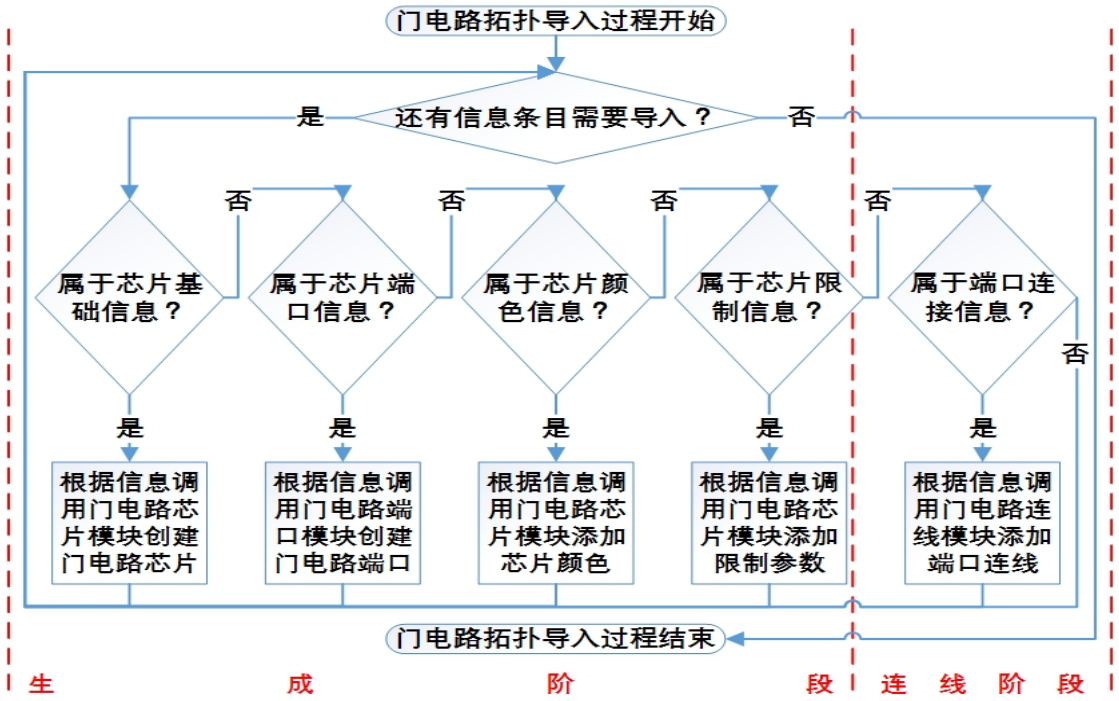


图 5.10 门电路拓扑导入流程图

门电路拓扑导入过程主要负责通过 BufferedReader[12] 对象将静态拓扑数据从文件中提取出来并以一定数据结构重新导入内存中，以方便系统对拓扑数据的动态修改。拓扑文件的数据组织形式是在考虑过拓扑导入时的情形后得到的，也就是说拓扑文件中的数据形式是有利于从文件中导入数据的。这个有利性体现在当系统扫描到一条门电路芯片基础信息条目后，系统会在接下来扫描到该芯片所携带端口的相关信息条目和芯片颜色信息及其限制参数信息。这样做的好处就是不用定位已导入的门电路芯片所在内存位置了，而是直接就近原则对最近导入的门电路芯片做相应的操作如添加端口到门电路芯片上。当然这样做的另一个好处就是门电路芯片所携带端口的信息将不用再绑定端口所属芯片的定位信息，即芯片名字，减少了拓扑文件的大小。

总体上，门电路拓扑导入过程分为前后两个阶段，如图 5.10。第一个阶段为生成阶段，这个阶段主要提取门电路芯片基础信息及其携带端口的信息、门电路芯片颜色方案及其操作限制，并生成相应的门电路芯片及其携带端口。在这个阶段期间，不会有任何端口连接操作，以避免出现端口未生成就用于连接并导致系统异常的情况。

第二个阶段为连线阶段，这个阶段主要根据拓扑文件相关端口连接信息，把相应

端口进行连接。进入这个阶段就意味着之前阶段的所有门电路芯片及其端口都已经生成并处于内存中，因此此时的端口连接操作是安全的，不会出现要连接的端口还没有生成的情况。不过安全的代价是丧失了部分连接效率，由于连接阶段运行在生成阶段之后，失去了就近处理的条件，因此每条连接都涉及到 4 个名字的定位，这 4 个名字定位分别用于寻找连接双方端口所属门电路芯片和双方端口在所属门电路芯片下的内存对象位置。在寻找到连接双方端口在内存中位置的引用后，就可以调用相应函数来让它们进行连接了。在连接过程中，将会如之前所说的，会首先进入门电路连线执行过程并在随后调用端口连接信息分发机制。由于端口连接信息分发机制使用了队列来存储周围端口发来的连接信息公告，因此即使处理连接信息公告的各个端口线程的处理速度和因建立端口连接所产生的连接信息公告的生成速度不尽相同，也不会导致连接信息的丢失而产生最终生成的门电路拓扑在各个连通图内部的端口连接信息不同步的情况。所以在连接阶段不用担心端口连接信息分发机制的可靠性问题。

5.4.2 门电路拓扑导出过程的设计与实现

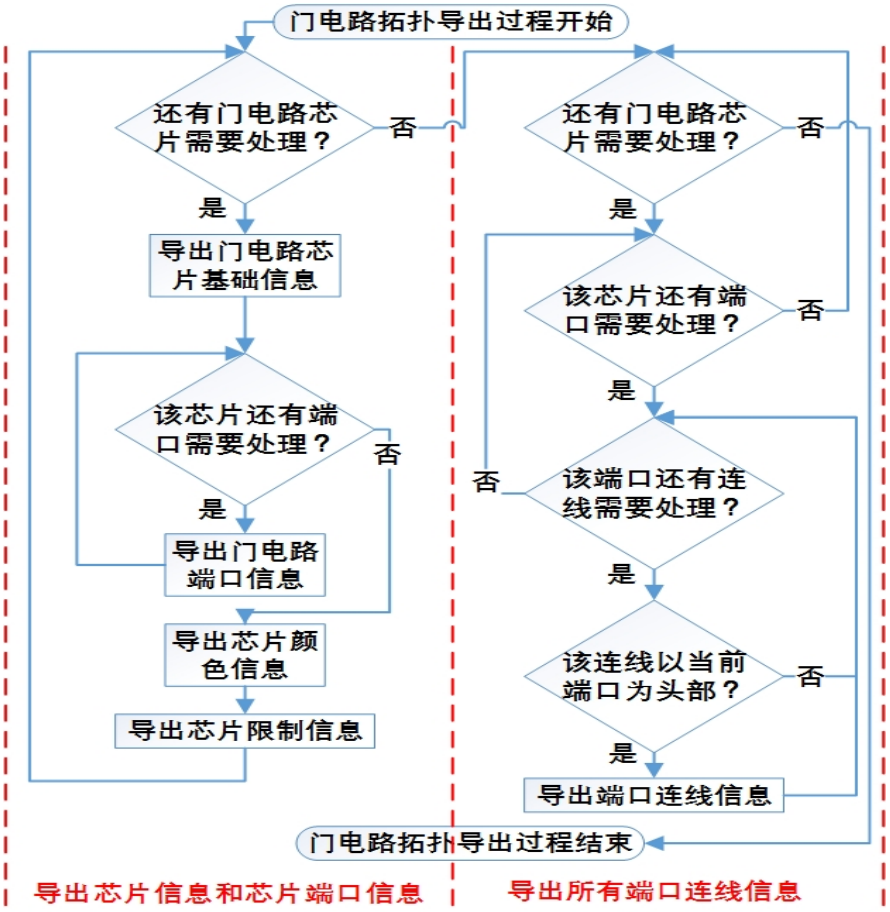


图 5.11 门电路拓扑导出流程图

门电路拓扑导出过程主要负责将以一定数据结构分散于内存中的动态拓扑信息集中起来并以静态数据形式通过 FileWriter[12]对象来导出数据。 之前说过，拓扑文件的数据组织形式应对数据从文件中导入是有利的。根据门电路拓扑导入过程的细节，门电路拓扑导出过程也分为两步，见图 5.11。以非门为例，第一步是导出芯片和端口信息。这步会先导出门电路芯片基础信息，见条目 5.6， 接着导出该芯片携带端口的信息，见条目 5.7 和 5.8，最后把芯片的颜色参数和限制参数导出，见条目 5.9 和 5.10。

第二步是导出端口连接信息。在导出所有芯片和端口信息后，剩下的就是导出端口连线信息。以 Pulser-0 芯片的 Y 端口和 Not-0 芯片的 A 端口之间的连线为例，其相关信息的组织形式见条目 5.11：

Host#0#NOT-0#$Y=!A#479.0#176.0#560.0#320.0#1.0#0.0#10.0#0 (5.6)

Plug#Y#0#false#false#false#true (5.7)

Plug#A#2#true#false#false#false (5.8)

Colr#NULL#-16777216#-65536#-16777216#-14336#-14336#-14336#-14336#-14336#-14336#-14336#-14336#-16711936 (5.9)

Lmit#false#true#true#false#true#true#false#false#false#true#true#true#false#true (5.10)

Conn#Pulser-0#Y#NOT-0#A#445.0#299.0#445.0#248.0 (5.11)

在这里说明一下这些信息条目被“#”号分割出来的各个部分所代表的意思含义。信息条目 5.6 从左到右被“#”号分割出来的各个部分所代表意思分别为：

1. 信息条目类型标识为芯片基础信息类型；
2. 门电路芯片类型，0 为普通芯片，1 和 2 分别代表感应器和开关器；
3. 门电路芯片的名字标识；
4. 门电路芯片的逻辑表达式；
5. 门电路芯片左上角横轴坐标；
6. 门电路芯片左上角纵轴坐标；
7. 门电路芯片右下角横轴坐标；
8. 门电路芯片右下角纵轴坐标；
9. 门电路芯片的角度向量的横轴参数；
10. 门电路芯片的角度向量的纵轴参数；
11. 门电路芯片上各个鼠标控制区域的半径；

12. 门电路芯片的线程睡眠时间。

信息条目 5.7 和 5.8 从左到右被“#”号分割出来的各个部分所代表意思分别为：

1. 信息条目类型标识为门电路端口信息类型；
2. 门电路端口的名字标识；
3. 门电路端口吸附在芯片上的哪条边，0 到 3 分别代表在芯片右下左上；
4. 门电路端口类型，false 为输出端口类型，true 为输入端口类型；
5. 门电路端口是否被隐藏用于芯片内部使用，false 为非隐藏，true 为隐藏；
6. 门电路端口是否被高亮标注，false 为非高亮标注，true 为高亮标注；
7. 门电路端口真值状态，false 为假，true 为真。

信息条目 5.9 从左到右被“#”号分割出来的各个部分所代表意思分别为：

1. 信息条目类型标识为门电路芯片颜色信息类型；
2. 门电路芯片的填充颜色，NULL 为无填充颜色；
3. 门电路芯片的内边框颜色；
4. 门电路芯片被选中时边框为正方形时的外边框颜色；
5. 门电路芯片被选中时边框不为正方形时的外边框颜色；
6. 门电路芯片的右边鼠标控制区域的颜色；
7. 门电路芯片的下边鼠标控制区域的颜色；
8. 门电路芯片的左边鼠标控制区域的颜色；
9. 门电路芯片的上边鼠标控制区域的颜色；
10. 门电路芯片的右上角鼠标控制区域的颜色；
11. 门电路芯片的右下角鼠标控制区域的颜色；
12. 门电路芯片的左下角鼠标控制区域的颜色；
13. 门电路芯片的左上角鼠标控制区域的颜色；
14. 门电路芯片的旋转控制区域的颜色。

信息条目 5.10 从左到右被“#”号分割出来的各个部分所代表意思分别为：

1. 信息条目类型标识为门电路芯片限制信息类型；
2. 门电路芯片是否显示被隐藏端口；
3. 门电路芯片是否显示内边框；
4. 门电路芯片是否填充颜色；

5. 门电路芯片是否显示外边框；
6. 门电路芯片是否显示芯片名字标识；
7. 门电路芯片是否显示端口名字标识；
8. 门电路芯片是否被禁止调整大小；
9. 门电路芯片是否被禁止调整角度；
10. 门电路芯片是否被禁止调整其中端口位置；
11. 门电路芯片是否被禁止往其中添加端口；
12. 门电路芯片是否被禁止从其中删除端口；
13. 门电路芯片是否被禁止修改其中端口名字标识；
14. 门电路芯片是否被禁止修改芯片名字标识；
15. 门电路芯片是否被禁止修改芯片的逻辑表达式。

信息条目 5.11 从左到右被“#”号分割出来的各个部分所代表意思分别为：

1. 信息条目类型标识为端口连线信息类型；
2. 连线的第一个端口所属门电路芯片的名字标识；
3. 连线的第一个端口的名字标识；
4. 连线的第二个端口所属门电路芯片的名字标识；
5. 连线的第二个端口的名字标识；
6. 剩余参数表示这条连线的轨迹路径信息。

这里说明一下连线的轨迹路径信息，由于每两个参数代表一个点坐标，所以剩余参数的个数必须为偶数，并且由于轨迹路径的多样性，剩余参数的个数也是可多可少的。

5.5 用户与拓扑交互模块

5.5.1 用户与门电路芯片交互过程的设计与实现

1. 用户调整门电路芯片大小

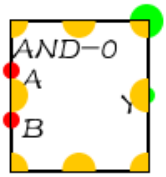


图 5.12 被选定门电路芯片的示意图

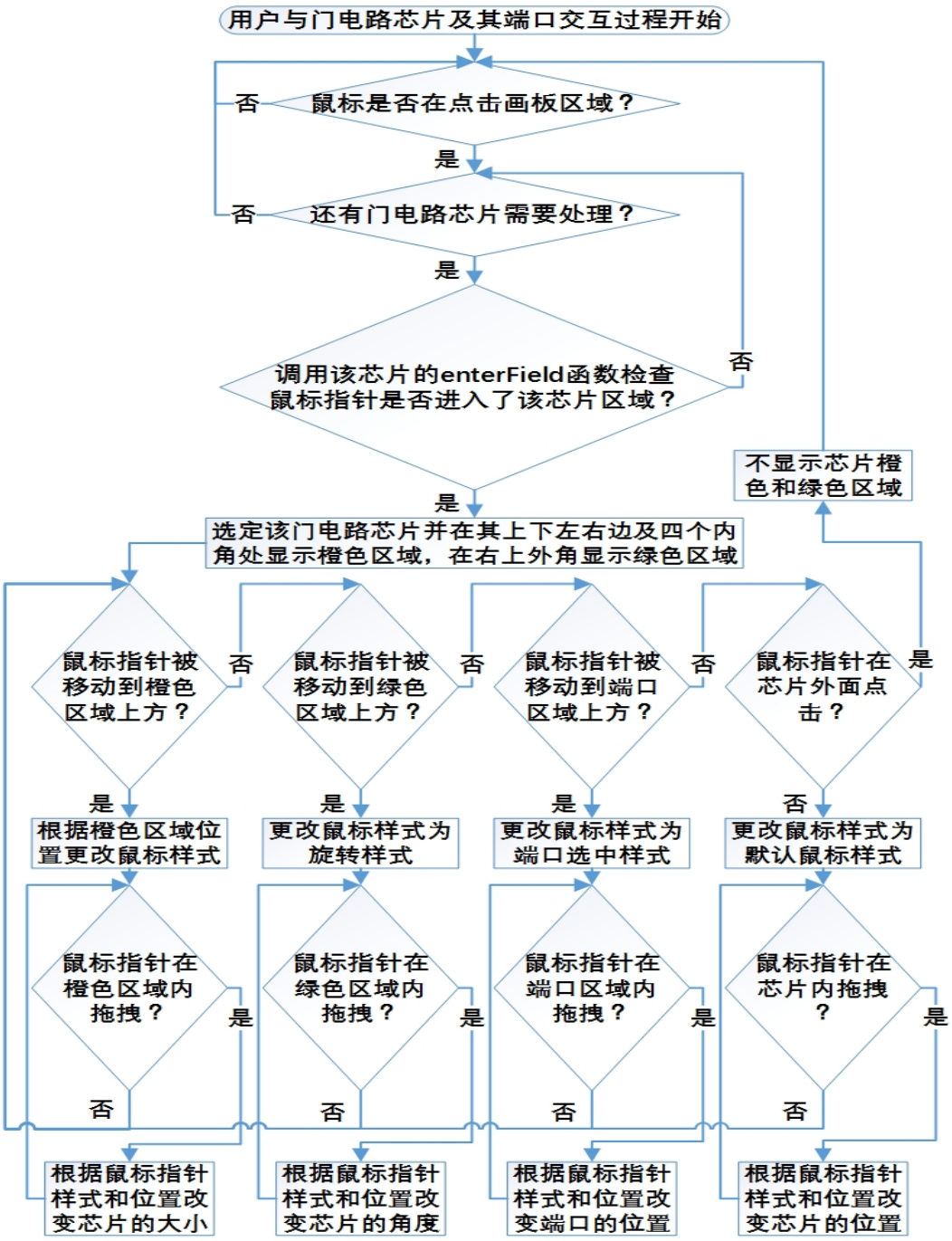


图 5.13 用户与门电路芯片及其端口交互过程流程图

如图 5.12 和 5.13， 用户在仿真系统画板区域上的点击操作会触发一系列检查流程。当检查流程通过调用每个门电路芯片上的 enterField 函数并检查到当前鼠标的点击位置坐标进入了某个门电路芯片的区域上，那么该门电路芯片就会被选定且选定的门电路芯片会在上下左右及四个内角处显示橙色区域。此时鼠标移动操作会触发一系列区域检查流程， 被选定门电路芯片的函数 entersButtE、entersButtW、entersButtS、entersButtN、entersButSE、entersButNE、entersButSW、entersButNW、 entersRotat 会

被依次执行。一旦进入了某个区域就更改鼠标的样式，如进入右边的橙色区域，就把鼠标样式设为 Cursor.E_RESIZE_CURSOR。 当鼠标进入拖拽操作，即按住左键不放并移动鼠标，根据此时鼠标样式就可知道被拖拽的边或角，然后就可以调用相应的移动边或角的函数来改变门电路芯片大小了， 如调用门电路芯片的 movHstEdgeE 函数来移动芯片右边。移动边或角的函数会根据芯片的角度参数和原来的位置参数来计算修改芯片的左上角或右下角的实际位置坐标参数，并在修改完相关的位置参数后，调用内部函数 setHostComp 来重建用于拖拽边或角的区域以方便下一轮的区域检测。

为了精确移动门电路芯片的边或角，在拖拽边或角的开始，即按住鼠标左键的一瞬间，会调用相应的函数计算鼠标位置到边的距离，如 getDisEdgeE 函数，得出的距离值用于校正移动位置。计算鼠标位置到边的距离的函数和移动边或角的函数一样，需要根据当前芯片角度参数和原来位置参数来计算的，具体来说就是先把鼠标位置坐标转变为以芯片中心，以芯片角度为旋转角度的坐标系上的坐标位置，然后以转换后的坐标和芯片的各个边的参数作对比。拖拽角操作的开始会记录距离最近两边的距离，这与拖拽边操作的开始记录一边的距离不同。在实现上，只要实现了拖拽边的操作，拖拽角的操作就完成，因为拖拽角的操作实际上就是同时拖拽两条边的操作。

2. 用户调整门电路芯片角度

如图 5.12 和 5.13， 用户在仿真系统画板区域上的点击操作会触发一系列检查流程。当检查流程通过调用每个门电路芯片上的 enterField 函数并检查到当前鼠标的点击位置坐标进入了某个门电路芯片的区域上，那么该门电路芯片就会被选定。选定的门电路芯片会在芯片的右上外角显示绿色区域。此时鼠标移动操作会触发一系列区域检查流程，其中被选定门电路芯片上的函数 entersRotat 会在该过程中被执行。一旦进入了绿色区域就更改鼠标的样式为 Cursor.MOVE_CURSOR。 当鼠标进入拖拽操作，即按住左键不放并移动鼠标，根据此时的鼠标样式就可以知道这是一个旋转操作，然后就可以调用 movHostRota 函数来变更门电路芯片的角度了。 movHostRota 函数会根据当前芯片的角度参数和原来的位置参数来计算新的角度参数并在修改完角度参数后调用内部函数 setHostComp 来重建用于拖拽边或角的区域以方便下一轮的区域检测。

3. 用户调整门电路芯片位置

如图 5.13，用户在仿真系统的画板区域上的点击操作会触发一系列检查流程。当检查流程通过调用每个门电路芯片上的 enterField 函数并检查到当前鼠标的点击位置

坐标进入了某个门电路芯片的区域上，那么该门电路芯片就会被选定。如果鼠标没有进入该模块的可操作区域但仍在模块内，则当鼠标进入拖拽操作，即按住左键不放并移动鼠标，根据此时的鼠标默认样式就可知道这是一个移动芯片的操作，然后就可以调用 movHstPoint 函数来变更门电路芯片的位置并调用内部函数 setHostComp 来重建用于拖拽边或角的区域以方便下一轮的区域检测。

为了精确移动门电路芯片位置，在拖拽开始即按住鼠标左键的一瞬间，会调用相应函数计算鼠标位置到芯片左上角的水平距离和垂直距离以校正芯片的最终位置。需要注意的是这里的芯片左上角是没有考虑角度的芯片左上角，也就是说用函数得到的芯片左上角坐标位置和在画板上显示的芯片左上角坐标位置可能是不一样的。之所以这样做，是因为获取画板上显示的芯片左上角的坐标位置是多余的甚至是不理智的选择。实际上，芯片中记录的位置参数，即左上角和右下角位置坐标，都是不考虑芯片角度参数的，只是在显示到画板的时候再混合了角度参数来考虑而已，亦即先画芯片形状再旋转。所以移动一个有旋转角度的形状和先移动没有旋转角度的形状再旋转是同等效果的，简单来说就是移动芯片不用考虑角度参数。如果考虑角度来移动芯片，不但增加逆旋转转换的计算代码和计算时间，还损失了计算精度，简直吃力不讨好。

4. 用户调整门电路芯片所在层

用户调整门电路芯片所在层在代码实现方面还是很简单的。由于画板中的所有门电路芯片都是存在于一个数组中，因此在画板上显示的门电路芯片在绘制时候都是按照其在数组中的位置按顺序绘制的，也就是说门电路芯片在数组中的位置越靠前其所在层越靠近画板底部，数组中的位置越靠后越靠近画板顶部。有了上面的逻辑，我们需要让门电路芯片上浮一层，则把它在数组中的位置向后挪一位；需要让门电路芯片下沉一层，则把它在数组中的位置向前挪一位；需要让门电路芯片上浮到顶，则把它放置到数组的最后；需要让门电路芯片下沉到底，则把它放置到数组的头部。在调整完门电路在数组中的位置后，再让系统重新遍历数组从头到尾绘制门电路芯片即可。

5.5.2 用户与门电路端口交互过程的设计与实现

1. 用户调整门电路端口位置

如图 5.13，用户在仿真系统的画板区域上的点击操作会触发一系列检查流程。当检查流程通过调用每个门电路芯片上的 enterField 函数并检查到当前鼠标的点击位置

坐标进入了某个门电路芯片的区域上，那么该门电路芯片就会被选定。此时鼠标移动操作会触发一系列区域检查流程，其中被选定门电路芯片的函数 getPointPlg 会在该过程中被执行。一旦鼠标进入了被选定门电路芯片上某个端口区域，getPointPlg 会返回该端口对象的引用，并更改鼠标样式为 Cursor.CROSSHAIR_CURSOR。 当鼠标进入拖拽操作即按住左键不放并移动鼠标，根据此时鼠标样式就可知道这是一个调整门电路端口位置的操作，然后就可调用 movPlgPoint 函数来调整门电路端口的位置了。

movPlgPoint 函数会根据鼠标位置坐标和芯片中心所连直线与门电路芯片旋转后四条边的交点来确定端口被移动到哪条边上，并根据交点的位置坐标来确定端口应该被插入的位置。这里被插入的位置要说一下，由于门电路芯片上的四条边都各有一个端口列表数组来记录每条边的端口，其中数组从小到大的位置代表了端口在对应边上从左到右或从上到下的位置，因此刚才所说的交点坐标会逐一与各边上的端口坐标对比，一旦发现交点坐标比当前对比的端口在边上的坐标更靠右或更靠下就交换被移动端口和被对比端口它们在数组上的位置。另外交点坐标并不是被移动端口在边上的最终坐标，因为边上的端口的坐标是根据门电路芯片位置、所属边的长度和该边所对应数组中的位置平均分配的，也就是说边上端口的坐标会根据该边端口数目平分该边。

2. 用户添加门电路端口到芯片中

用户在仿真系统的画板区域上的右击操作会触发一系列检查流程。当检查流程通过调用每个门电路芯片上的 enterField 函数并检查到当前鼠标的点击位置坐标进入了某个门电路芯片的区域上，那么该门电路芯片就会被选定并调用 getHostComm().show 来显示针对该门电路芯片的所有可用操作。此时如果在弹出的右键菜单中选择“添加输入端口”或“添加输出端口”，就会调用 JOptionPane.showInputDialog 来显示“添加输入端口”或“添加输出端口”对话框。在该对话框中使用者需要键入要添加端口的名称。键入端口名称后，系统会检查名称是否为空或者已经被同芯片内其他端口所占用。检查通过后，会对选定门电路芯片上调用 addHostPlug 函数创建带有指定名称的输入或输出端口。以下为 addHostPlug 函数的定义：

```
HostPlug addHostPlug(
                String plugName, int plugEdge, boolean plugType,
                boolean plugHide, boolean plugMark, boolean plugData)
```

该函数会根据参数列表中各个具体数值的含义来创建端口并返回该端口的引用，

其中 plugName 为用户指定端口名称，plugEdge 指定端口在芯片上哪条边，plugType 为端口输入输出属性， plugHide 指定端口是否为隐藏端口，plugMark 指定端口是否高亮，plugData 指定端口真值，默认为 false。在创建输入端口时，各参数 plugEdge、plugType、plugHide、plugMark、plugData 分别为 HostPlug.WESTEDGEPLUG、true、false、false、false。在创建输出端口时，plugEdge、plugType、plugHide、plugMark、plugData 分别为 HostPlug.EASTEDGEPLUG、false、false,、false、false。

3. 用户从芯片中删除门电路端口

用户在仿真系统的画板区域上的点击操作会触发一系列检查流程。当检查流程通过调用每个门电路芯片上的 enterField 函数并检查到当前鼠标的点击位置坐标进入了某个门电路芯片的区域上，那么该门电路芯片就会被选定。此时鼠标移动操作都会触发一系列区域检查流程，其中被选定门电路芯片上的函数 getPointPlg 会在该过程中被执行。一旦鼠标进入了被选定门电路芯片上某端口区域，getPointPlg 会返回该端口对象的引用，并更改鼠标样式为 Cursor.CROSSHAIR_CURSOR。 此时鼠标右键会调用 getPlugComm().show 来显示针对该端口的所有可用操作。此时若在弹出的右键菜单中选择“删除端口”，则调用 JOptionPane.showConfirmDialog 来显示“确认删除端口”确认框，若选是，则对芯片调用 delHostPlug 函数删除端口，否则放弃

delHostPlug 函数会首先检查传来的端口引用是否与该模块上任一个端口对应，如有对应端口，则对该端口调用 delAllConns 函数来切除与该端口有关的连线然后再从芯片的端口数组中删除该端口。切除连线时， delAllConns 函数会在所有连线的两端都调用一次 pshPlugLink 函数以在两个方向上触发端口连接信息撤销机制来更新分离后连通图的连接状态。事实上， delAllConns 函数执行的是一个批量的门电路连线撤销过程，它与执行单个连线的撤销的 delPlugConn 函数在原理上是一样的。只不过 delAllConns 函数针对端口的所有连线，delPlugConn 函数针对端口的其中一个连线。

5.5.3 用户与门电路连线交互过程的设计与实现

1. 用户移动连线轨迹路径折线

用户在仿真系统画板区域上按下鼠标左键会触发一系列检查流程。当检查流程通过调用 getConnLine 函数遍历拓扑上的所有连线并检测到当前鼠标位置坐标进入了某条连线的可控区域上，那么该连线就会被选定并返回该连线信息的引用。此时左键不

放并进入拖拽操作，就会以该连线引用为参数调用 movPathLine 函数移动连线轨迹路径折线。 这里要注意的是，由 getConnLine 函数返回的连线信息引用包含了连线双方端口、连线的轨迹路径和与鼠标位置最近的那段折线在连线轨迹路径数组上的位置。因此可以通过连线引用中包含的信息来确定鼠标要移动的是连线上的哪一段折线。另外离端口最近的那段折线不受这里的操作影响，因为离端口最近的那段折线是由端口的位置来确定，也就是说那段折线的位置是由端口所属门电路芯片的位置和该端口在芯片上的附着位置来共同确定的，不能对其单独操作。

2. 用户从拓扑中删除连线

用户在仿真系统画板区域上的右击操作会触发一系列检查流程。检查流程通过调用 getConnLine 函数遍历所有连线， 若检测到当前鼠标位置坐标进入了某条连线的可控区域上，那么该连线就会被选定并返回该连线信息的引用，与此同时系统还会调用 getLineComm().show 来显示针对该连线的所有可用操作。 此时若选择“删除连线”，则调用 JOptionPane.showConfirmDialog 来显示“确认删除连线”确认框， 若选是，则通过之前得到的连线引用获取连线的任一端的端口 A， 并以另一端端口 B 的引用作为参数对端口 A 调用 delPlugConn 函数即可删除连线。

3. 用户对连线轨迹路径添加折线

用户在仿真系统画板区域上的右击操作会触发一系列检查流程。检查流程通过调用 getConnLine 函数遍历所有连线， 若检测到当前鼠标位置坐标进入了某条连线的可控区域上，那么该连线就会被选定并返回该连线信息的引用，与此同时系统还会调用 getLineComm().show 来显示针对该连线的所有可用操作。此时如果在该右键菜单中选择“添加折线”，就会以右击时位置作为折点坐标参数对连线引用调用 addPathLine 函数，即可添加一段折线。

4. 用户对连线轨迹路径删除折线

用户在仿真系统画板区域上的右击操作会触发一系列检查流程。检查流程通过调用 getConnLine 函数遍历所有连线， 若检测到当前鼠标位置坐标进入了某条连线的可控区域上，那么该连线就会被选定并返回该连线信息的引用，与此同时系统还会调用 getLineComm().show 来显示针对该连线的所有可用操作。 若选择“删除折线”，则对连线引用调用 delPathLine 函数， delPathLine 函数根据连线引用中的折线位置来删除特定的一段折线。

5.6 拓扑数据可视化模块

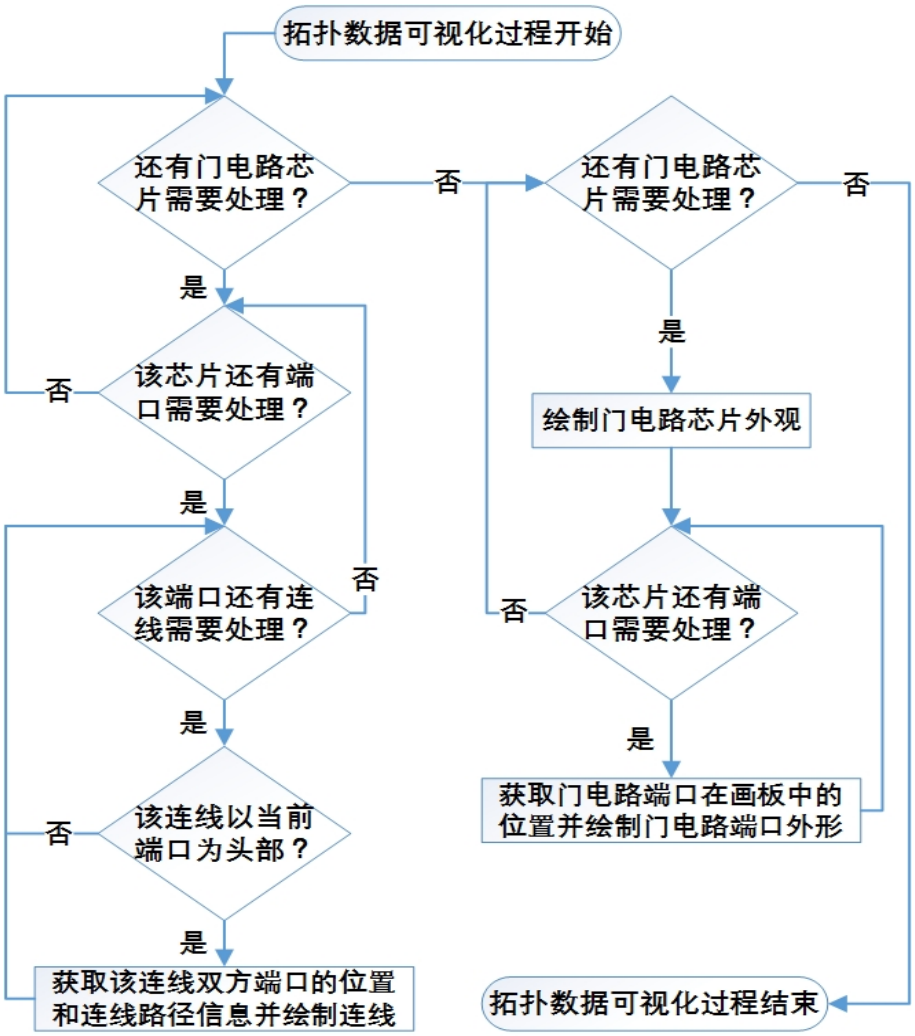


图 5.14 拓扑数据可视化过程流程图

5.6.1 门电路芯片可视化过程的设计与实现

如图 5.14 右半边部分，当拓扑因用户各类操作发生变化时，就会激发画板的重绘操作。 画板的重绘操作会遍历所有的门电路芯片， 并对每个门电路芯片都调用一次 drwHostGrap 函数来把每个门电路芯片的外观绘制到画板上。 drwHostGrap 函数会根据门电路芯片的位置参数和旋转参数、颜色参数和限制参数来绘制门电路芯片外观。

5.6.2 门电路端口可视化过程的设计与实现

如图 5.14 右半边部分，当拓扑因为用户各类操作发生变化时，就会激发画板的重绘操作。画板的重绘操作会遍历所有的门电路芯片， 并对每个门电路芯片都调用一次 drwHostGrap 函数来把每个门电路芯片绘制到画板上。由于门电路端口属于门电路芯

片的一个组成部分，所以 drwHostGrap 函数除了绘制门电路芯片外观，还会调用门电路芯片上的 drwPlugGrap 函数绘制门电路芯片上的所有端口。 该 drwPlugGrap 函数会遍历该门电路芯片上的所有端口，并对每个端口都调用它们自身 drwPlugGrap 函数。端口上的 drwPlugGrap 函数会首先通过 getPlgPoint 函数来获得每个端口的坐标位置，然后再根据这个坐标来绘制端口的可视化表示。getPlgPoint 函数会根据所属门电路芯片的坐标位置和旋转参数、端口在芯片上的所属边、以及端口在该边所对应端口数组中的位置这四方面参数来得到端口在画板中的实际坐标位置。

5.6.3 门电路连线可视化过程的设计与实现

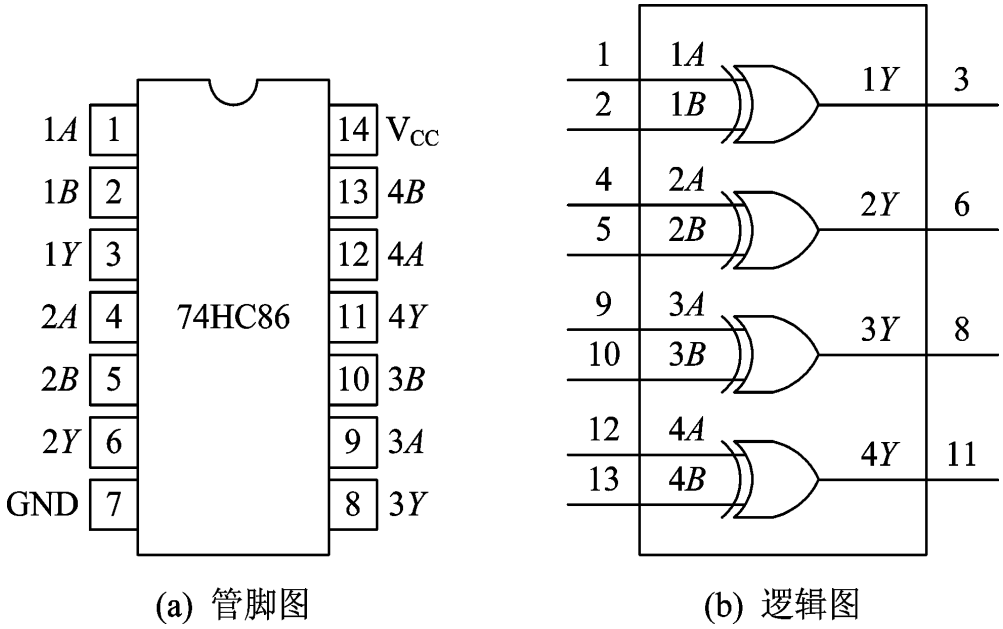
如图 5.14 左半边部分， 当拓扑为用户各类操作发生变化时， 就会激发画板的重绘操作。画板的重绘操作会遍历所有的门电路芯片， 并对每个门电路芯片都调用一次 drwPlugConn 函数来把每个芯片上的部分连线绘制到画板上。 芯片上的 drwPlugConn 函数会遍历门电路芯片上的所有端口，并对每个端口调用它们自身的 drwPlugConn 函数。门电路端口上的 drwPlugConn 函数会根据门电路端口的坐标位置和连线的轨迹路径参数来绘制连线。 这里要注意的是，并不是所有连线都会被绘制，而是连线的头部为当前处理的端口时，才会被绘制。 每条连线的两端要么一端为头另一端为尾，要么一端为尾另一端为头，这样做是为了每条连线都只被绘制一次，从而提高绘制效率。

6 系统仿真实验案例设计

6.1 基本门电路实验案例

6.1.1 实验拓扑的结构设计

本节以 74HC86—2 输入异或门为例， 使用该数字逻辑仿真实验系统来对其逻辑特性进行仿真。74HC86—2 输入异或门的管脚图和逻辑图见下图 6.1。



(a) 管脚图　　(b) 逻辑图

图 6.1 74HC86—2 输入异或门管脚图和逻辑图

由于 74HC86 芯片提供了 4 个异或门，因此本节仅使用第一个异或门来做仿真实验。根据图 6.1，我们可以得到系统仿真实验的拓扑结构，见图 6.2。

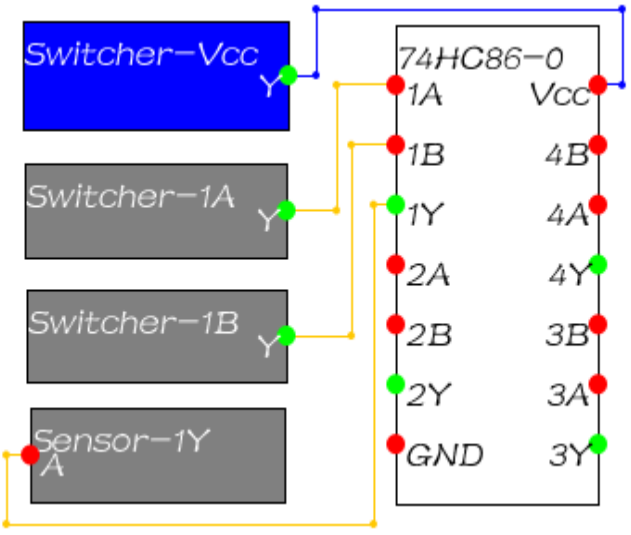


图 6.2 74HC86—2 输入异或门仿真实验拓扑结构图

6.1.2 实验拓扑的仿真结果

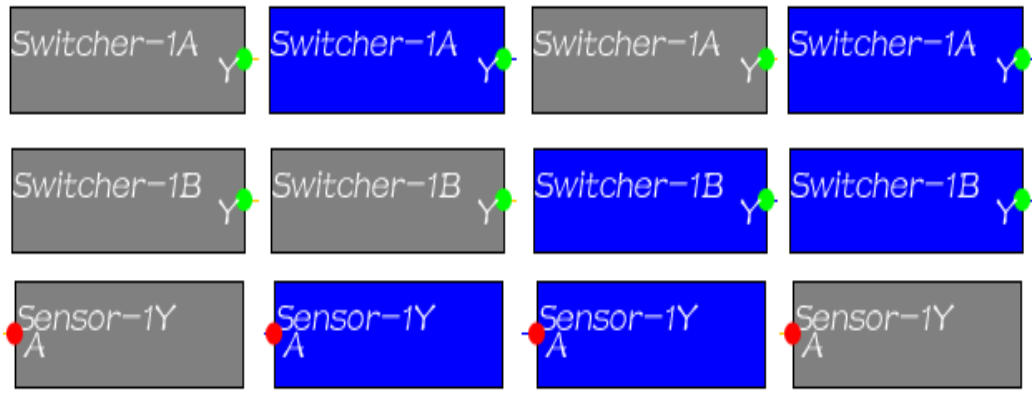


图 6.3 74HC86—2 输入异或门仿真实验结果图

从图 6.3 可以看出， 只有 Switcher-1A 和 Switcher-1B 的真值相同时， Sensor-1Y 的真值才变为 0，而其余情况均保持 1。这与 74HC86—2 输入异或门的逻辑特性是相符的。

6.2 组合电路实验案例

6.2.1 实验拓扑的结构设计

本节以 74HC138—3 线-8 线译码器为例，使用该数字逻辑仿真实验系统来对其逻辑特性进行仿真。74HC138—3 线-8 线译码器的管脚图及逻辑图见下图 6.4。

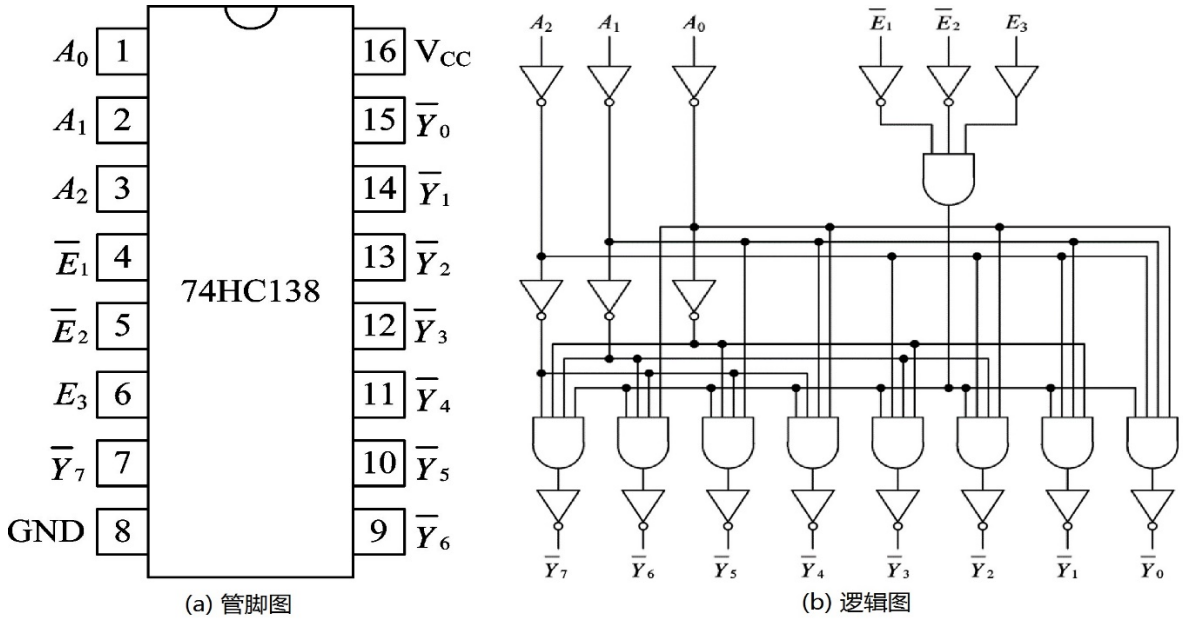


图 6.4 74HC138—3 线-8 线译码器管脚图和逻辑图

根据图 6.4，我们可以得到系统仿真实验的拓扑结构，见图 6.5。

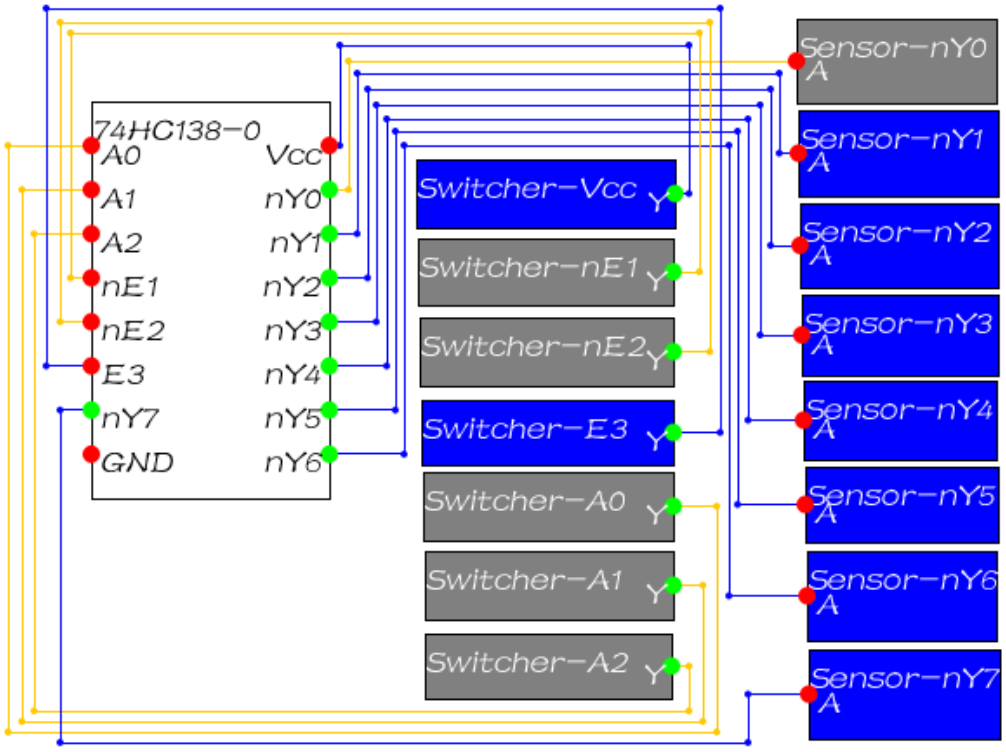


图 6.5 74HC138—3 线-8 线译码器仿真实验拓扑结构图

6.2.2 实验拓扑的仿真结果

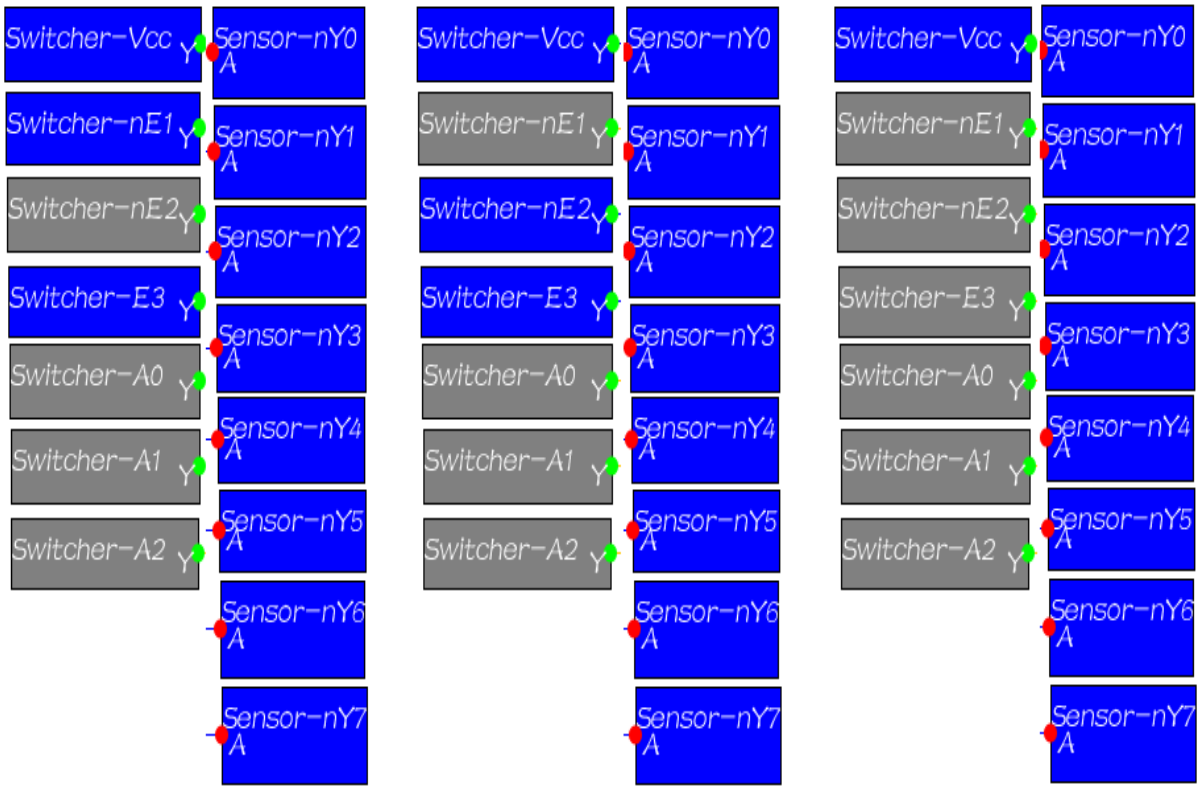


图 6.6 74HC138—3 线-8 线译码器仿真实验结果图 1

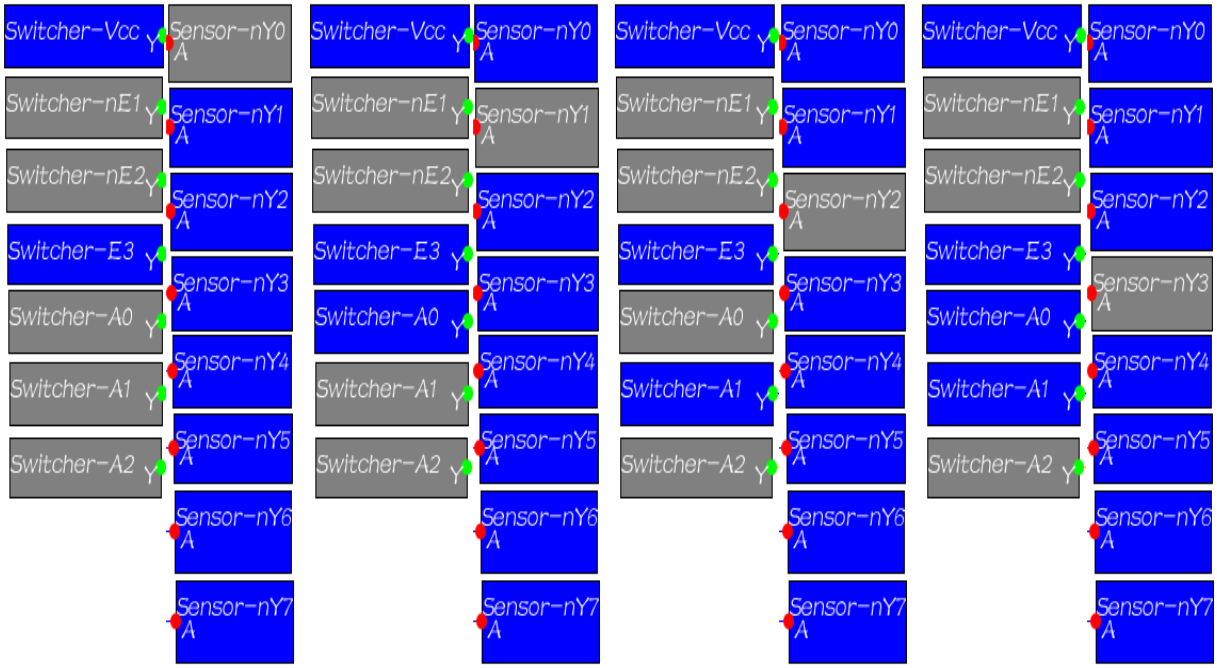


图 6.7 74HC138—3 线-8 线译码器仿真实验结果图 2

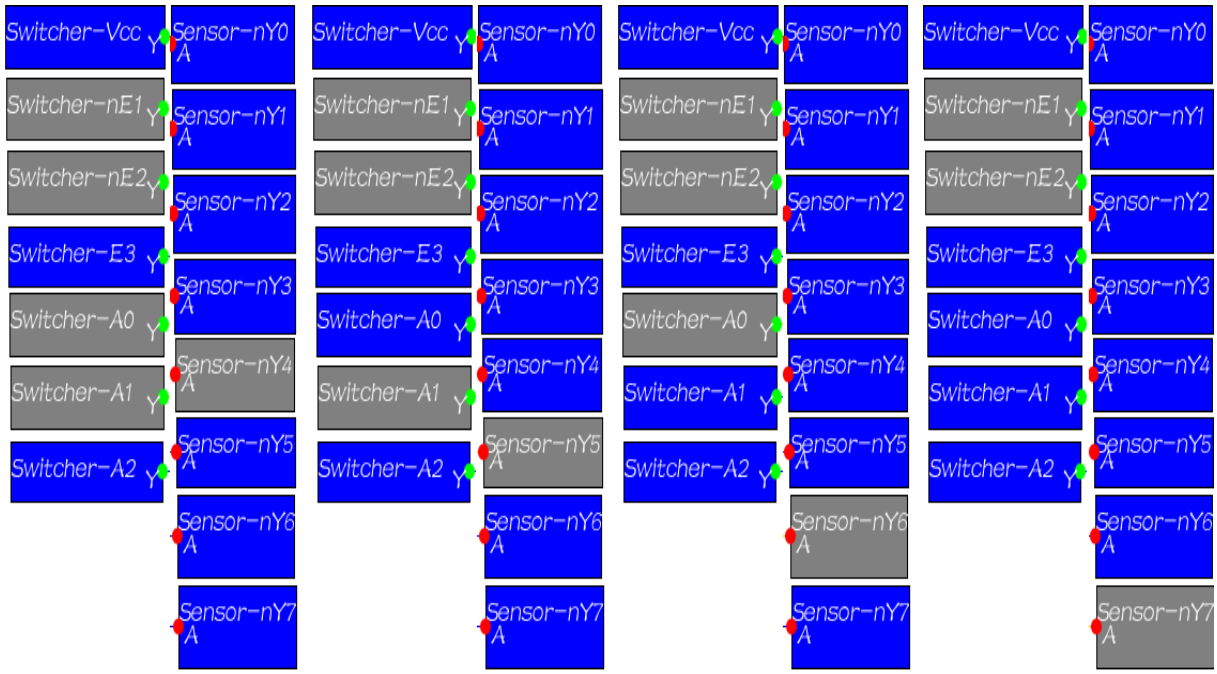


图 6.8 74HC138—3 线-8 线译码器仿真实验结果图 3

如图 6.6、6.7 和 6.8 所示， 只有当 Switcher-nE1、Switcher-nE2 和 Switcher-E3 的真值为 001 时，Switcher-A2、Switcher-A1 和 Switcher-A0 的真值状态才会影响译码器的输出。当 Switcher-A2、Switcher-A1 和 Switcher-A0 的真值分别为 000、001、010、011、100、101、110 和 111 时，Sensor-nY0、Sensor-nY1、Sensor-nY2、Sensor-nY3、Sensor-nY4、Sensor-nY5、Sensor-nY6 和 Sensor-nY7 的真值分别变为 0， 而其余仍保持为 1。 这与 74HC138—3 线-8 线译码器的逻辑特性是相符的。

6.3 时序电路实验案例

6.3.1 实验拓扑的结构设计

本节以 74HC161 芯片设计十二进制计数器为例， 并使用同步置位方式置零来仿真实验[2]。该方式原理在于它当计数值达到某个值时， 通过置数方式将$D_3 \sim D_0$ (其值为 0000)并行输入至计数器。 当输出为$Q_3Q_2Q_1Q_0 = 1011$时产生同步置位信号，而该置位信号将在下个时钟脉冲到来时产生置位操作。当输出端状态$Q_3Q_2Q_1Q_0 = 1011$时使$\overline{PE} = 0$，即$\overline{PE} = \overline{Q_3^n Q_1^n Q_0^n}$。同时使进位$C = 1$，即$C = Q_3^n Q_1^n Q_0^n$。 由上述置零逻辑及进位逻辑，可画出由 74HC161 及门电路构成的十二进制计数器的逻辑图，见图 6.9。

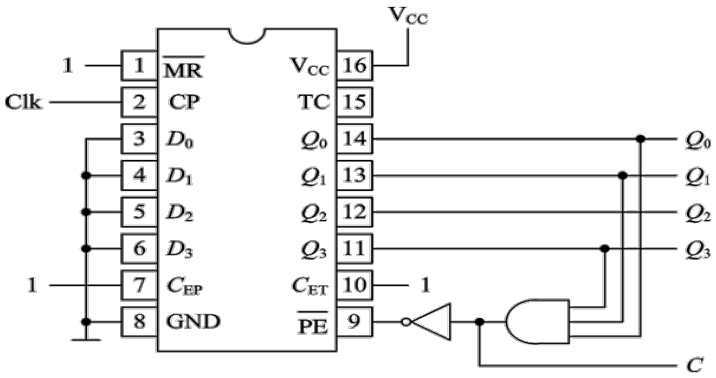


图 6.9 74HC161 及门电路构成的十二进制计数器的逻辑图

根据图 6.9，我们可以得到系统仿真实验的拓扑结构，见图 6.10。

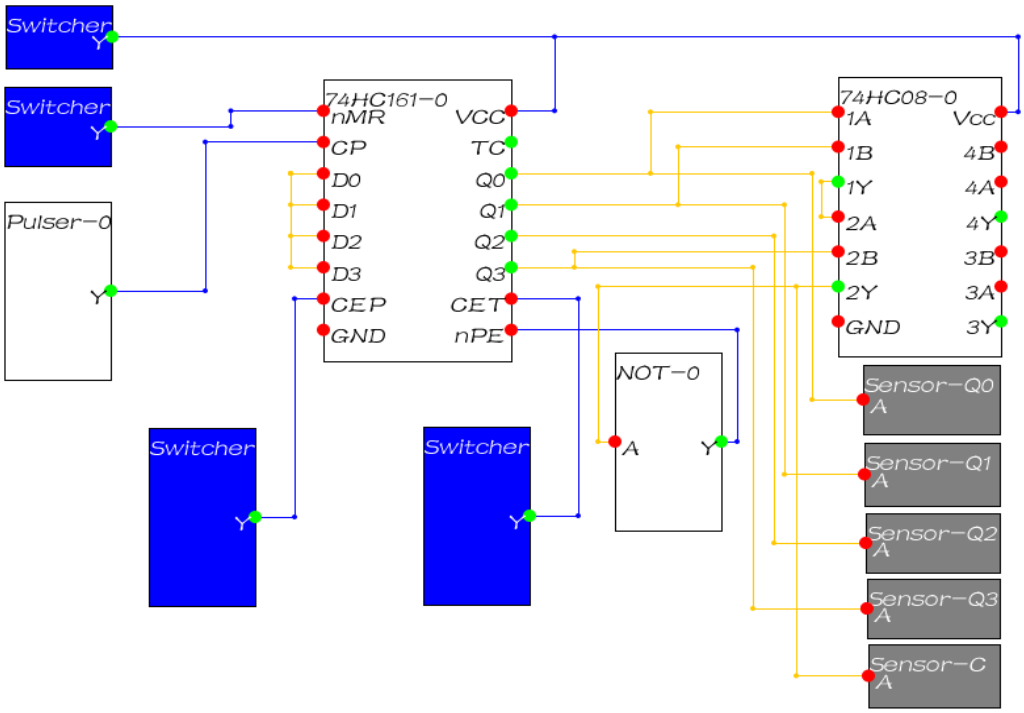


图 6.10 74HC161 及门电路构成的十二进制计数器的仿真实验拓扑结构图

6.3.2 实验拓扑的仿真结果

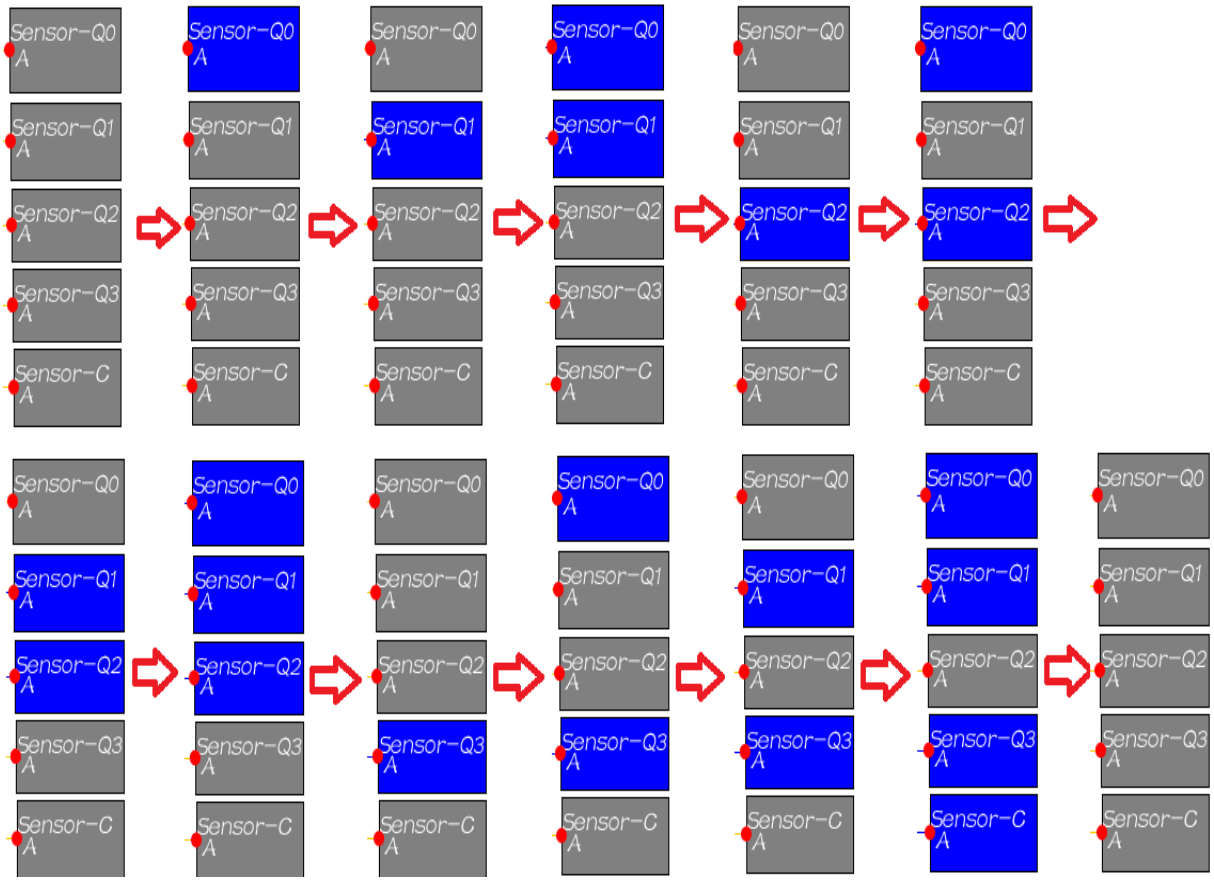


图 6.11 74HC161 及门电路构成的十二进制计数器的仿真实验结果

从图 6.11 仿真实验结果来看， 随着时序脉冲的作用下，Sensor-Q3、Sensor-Q2、Sensor-Q1 和 Sensor-Q0 真值在 0000 到 1011 之间变化，并在 Sensor-Q3、Sensor-Q2、Sensor-Q1、Sensor-Q0 真值为 1011 时 Sensor-C 真值从 0 变为 1。之后一直循环上述状态变化过程。 可见该数字逻辑仿真实验系统能够对 74HC161 及门电路构成的十二进制计数器进行仿真并能够得到正确的仿真实验结果，从侧面进一步验证了该数字逻辑仿真实验系统的可行性和正确性。

结论

通过前期辛苦的文献资料收集和后期长时间的摸索研究和开发，最终实现了一套基于 Java 语言的较为完整实用的数字逻辑仿真实验系统。 该系统各项功能都达到了项目最初的设计要求和功能，并在此基础上添加了一些人性化的细节功能来完善产品功能，提升了用户的使用体验。另外，系统还添加了一些数据接口以供外部组件或插件的使用，提高了系统的可扩展性。

论文从课题的研究背景和意义说起，介绍了国内外对该类课题的发展情况。通过分析初学者的数字逻辑课程实验操作所遇到的问题，和国内外该类课题研究成果对初学者实行数字逻辑课程实验的支持程度，我得出了该类课题研究的不足之处和需要去攻克的技术实现。该论文在介绍完该课题系统所需要的技术支持后，对系统的设计原则、需求分析、模块功能分析和可行性分析四方面进行了详细讨论。其中系统需求分析对学生和教师所需系统功能进行了分析，系统模块功能分析对门电路芯片模块、门电路端口模块、门电路连线模块、拓扑导入和导出模块等系统各功能模块进行了功能划定。在系统设计方面，系统引入了仿真内核层、操作交互层和数据可视化层来对系统进行更大范围的功能划分，以确定各系统功能模块在系统架构中所处的功能地位。在系统实现方面，分别对各功能模块的实现细节进行了理论上或算法上的分析并附以流程图和表格来帮助设计相应的功能代码，从而令项目得以顺利实现。

在整个开发过程中，按照文档的设计进行开发并实现了文中所列举的所有功能，并且在细节上添加了一些具备一定创新性的设计，比如：用户可以根据需要自行定义芯片逻辑功能，逻辑表达式中引入特殊符号来限定芯片的触发模式，引入一些机制来达到连线防环路功能的实现。该数字逻辑仿真实验系统虽然已经能够让初学者实行很多数字逻辑实验，但仍需对该数字逻辑仿真系统做进一步研究开发工作：

1. 添加更多运算符以提高编写逻辑表达式的灵活性；
2. 添加更多的用户交互操作到画板窗口以提高用户操作拓扑的灵活性；
3. 添加更多的可视化组件以提高拓扑数据可视化形式的灵活性；

该系统的创意与功能已得到数字逻辑课程组教授的认可，其认为相比于使用课程实验机箱来完成相关的实验，使用该系统来完成实验具备更好的可靠性和使用体验。同时该系统也方便了教师布置课程实验。现已准备向学生和教师推广该款系统。

参考文献

[1] 丁磊，冯永晋，张海笑 . 数字逻辑与 EDA 设计实验指导书[M] . 西安：西安电子科技大学出版社, 2012：18-19

[2] 丁磊，冯永晋，张海笑 . 数字逻辑与 EDA 设计[M] . 西安：西安电子科技大学出版社, 2012：219-220,124-125

[3] 4 种电路仿真软件比较[EB/OL] . http://my.oschina.net/quanpower/blog/113880, 2013-03-15

[4] James Gosling, Bill Joy, Guy Steele, Gilad Bracha . The Java Language Specification[M]. Addison-Wesley, 2005-06-01

[5] Y.Daniel Liang . Introduction to Java Programing, Eighth Edition[M]. Prentice Hall，2010：406

[6] MyEclipse 的功能特点[EB/OL] . http://www.myeclipsecn.com/features/, 2015

[7] 王文彦 . 计算机网络实践教程:基于 GNS3 网络模拟器(CISCO 技术)[M] . 北京：人民邮电出版社，2014：7-11

[8] Brain W.Kernighan, Dennis M.Ritchie. The C Programming Language[M] . Prentice Hall, 1988：207-208

[9] Alfred V.Aho, Monica S.Lam, Ravi Sethi, Jeffrey D.Ullman . Compilers:Principles, Techniques, and Tools, Second Edition[M] . Addison-Wesley, 2006：53-60

[10] C. Hedrick . RFC-1058: Routing Information Protocol[S] . Network Working Group, 1982：5-15

[11] 谢希仁 . 计算机网络(第 6 版)[M] . 北京：电子工业出版社，2013：152-157

[12] 明日科技 . Java 从入门到精通[M]. 北京：清华大学出版社，2012：267-270

致谢

我的毕业论文是在张静老师的精心指导和大力支持下完成的，她渊博的知识开阔的视野给了我深深的启迪，论文凝聚着她的血汗，她以严谨的治学态度和敬业精神深深的感染了我对我的工作学习产生了深渊的影响，在此我向她表示衷心的谢意。

附录 A 系统运行环境部署说明

该系统使用 Java 语言开发以 Jar 文件形式发布，因此在使用前请先配置 JRE 环境。JRE 环境的部署流程见下方：

1. 首先到 http://www.oracle.com 官方网站下载 JRE 环境安装包；
2. 然后在机器上安装此 JRE 环境安装包；
3. 安装完成后即可双击该系统可执行 Jar 文件来运行系统。

附录 B 系统使用说明

1. 门电路芯片的添加操作

首先在右边的门电路芯片素材窗口中用鼠标左键点击所需门电路芯片（这里以与门为例），随之与门按钮变蓝色，并在画板左上角显示该芯片，如图 B.1 所示。

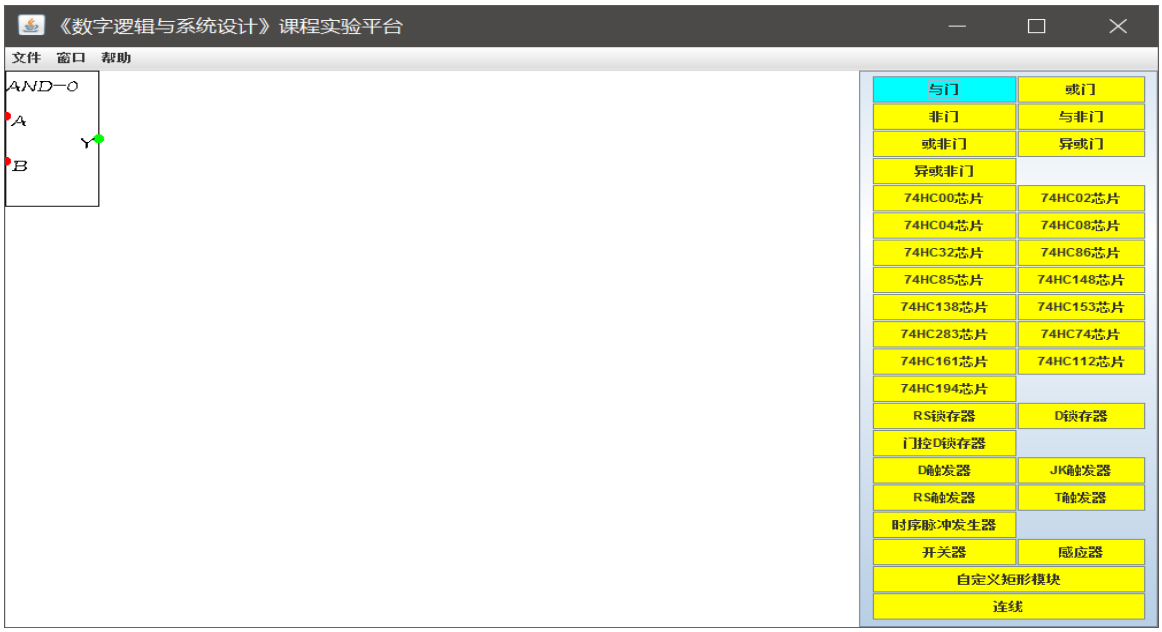


图 B.1 门电路芯片的添加操作示意图 1

然后把鼠标移动到画板区域，此时可看到该芯片正在跟随鼠标移动。最后在画板中选择合适的位置按下鼠标左键，此时该芯片被正式放置于画板中，如图 B.2 所示。

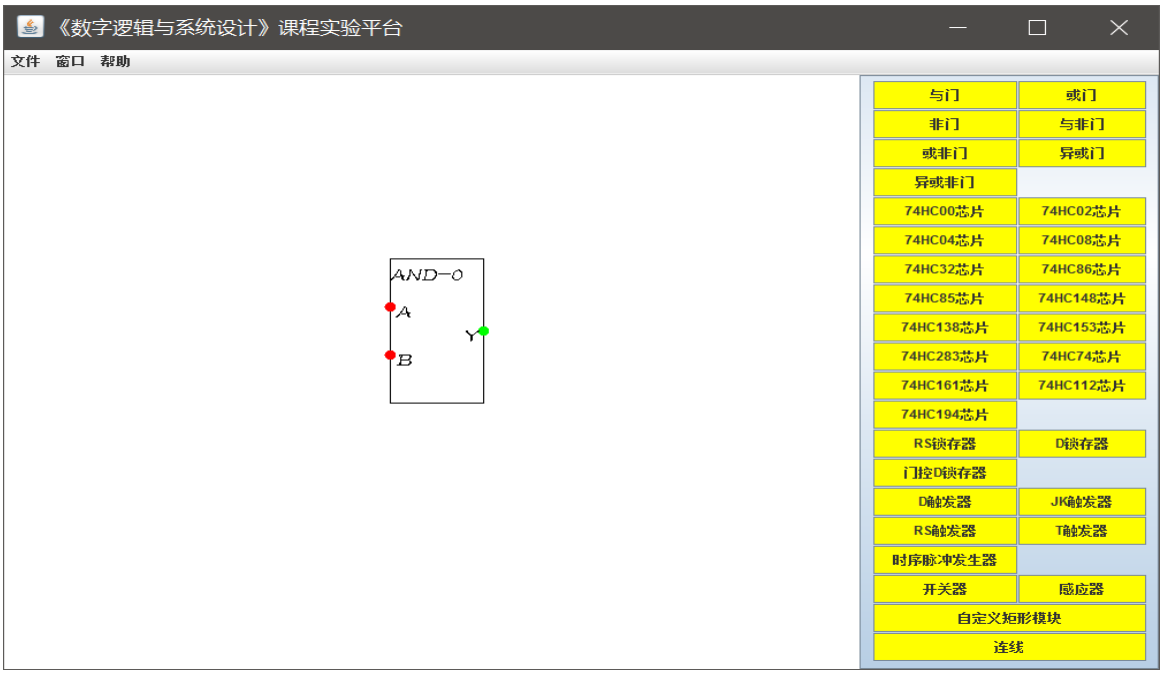


图 B.2 门电路芯片的添加操作示意图 2

2. 门电路端口的连线操作

首先在右边的门电路芯片素材窗口中用鼠标左键点击“连线”按钮，随之标识连线的按钮变蓝，这表明画板模式已转为连线模式，如图 B.3 所示。

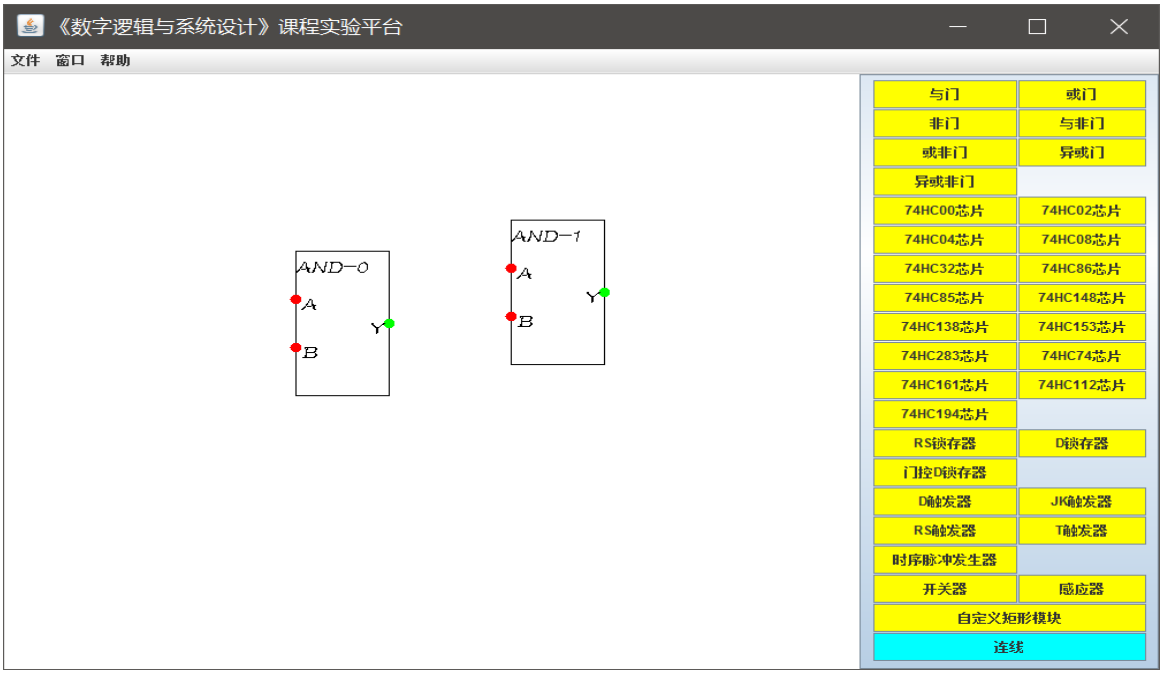


图 B.3 门电路端口的连线操作示意图 1

接着鼠标左键点击画板中的任意一个芯片，此时弹出该芯片中可选择的端口列表，如图 B.4 所示。

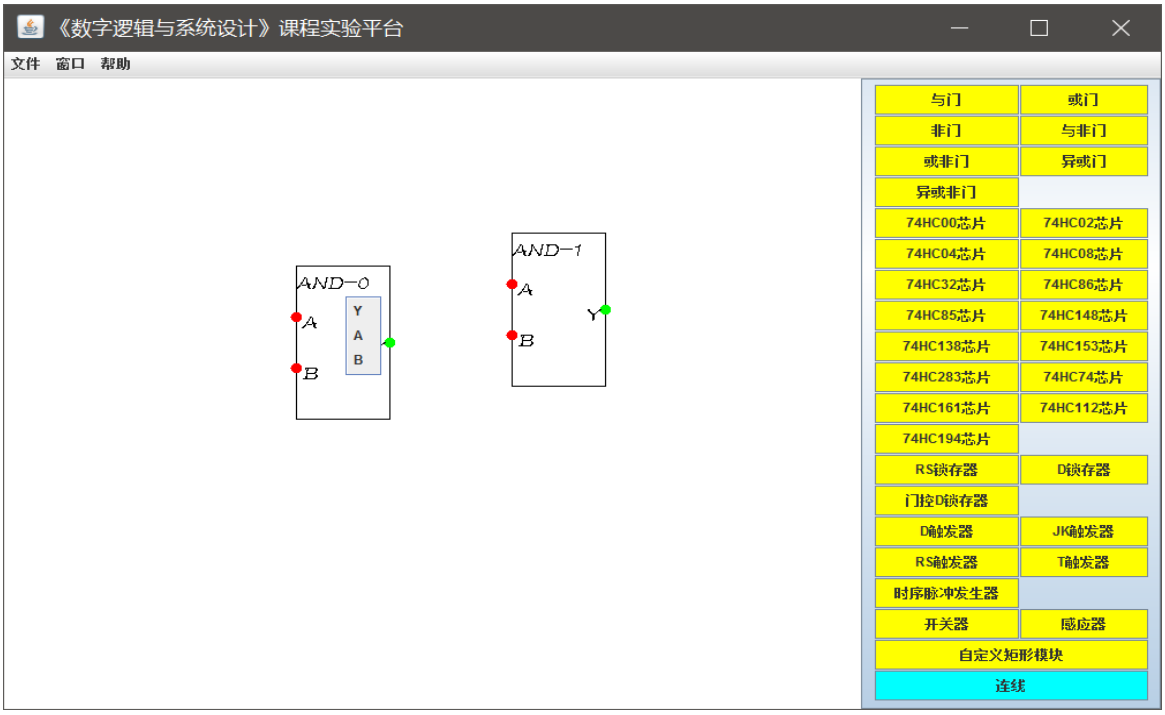


图 B.4 门电路端口的连线操作示意图 2

根据该端口列表，鼠标左键点击选中该芯片的某个端口，随之一条以端口位置为始点以鼠标位置为终点的线段随鼠标游走，如图 B.5 所示。

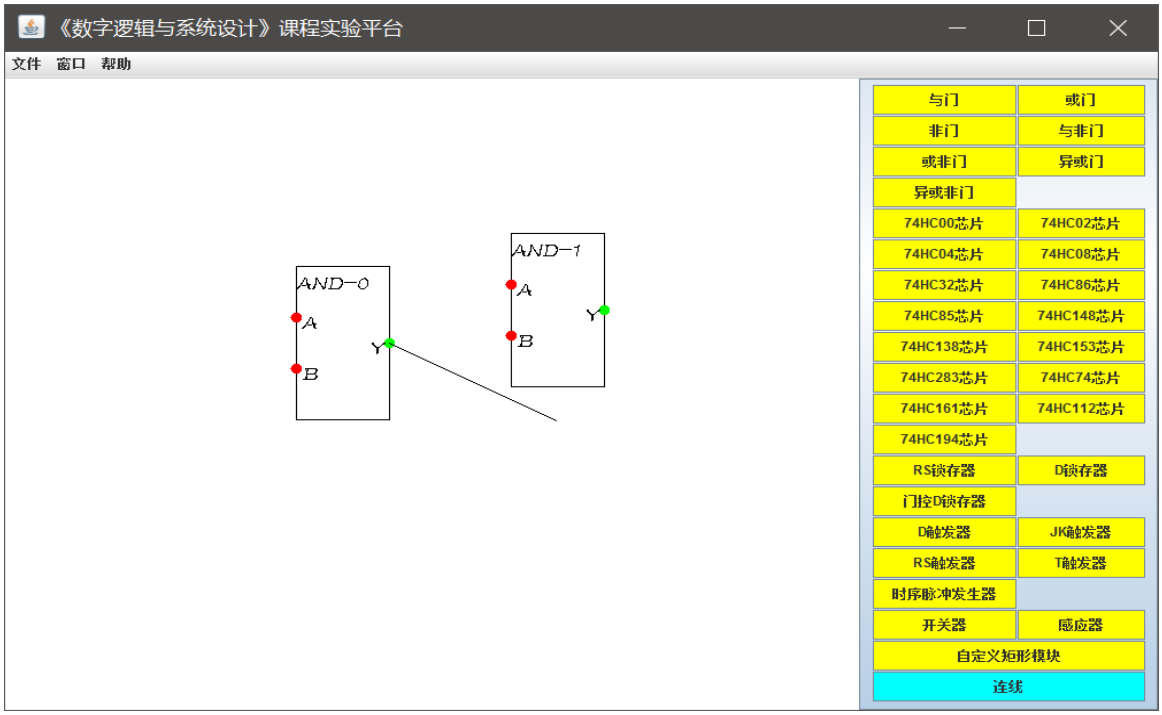


图 B.5 门电路端口的连线操作示意图 3

然后把鼠标移动到另一个芯片上并点击，弹出芯片可选择的端口列表，如图 B.6。

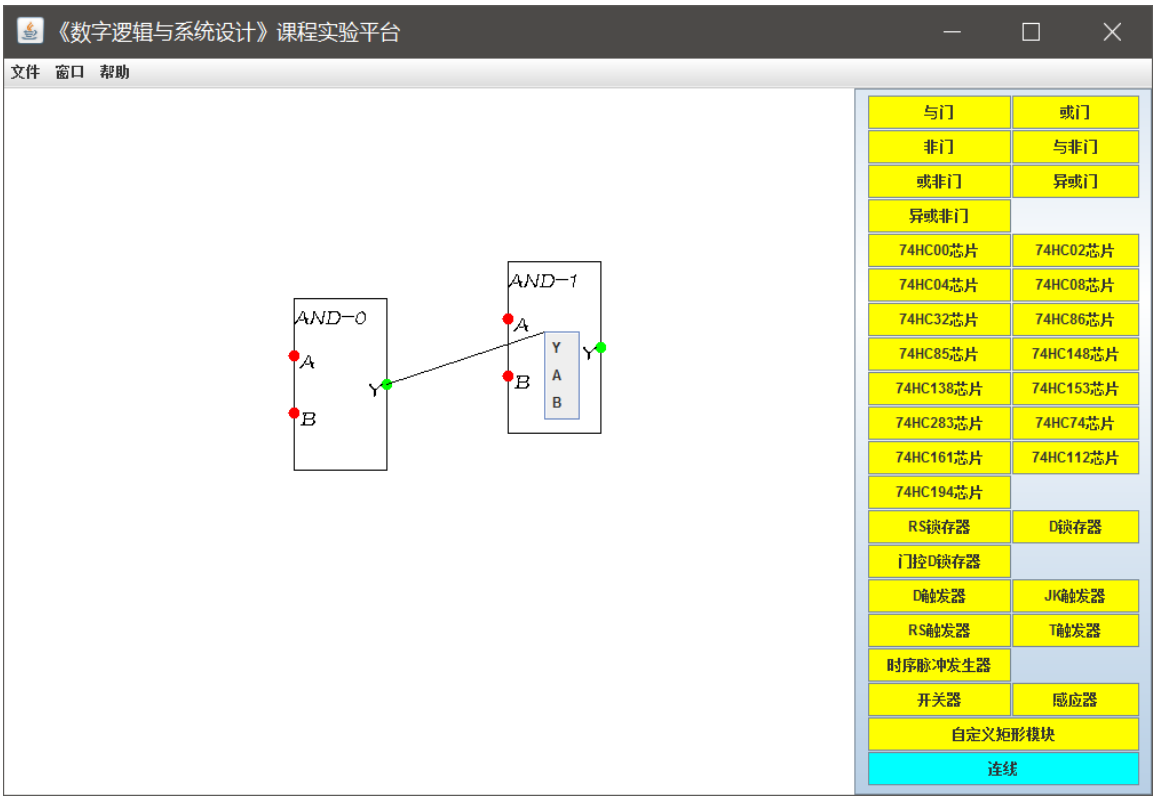


图 B.6 门电路端口的连线操作示意图 4

最后根据该端口列表，鼠标左键点击选中该芯片的某个端口，随之两个端口就连上了，如图 B.7 所示。

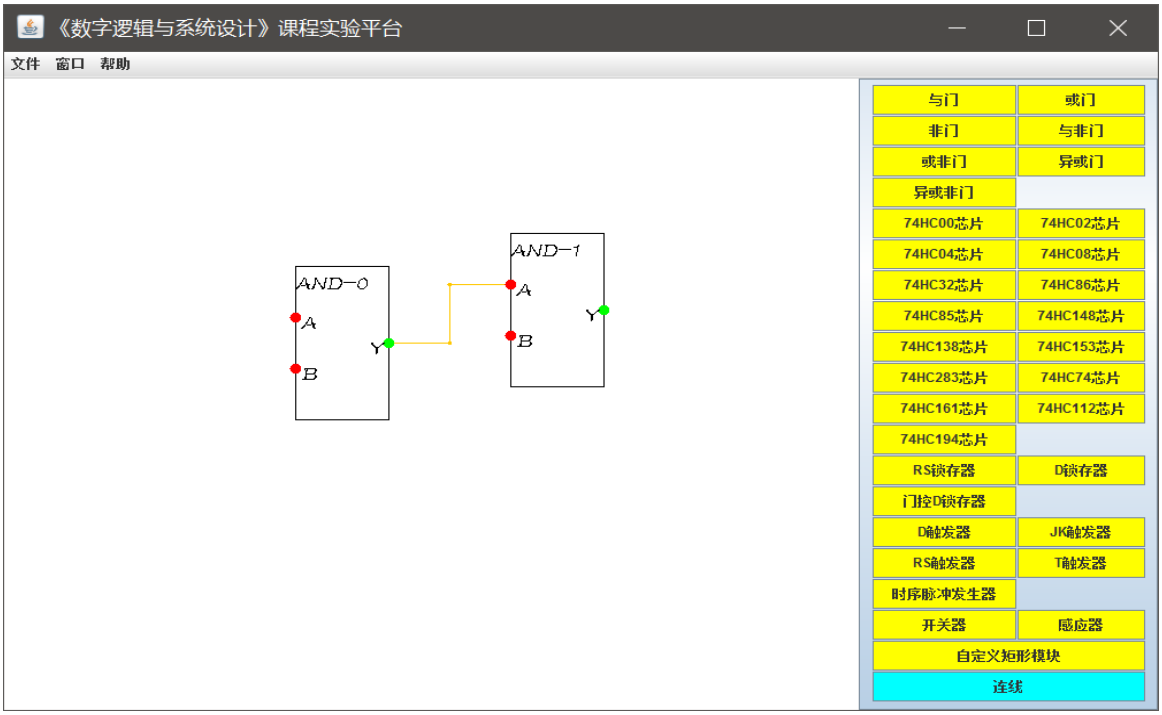


图 B.7 门电路端口的连线操作示意图 5