文章编号:1671-4598(2021)05-0150-04 DOI:10.16526/j.cnki.11—4762/tp.2021.05.030 中图分类号:TP273 文献标识码:A

基于 Qt 的集成电路测试软件设计与实现

孙好婕，赵利强，郑惠泽，孙振山
（北京化工大学 信息科学与技术学院，北京 100029）

摘要：集成电路测试在集成电路生产中具有重要的作用，针对集成电路测试软件不易跨平台使用的问题，基于 Qt 设计实现了一款采用视图/委托/模型框架的集成电路测试软件，给出了测试参数配置、测试程序运行及测试结果管理 3 部分功能的设计与实现方法；使用 Lua 开发了软件二次接口，利用多线程技术设计了软件运行引擎，并且采用标准测试数据格式实现了测试数据的存储；实验表明，所设计的软件能够进行集成电路测试并且具有良好的可扩展性和通用性。

关键词：集成电路测试；视图/委托/模型；Lua；多线程；标准测试数据格式

Design and Implementation of Integrated Circuit Test Software Based on Qt

Sun Yujie, Zhao Liqiang, Zheng Huize, Sun Zhenshan
(College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China)

Abstract: Integrated circuit test plays an important role in the integrated circuit production. Aiming at the problem that integrated circuit test software is not easy to be used across platforms, an integrated circuit test software using view/delegate/model (MVD) framework is implemented based on Qt. The design and implementation methods of test parameter configuration, test program running and test result management are given. The secondary interface of the software is developed by Lua. The software running engine is designed by using multi-thread technology, and the Standard Test Data Format (STDF) is adopted to realize the storage of test data. The experiment shows that the software can carry out integrated circuit test and has good expansibility and universality.

Keywords: integrated circuit test; MVD; Lua; multithread; STDF

0 引言

集成电路测试是集成电路生产环节中的最后一道生产工序，为了保证集成电路的质量，测试在集成电路生产中具有非常重要的作用[1]。而测试软件则是测试系统的核心，肩负着驱动整个测试系统硬件模块的重要使命，是连接测试资源和被测器件的必不可少的中间介质[2]。

目前，美国、日本等国家的集成电路自动测试软件 IG-XL、SmartTest 等具备了较为成熟的技术，均支持通用格式 STIL 和 STDF 并且提供软件二次接口，具有良好的通用性和可扩展性，可以方便、快捷地完成测试程序的开发[3]。但其大部分基于 Windows 操作系统。目前国内集成电路软件的开发与研究也处于发展之中，曹菲[4]基于 PCIe 总线技术设计并实现了一种通用数字集成电路测试系统，该测试系统提供直流、交流、功能测试等多种测试需求，但测试软件的测试程序是由测试码生成的，存在对于使用者的编程能力有较高的要求的缺点。戴春翟[5]等设计了一款集成电路通用测试软件，利用多层次、模块化的软件结构设计方法，令参数和测试程序互相分离，提高了测试的效率，但该软件在测试数据存储方面不支持通用的 STDF 格式存储[5]。以上集成电路测试软件的开发都是基于 Windows 系统的，对于硬件仪器的使用，只能调用 Windows 平台支持的仪器驱动，具有一定的局限性，不能很好地满足测试软件的跨平台测试需求。

Qt 是一种 C++图形用户界面应用程序开发框架，可在 Windows、Linux 等平台下运行，并且具有良好的 UI 特性和丰富的 API[6]，可用来设计良好的界面并且满足跨平台的运行需求，因此可应用于集成电路测试软件的开发。

本文基于 Qt 设计实现一款集成电路测试软件。给出了测试参数配置、测试程序运行及测试结果管理 3 部分功能的设计与实现方法，并进行了实验测试。

1 集成电路测试软件架构设计

集成电路测试软件由测试程序开发、测试程序运行和测试数据管理组成的。在集成电路的测试过程中会涉及到直流参数、功能测试等多种测试需求。测试中首先要对被测芯片的引脚信息进行定义，然后设置测试负载板上相应通道编号，将引脚通过测试负载板与测试仪器实现连接。同时，需要实现对测试机器资源的分配，即控制测试硬件产生必需的电压、波形和时序需要的信息。最重要的一部分还涉及到测试向量数据，包含施加到被测器件输入端逻辑状态和输出端期望逻辑状态。测试向量通常由仿真文件（如 VCD、WGL 等类型）通过自动转换软件将仿真文件转

收稿日期:2020-10-12； **修回日期**:2020-10-29。

作者简介:孙好婕(1996-)，女，河北沧州人，硕士研究生，主要从事集成电路测试软件方向的研究。
通讯作者:赵利强(1982-)，男，河南孟津人，副教授，博士，主要从事智能检测与传感技术方向的研究。

引用格式:孙好婕，赵利强，郑惠泽，等. 基于 Qt 的集成电路测试软件设计与实现[J]. 计算机测量与控制，2021，29(5)：150-153，168.

换而来，也可由测试工程师编写来完成[7]。

集成电路测试中需要对上述种类繁多，关系复杂的参数进行处理，因此采用界面层、引擎层和数据层 3 层架构对软件进行设计。其中，采用模型/视图/代理（MVD）的框架进行软件界面层的设计。MVD 是 Qt 在 Model View Controller（MVC）的基础上引入模型/视图结构得来的[8]，MVC 是一种业务逻辑、数据、界面显示分离的方法组织代码的方式，MVD 中把视图和控制部件结合在一起，使得框架更为简洁，通过使用该框架，能够有效地分离数据和界面。测试软件架构图如图 1 所示。

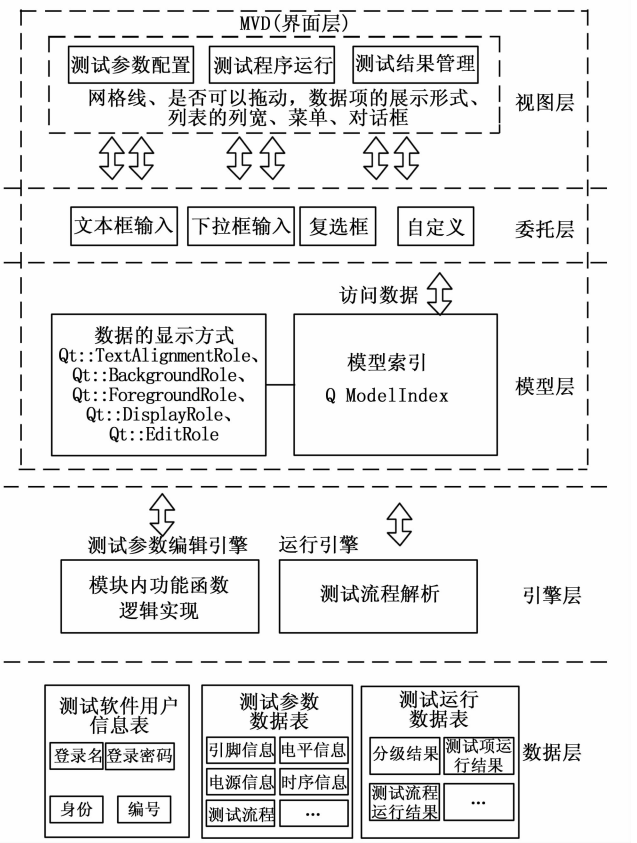

图 1　测试软件架构图

1）软件的最上层也就是界面层，界面层的最上层是视图层，用于管理用户和软件的交互，利用 Qt 的 UI 特性，提供给用户良好的操作使用界面，用户可以通过在界面上对控件进行操作从而对测试参数进行配置和修改。测试参数配置、测试程序运行以及测试结果管理 3 部分的界面均采用列表的形式进行展现，其中所涉及到的网格线、是否可以拖动，数据项的展示形式、列表的列宽、菜单的内容和样式以及会使用到对话框的样式和内容都在视图层进行管理，均采用 QTableView 表格视图来实现。

委托层处理的也是用户的输入，也就是把用户的输入委托给 Qt 的某个部件进行处理。对于在集成电路测试过程中需要用户输入的测试参数信息采用文本框输入的方式进行处理，对于给出用户可选项的信息采用下拉框的方式进行处理，对于可选的测试信息采用复选框的方式进行处理，此外还可以自定义委托，当用户处理完这些输入之后，将数据发送给数据模型层。

模型层有 3 个主要功能，首先，与引擎层进行通信，获取数据信息；其次，开放数据访问接口，委托层和视图层通过模型索引访问数据模型层的数据；其三，该层还设置了数据的显示方式，在集成电路测试软件界面中可以被编辑的数据设置为 Qt::EditRole，需要以文本形式显示的数据则被设置为 Qt::DisplayRole。

2）引擎层的测试参数编辑引擎定义了数据模型层功能函数中真正的逻辑实现，当数据模型层的函数被触发时会调用引擎层定义的逻辑功能。运行引擎则实现了测试流程的解析。

3）数据层则以数据表的形式实现了整个测试过程中所涉及到数据的存储，分别对测试软件用户信息、测试参数信息、测试运行信息以及测试结果信息进行存储。数据库层直接和引擎层进行通信，数据库层会提供数据给引擎层，引擎层最终对测试参数进行的修改会被更新到数据库层的数据库中。

2　集成电路测试软件的设计与实现

2.1　测试参数配置设计与实现

测试参数配置需要完成对被测芯片的描述以及进行配置测试过程中所需要的参数，这部分是完成测试的基础。

为方便用户的操作，测试参数配置部分采用模块化的设计，测试中所需要的被测管脚信息、管脚通道映射、时序信息、电源信息、电平信息、向量信息等在界面的显示上被分为不同的模块。测试参数配置给用户提供友好交互的向导界面，实现各模块参数的配置。所以在程序的实现过程中，视图层要提供设置测试内容的人机交互接口，需要进行配置的参数信息在界面上以列表的形式进行显示，这部分在数据模型层中进行设置。对于具体参数值的设置是用户进行输入的，因此视图层都是用于实现模块人机交互部分的响应，即实现用户点击的引起的信号触发以及对相应弹窗的触发，并获取用户输入内容。

2.1.1　模块间通信设计与实现

对于测试参数配置部分整体而言，其中的每个模块都不是独立的，很多模块间存在着一定的联系。为解决模块间存在关联的问题，在引擎层采用如图 2 所示的实现方式。信号管脚模块的 CPin 类当信号发生变动时通过 Connect PinGroup（CPinStore * pPinStore，std::vector <int> vecID，int nPin GroupID）接口函数通知信号库 CPinStore，当信号库发生变动会通知信号组库。当信号组 CPinGroup 发生变动也会通知信号组库 CPin Group Store，信号组库通过接口函数 Auto _ Connect（）连接信号库，监视信号库修改事件。管脚映射模块通过 CChannelBlockStore 的接口函数 Auto _ Connect（）连接信号库，监视信号的变化。时序配置模块、电源配置模块和电平配置模块分别通过 CTimingBlockStore 类、CPowerBlockStore 类、ClevelBlock-

Store 类的接口函数 Auto _ Connect（）连接信号库、变量库、信号组库，监视信号库、变量库和信号组库的变化。界面层会与引擎层进行通信，当对界面上对某个模块的信息进行编辑，该模块的信息发生变化，引擎层监控到信息的变化时，会通知界面层和数据层，对与该模块有关联的模块的界面和数据层进行更新。

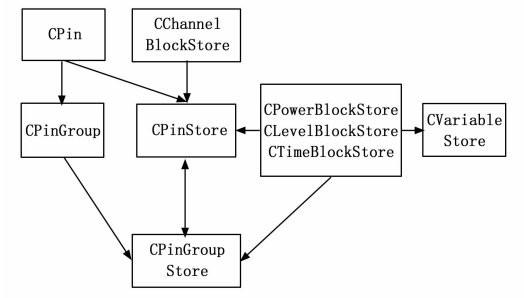

图 2 模块间关系图

2.1.2 软件二次接口开发

此外，由于用户会有个性化的测试需求，测试参数配置部分提供二次开发接口功能，设置为用户代码模块，在这个模块用户可实现对 API 函数的调用，通过将用户代码模块配置到测试项中，实现用户自定义的测试需求。

采用 Lua 作为用户代码的开发调试工具，因为 Lua 有着良好的可嵌入性[9]，将 Lua 嵌入 C++之后，如果对用户代码的修改均可以在 Lua 中进行，不用再重新编译 C++项目。集成电路测试软件开发平台作为宿主，Lua 作为嵌入式脚本，嵌入到宿主语言中，为测试软件提供自定义测试的功能，用 C++编写的 API 函数作为 Lua 的底层扩展库，是 Lua 脚本访问主要的功能函数。实现流程如图 3 所示。

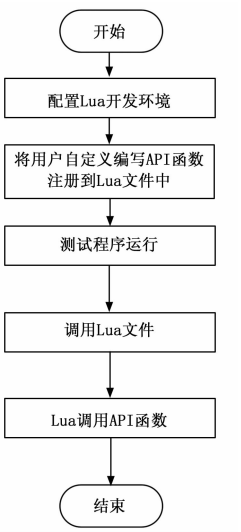

图 3 用户代码模块流程图

1）用户代码模块用于完成自定义测试功能的任务，该模块运行时会加载用户自定义的测试。

2）调用 Lua 语言提供的 Lua _ newstate 接口函数创建 Lua 解释器，调用 lua _ register（）将用户自定义编写的 API 函数注册到 Lua 文件中。

3）按照 Lua 语法将已经在 Lua 解释器中注册过的 API 函数编写成自定义测试需要的逻辑功能写到 Lua 脚本中，保存为 user. lua 文件。

4）将 user. lua 配置到用户代码模块中，当用户代码模块加载时，便会调用 Lua _ dofile 函数加载 Lua 文件，此时，Lua 文件运行触发 Lua _ getglobal 函数，在 Lua 解释器中被注册的 API 函数被调用，用户自定义的测试功能被实现。

2.2 测试程序运行设计与实现

测试运行部分实现对所测试芯片进行功能和性能测试。仿真运行部分调用仪器资源模型和 DUT 仿真模型，完成仿真测试；调试运行部分调用测试仪器的硬件设备驱动函数，通过计算机与测试仪器的通信接口向硬件端口发送控制命令，硬件电路解析命令使硬件电路完成相应的动作，并具有设断点调试等功能。

测试运行部分的视图显示共分为 3 部分：引脚通道配置的选择，测试工程的信息以及测试结果的显示。测试结果的显示包括测试项的测试结果和分级结果。在视图层，定义 4 个类 BinResultTableView、TestResult TableView、ProjectInfoTableView、Pakage InfoTableView 分别进行视图显示信息的设置，对于引脚通道配置的选择和测试流程的选择均在委托层采用下拉框的特殊形式进行处理，在模型层则定义 4 个类 BinResultData、TestResultTableData、ProjectInfoData、PakageInfoData 对表头等数据的显示形式进行定义并和引擎层进行通信。

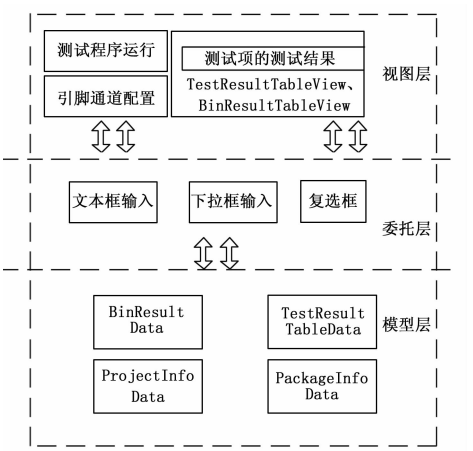

图 4 测试运行界面显示架构图

根据集成电路测试的需求，测试软件的运行过程中需要创建用户界面并进行测试的线程（主线程）和管脚映射下载线程（子线程）。主线程负责界面的显示和测试的进行，子线程负责管脚映射关系的下载，两个线程的运行是并列的不相互影响，因此，使用多线程技术可以提高软件的运行效率。

用户运行测试软件时，会建立一个进程同时为这个进程创建一个主线程。在进行下载管脚映射关系时，就需要再创建一个测试线程来完成测试工作。Qt 开启多线程，主

要用到类 QThread。用 CWorkThread 类继承 QThread，然后重写虚函数 run（）。当要开启新线程时，只需要实例该类，调用 LoadPackage（const std::string strRunFile，int nPackageID）函数触发 start（）函数间接调用 run（）函数来创建新的线程。当需要结束该线程时，调用 UnLoadPackage（），触发 Quit（）函数来删除非主线程管理的任务，结束该线程。

在多线程程序设计中，线程与线程之间进行通信是非常重要的。用户需要在界面中点击“下载”按钮来进行下载管脚映射关系，也就是说需要在用户界面中控制线程，为了实现这种功能，采用 Qt 自带的信号与槽的方式进行线程的通信。在主线程中定义 slot 函数，在子线程中定义 signal 函数，通过 connect 函数实现信号与槽函数之间的通信，从而实现主线程与子线程之间的通信。

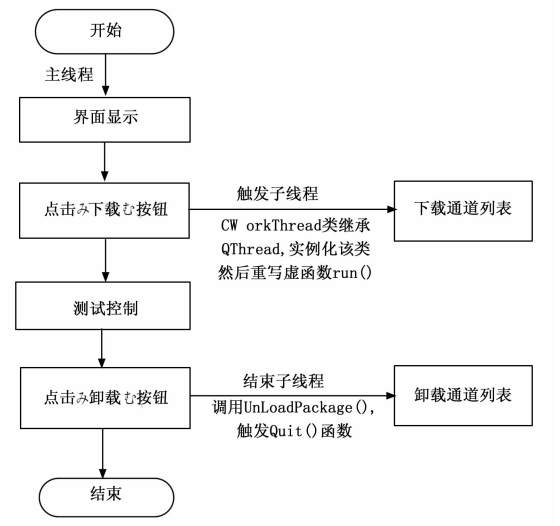

图 5　多线程实现流程图

2.3　测试数据管理设计与实现

测试数据管理模块实现对测试运行结果进行存储并提供测试结果的可视化显示功能。

为了提高软件的通用性，采用 STDF 格式对测试数据进行存储。STDF 文件由多个数据模块构成，V4 版本共计 25 个模块，不同的模块被称为不同的记录类型，每个模块有其固定的记录内容，这些模块按照一定的顺序，共同组合构成了整个 STDF 文件[10]。实现 STDF 存储的流程如图 6 所示。

基于 STDF 的格式特点，对 STDF 格式的存储也以数据模块为单位进行实现，每个数据模块都由以下三部分组成：记录标题、必选的数据记录以及可选的数据记录。

首先创建一个数据容器向量，命名为 Vector，Vector 用于存储数据模块的必选的数据记录以及可选的数据记录，将二进制数据转换成 ASCII 码之后依次存入 Vector，然后获取 Vector 的长度，因为记录标题占 4 字节的长度，所以将 Vector 的长度加 4 字节后存到数组 List 中，再根据数据模块的类型确认记录标题的内容后将记录标题存入 List，List 中便是 STDF 中数据模块的内容，将各个模块按照规定的顺序写入 STDF 文件，便完成了文件的存储。

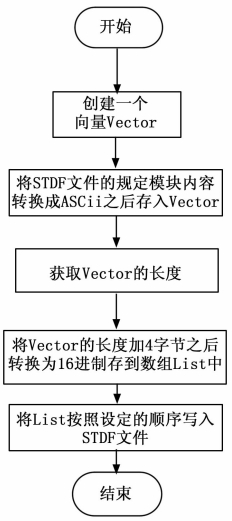

图 6　STDF 格式存储流程图

3　实验结果与分析

在软件的功能基本开发完成后，以二进制计数器芯片 M54HCT160 作为被测芯片，对软件的仿真运行功能进行了测试。

1）新建工程：

启动软件进入系统登录界面，输入正确的用户名和密码，点击菜单栏中“工程”按钮选中“新建工程”，填写工程名称、保存路径。点击“确定”后在工程存放目录下生成文件名为“M54HCT160 _ TEST”的工程文件。

2）工程参数设置：

进入工程参数设置界面后，可以对芯片测试工程的参数进行输入设置了，工程参数设置界面主要包括：管脚信号定义、管脚映射配置、电源模块配置、电平模块配置，直流参数模块配置。

（1）管脚信号定义，首先定义待测芯片的管脚名称和管脚类型，然后定义信号组，方便接下来的测试。

（2）管脚映射配置，在定义好信号管脚信息后，配置管脚映射模块，将信号与仪器通道建立起连接。

（3）电源模块配置接下来配置开短路测试中所需要的电源参数。

（4）电平模块配置接着配置管脚的驱动电平与比较电平等参数。

（5）直流参数模块配置，需要说明的是开短路测试的判断条件，如果引脚接触正常，那么测试结果应该为二极管的导通电压 0.7 V 左右，设置判断条件为 0.2～1.5 V 之间。

（6）测试项配置，将已经配置好的电源、电平模块配置到“Open/Short”测试项中。

3）测试结果：

测试结果如图 8 所示，管脚“MR/”、“D［2］”、“D［3］”、“Q［0］”、“Q［2］”没有通过测试，结果为“FAIL”，

（下转第 168 页）