

南通大學



本科毕业设计

题目	面向量子计算的辅助逻辑设计及仿真平台设计与实现
----	-------------------------

作者：郁可人

专业：计算机科学与技术

指导教师：管致锦

完成日期：2016 年 5 月 25 日

原创性声明

本人声明：所呈交的论文是本人在导师指导下进行的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已发表或撰写过的研究成果。参与同一工作的其他同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

签 名：_____日 期：_____

本论文使用授权说明

本人完全了解南通大学有关保留、使用学位论文的规定，即：学校有权保留论文及送交论文复印件，允许论文被查阅和借阅；学校可以公布论文的全部内容。

（保密的论文在解密后应遵守此规定）

学生签名：_____指导教师签名：_____日期：_____

南通大学毕业设计立题卡

课题名称	面向量子计算的辅助逻辑设计及仿真平台设计与实现	出题人	管致锦
课题表述 (简述课题的背景、目的、意义、主要内容、完成课题的条件、成果形式等)	<p>课题背景：量子计算仿真平台是量子计算机电路、量子计算以及新的量子算法的开发研究的一个有用工具。借助量子电路模型模拟量子计算，实现量子信息演算可视化是一个复杂、包含多学科理论技术的计算机应用系统。与量子计算机对应的量子计算有别于经典计算机的经典计算。经典计算机，其输入态和输出态都是经典信号，用量子力学的语言来描述，也即是，其输入态和输出态都是某一力学量的本征态。其内部的每一步变换都将正交态演化为正交态，而一般的量子变换没有这个性质，因此，经典计算机中的变换（或计算）只对应一类特殊集。相应于经典计算机的以上两个限制，量子计算机分别作了推广。量子计算机的输入态和输出态为一般的叠加态，其相互之间通常不正交；量子计算机中的变换为所有可能的么正变换。得出输出态之后，量子计算机对输出态进行一定的测量，给出计算结果。由此可见，量子计算对经典计算作了极大的扩充，经典计算是一类特殊的量子计算。量子计算最本质的特征为量子叠加性和相干性。量子计算机对每一个叠加分量实现的变换相当于一种经典计算，所有这些经典计算同时完成，并按一定的概率振幅叠加起来，给出量子计算机的输出结果，这种计算称为量子并行计算。量子并行处理大大提高了量子计算机的效率，使得其可以完成经典计算机无法完成的工作。</p> <p>目的：通过本课题的研究，设计软件模拟量子电路进行量子计算，建立完整的量子计算仿真平台通用架构。</p> <p>意义：量子计算仿真平台作为研究量子电路设计和开发新的量子算法的有力工具，可以为相关的人员测试设计和算法提供很大的帮助。</p> <p>主要内容：</p> <ol style="list-style-type: none"> 1. 设计量子计算平台通用架构； 2. 软件实现量子计算平台计算内核； 3. 软件实现对辅助功能和输入输出接口。 <p>完成课题的条件:Java 虚拟机、GNU 或 Windows 环境</p> <p>成果形式：提交量子计算仿真平台系统一份、毕业设计论文一份。</p>		
课题来源	科研	课题类别	毕业设计
该课题对学生的要求	<ol style="list-style-type: none"> 1. 熟悉相关专业知识； 2. 具备有相关软件开发的理论知识； 3. 具备相关文献和学习资料的学习能力； 4. 有较强的钻研精神、实践能力和较好的写作水平。 		
系、部意见	<div style="text-align: right;">系、部主任签名：_____</div> <div style="text-align: right;">_____年_____月_____日</div>		
学院意见	<div style="text-align: right;">同意立题（ ）</div> <div style="text-align: right;">不同意立题（ ）</div> <div style="text-align: right;">教学院长签名：_____</div> <div style="text-align: right;">_____年_____月_____日</div>		

南 通 大 学

毕业设计任务书

题目 面向量子计算的辅助逻辑设计及仿真平台设计与实现

学 生 姓 名 郁可人

学 院 计算机科学与技术学院

专 业 计算机科学与技术

班 级 计 123 班

学 号 1213022082

起 讫 日 期 2016 年 1 月 15 日~2016 年 6 月 12 日

指导教师 管致锦 职称 教授

发任务书日期 2016 年 1 月 15 日

课题的内容和要求（研究内容、研究目标和解决的关键问题）
<p>研究内容：</p> <p>量子计算仿真平台是量子计算机电路、量子计算以及新的量子算法的开发研究的一个有用工具。</p> <p>量子电路计算效率比集成电路高很多，其进行计算时具有量子并行性，设计成量子计算机可以大大提高计算的效率，可以完成许多经典计算机无法完成的工作。但实际上量子电路物理实现还处于雏形阶段，更多的是根据量子计算的逻辑进行先行研究。</p> <p>在量子计算研究过程中不可避免地会遇到建模复杂，模型更新麻烦等一系列问题。量子计算仿真平台就是为了解决这些问题而设计的。</p> <p>研究目标：</p> <p>通过本课题的研究，建立完整的量子计算仿真平台通用架构，设计软件模拟量子电路进行量子计算。</p> <p>解决的关键问题：</p> <ol style="list-style-type: none">1. 模拟量子计算的过程；2. 设计较优量子门的存储结构；3. 设计通用接口。
课题的研究方法和技术路线
<ol style="list-style-type: none">1. 查阅文献和阅读专业书籍；2. 审视已完成的量子计算仿真平台 ver0.0；3. 参考比对 University Bremen 的团队设计的可逆电路仿真模拟平台 revkit 以及其他可能存在的相关仿真平台；4. 设计量子计算仿真平台 ver1；5. 撰写论文；6. 整理与总结；7. 终期结题答辩。
基础条件
<p>本课题的指导教师长期从事科学研究工作。对课题的相关研究内容比较深刻的了解，对相关内容具备一定的指导经验。选择该设计课题的学生具备了相关研究的专业基础，曾以项目主持人身份参与设计了量子计算仿真平台 ver0.0，并进行了专利申请和软件著作权申请。</p>
参考文献

- [1]管致锦, 可逆逻辑综合[M], 北京, 科学出版社, 2011.2.
- [2]Ioannis G. Karafyllidis, “Quantum Computer Simulator Based on the Circuit”, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, VOL. 52, NO. 8, AUGUST, 2006
- [3]Steffen, M (Steffen, M.), “Quantum computing: An IBM perspective”, IBM JOURNAL OF RESEARCH AND DEVELOPMENT, VOL. 5, NO. 13, 2011
- [4]Amlan Chakrabarti, Susmita Sur-Kolay, Senior MemberAmlan Chakrabarti, Susmita Sur-Kolay, Senior Member , Ayan Chaudhury, “Linear Nearest Neighbor Synthesis of Reversible Circuits by Graph Partitioning”, arXiv:1112.0564.
- [5]Electr. & Comput. Eng. Dept., Portland State Univ., Portland, OR, USA, “Linear Reversible Circuit Synthesis in the Linear Nearest-Neighbor Model”, IEEE Computer society (2012): 157 – 160.
- [6]N.Alhagi, M.Lukac, L.Tran, M.Perkowski. Two-Stage Approach to the Minimization of Quantum Circuits Based on ESOP Minimization and Addition of a Single Ancilla Qubit[C], Proc. ULSI 2012.
- [7]<http://webhome.cs.uvic.ca/~dmaslov/definitions.html>.
- [8]PoKerntopf, M. Perkowski, K. Podlaski, Synthesis of Reversible Circuits: A View on the State-of-the-Art[C], 2012 12th IEEE International Conference on Nanotechnology. 2012, 260–269.
- [9]M. Saeedi, R. Wille, R. Drechsler. Synthesis of quantum circuits for linear nearest neighbor architectures[J] Quantum Information Process, 2011, 10(3): 355–377.
- [10]A. G. Fowler, S. J. Devitt, and L. C. L. Hollenberg. Implementation of Shor’ s algorithm on a linear nearest neighbor qubit array. Quantum Information and Computation, 2004, 4(4), pp. 237–251.
- [11]Y. Takahashi, N. Kunihiro, and K. Ohta. The quantum fourier transform on a linear nearest neighbor architecture. Quantum Information and Computation, 2007, 7(4), pp. 383–391.
- [12]M. Saeedi, R. Wille, and R. Drechsler. Synthesis of quantum circuits for linear nearest neighbor architectures. Quantum Information Processing, 2011, 10(3), pp. 355–377.

本课题必须完成的任务

1. 模拟量子计算的过程;
2. 设计较优量子门的存储结构;
3. 设计通用接口。

成果形式

提交量子计算仿真平台系统一份、毕业设计论文一份。		
进度计划		
起讫日期	工作内容	备注
2016.01.15-2016.03.15	查阅相关资料文献，了解国内外相关课题的研究方向和进展，确定项目的可行性方案。	
2016.03.16-2016.03.23	初步完成开题报告，准备开题答辩。	
2016.03.24-2016.03.31	学习 rekit 设计方案，比对量子计算仿真平台 ver0.0 设计方案。	
2016.04.1-2016.05.10	量子计算仿真平台 ver1 设计。	
2016.05.11-2016.05.31	撰写论文，上交检查，准备答辩。	
2016.06.01-2016.06.03	论文答辩，验收。	
系、部审核意见	<div>系、部主任签名：_____ 年__月__日</div>	
学院意见	<div>教学院长签名：_____ 年__月__日</div>	

南通大学本科毕业设计开题报告

学生姓名	郁可人	学 号	1213022082	专业	计算机科学与技术
课题名称	面向量子计算的辅助逻辑设计及仿真平台设计与实现				
阅读文献情况	国内文献 1 篇	开题日期	2016 年 03 月 23 日		
	国外文献 9 篇	开题地点	南通大学新校区		
<p>一、文献综述与调研报告：（阐述课题研究的现状及发展趋势，本课题研究的意义和价值、参考文献）</p> <p>研究的现状及发展趋势：</p> <p>量子计算仿真平台是量子计算机电路、量子计算以及新的量子算法的开发研究的一个有用工具。借助量子电路模型模拟量子计算，实现量子信息演算可视化是一个复杂、包含多学科理论技术的计算机应用系统。与量子计算机对应的量子计算有别于经典计算机的经典计算。经典计算机，其输入态和输出态都是经典信号，用量子力学的语言来描述，也即是，其输入态和输出态都是某一力学量的本征态。其内部的每一步变换都将正交态演化为正交态，而一般的量子变换没有这个性质，因此，经典计算机中的变换（或计算）只对应一类特殊集。相应于经典计算机的以上两个限制，量子计算机分别作了推广。量子计算机的输入态和输出态为一般的叠加态，其相互之间通常不正交；量子计算机中的变换为所有可能的么正变换。得出输出态之后，量子计算机对输出态进行一定的测量，给出计算结果。由此可见，量子计算对经典计算作了极大的扩充，经典计算是一类特殊的量子计算。量子计算最本质的特征为量子叠加性和相干性。量子计算机对每一个叠加分量实现的变换相当于一种经典计算，所有这些经典计算同时完成，并按一定的概率振幅叠加起来，给出量子计算机的输出结果，这种计算称为量子并行计算。量子并行处理大大提高了量子计算机的效率，使得其可以完成经典计算机无法完成的工作。</p> <p>实际上量子电路物理实现还处于雏形阶段，更多的是根据量子计算的逻辑进行先行研究。研究人员在进行量子逻辑研究时通常需要自行设计基础数据结构，不可避免地会遇到建模复杂，模型更新麻烦等一系列问题。量子计算仿真平台就是为了解决这些问题而设计的。</p> <p>目前国际上最主要的有 University Bremen 的团队设计的可逆电路仿真模拟平台 revkit，以及该团队制定的 benchmark revlib。本人在之前参与设计了量子电路仿真平台 ver0.0，作为测试技术细节和架构可行性的软件测试版。</p> <p>研究的意义和价值：</p> <p>通过本课题的研究，建立完整的改进的量子计算仿真平台通用架构，设计软件模拟量子电路进行量子计算——量子电路仿真平台 ver1。</p> <p>参考文献：</p> <p>[1]管致锦，可逆逻辑综合[M]，北京，科学出版社，2011.2.</p> <p>[2]Ioannis G. Karafyllidis, “Quantum Computer Simulator Based on the Circuit”, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, VOL. 52, NO. 8, AUGUST, 2006</p> <p>[3]Steffen, M (Steffen, M.), “Quantum computing: An IBM perspective”, IBM JOURNAL OF RESEARCH AND DEVELOPMENT, VOL. 5, NO. 13, 2011</p> <p>[4]Amlan Chakrabarti, Susmita Sur-Kolay, Senior MemberAmlan Chakrabarti, Susmita Sur-Kolay, Senior Member , Ayan Chaudhury, “Linear Nearest Neighbor Synthesis of Reversible Circuits by Graph Partitioning” , arXiv:1112.0564.</p>					

- [5] Electr. & Comput. Eng. Dept., Portland State Univ., Portland, OR, USA, “Linear Reversible Circuit Synthesis in the Linear Nearest-Neighbor Model”, IEEE Computer society (2012): 157 – 160.
- [6] N. Alhagi, M. Lukac, L. Tran, M. Perkowski. Two-Stage Approach to the Minimization of Quantum Circuits Based on ESOP Minimization and Addition of a Single Ancilla Qubit[C], Proc. ULSI 2012.
- [7] <http://webhome.cs.uvic.ca/~dmaslov/definitions.html>.
- [8] P. Kerntopf, M. Perkowski, K. Podlaski, Synthesis of Reversible Circuits: A View on the State-of-the-Art[C], 2012 12th IEEE International Conference on Nanotechnology. 2012, 260-269.
- [9] M. Saeedi, R. Wille, R. Drechsler. Synthesis of quantum circuits for linear nearest neighbor architectures[J] Quantum Information Process, 2011, 10(3): 355-377.
- [10] A. G. Fowler, S. J. Devitt, and L. C. L. Hollenberg. Implementation of Shor’ s algorithm on a linear nearest neighbor qubit array. Quantum Information and Computation, 2004, 4(4), pp. 237-251.
- [11] Y. Takahashi, N. Kunihiro, and K. Ohta. The quantum fourier transform on a linear nearest neighbor architecture. Quantum Information and Computation, 2007, 7(4), pp. 383-391.
- [12] M. Saeedi, R. Wille, and R. Drechsler. Synthesis of quantum circuits for linear nearest neighbor architectures. Quantum Information Processing, 2011, 10(3), pp. 355-377.

二、本课题的基本内容，预计解决的难题

主要内容：

1. 建立并改进量子电路仿真平台架构
2. 依据架构设计可投入使用的量子电路仿真平台新版本

预计解决的难题：

1. 模拟量子计算的过程；
2. 设计较优量子门的存储结构；
3. 设计合适的通用接口。

三、课题的研究方法、技术路线

1. 查阅文献和阅读专业书籍；
2. 审视已完成的量子计算仿真平台 ver0.0；
3. 参考比对 University Bremen 的团队设计的可逆电路仿真模拟平台 revkit 以及其他可能存在的相关仿真平台；
4. 设计量子计算仿真平台 ver1；
5. 撰写论文；
6. 整理与总结；
7. 准备终期结题答辩。

四、研究工作条件和基础					
本课题的指导教师长期从事科学研究工作。对课题的相关研究内容有比较深刻的了解，对相关内容具备一定的指导经验。选择该设计课题的学生具备了相关研究的专业基础，曾以项目主持人身份参与设计了量子计算仿真平台 ver0.0，并进行了专利申请和软件著作权申请。					
五、进度计划					
起讫日期		工作内容			
2016.01.15-2016.03.15		查阅相关资料文献，了解国内外相关课题的研究方向和进展，确定项目的可行性方案。			
2016.03.16-2016.03.23		初步完成开题报告，准备开题答辩。			
2016.03.24-2016.03.31		学习 rekit 设计方案，比对量子计算仿真平台 ver0.0 设计方案。			
2016.04.1-2016.05.10		设计改进新的平台架构，软件制作。			
2016.05.11-2016.05.31		撰写论文，上交检查，准备答辩。			
2016.06.01-2016.06.03		论文答辩，验收。			
论文阶段完成日期		文献调研完成日期	2016.03.15	论文实验完成日期	2016.05.15
		撰写论文完成日期	2016.05.31	评议答辩完成日期	2016.06.01
指导教师评语	<div style="text-align: right;">导师签名：_____ 年__月__日</div>				
系部意见	<div style="text-align: right;">系、部主任签名：_____ 年__月__日</div>				
学院意见	通过开题（ ） 开题不通过（ ） <div style="text-align: right;">教学院长签名：_____ 年__月__日</div>				

南 通 大 学

毕 业 设 计

题目：面向量子计算的辅助逻辑设计及仿真平台
设计与实现

姓 名：郁可人
指导教师：管致锦
专 业：计算机科学与技术

南通大学计算机科学与技术学院

2016 年 5 月 16 日

摘 要

量子计算机是近年来的热门话题，然而量子计算的物理实现仍处在瓶颈阶段，量子电路的逻辑设计也存在许多基本问题。为了在资源有限的情况下进行量子计算的逻辑研究，量子计算仿真平台是不可或缺的解决手段。目前国际上已有 RevKit 量子电路仿真平台，而国内尚未有相关仿真平台。

面向量子计算的辅助逻辑设计及仿真平台，是用软件方法实现的用于辅助量子逻辑研究的工具。该平台定义了一种通用架构，包括一套基于 C++ 的量子计算类库和一系列基于类库设计的量子电路演示组件。

本课题中，对量子电路中需要作为研究依据的部分参数格式化存储，其核心是存储量子电路中的门序列。同时还保存了用于解释这些量子门的量子逻辑规则，根据“基本门存储计算规则，非基本门存储构造方法”的设计思想，基本门规则保存为量子真值表，非基本门规则保存为量子电路序列。这些规则按门库分类组合。量子电路类库还提供了较完整的可以对量子电路进行的基本操作，可以满足绝大多数量子电路算法的编写需要。在导入类库后，用户可以自己设计并演示量子电路算法。

本课题设计的量子电路仿真平台相比 RevKit，量子门信息表达更加简单，可以对电路进行的操作同 RevKit 一样全面。测试表明，基于本仿真平台可以构造包括电路综合、优化、检测在内的各种量子电路算法。

关键词：量子电路，仿真平台，量子门库

ABSTRACT

Quantum computer is a hot topic in recent years, however, the physical realization of quantum computation is still in the bottle neck, and there are many basic problems in the logic design of the quantum circuit. In order to studying the logic of quantum computation with limited resources, the simulation platform of quantum computation is an indispensable solution. At present, there is RevKit quantum circuit simulation platform in the world, but there is no relevant simulation platform in China.

The Quantum Computation Oriented-Aided Design and the Simulation Platform, is a software toolkit that assists quantum logic research. A general framework is designed in the platform, which includes a Quantum Computation Class Library based on C++ and some quantum circuit presentation components based on the class library.

In this topic, the author picked general parameters of quantum circuits, formatted and saved them, especially the gate list. Also, the author designed a method to interpret the logic rules implied in the gate. Based on the theory that ‘Basic quantum gates save the computation rules, and non-basic gates save the construction rules’, rules of basic quantum gates saved as real table, while rules of non-basic gates as component gates list. Gate rules are classified by gate library. Quantum computation class library supplies complete fundamental operations act on quantum circuit as well. It is able to make up most quantum circuits algorithms. After importing the library, users can design their own circuit or import quantum circuit from file, and then select any appropriate gate library to explain the circuit.

The quantum circuit simulation platform is designed to extract the core functions of the class library, such as the gate decomposition and the gate calculation. It is a quite convenient tool.

Compared with RevKit, simulation of quantum gate information in this project is more simple, and the operation of the circuit can be as comprehensive as RevKit. Test shows that circuit algorithms including circuit synthesis, optimization, detection algorithms can be constructed with this simulation platform.

Key Words: quantum circuit, simulation platform, quantum gate library

目 录

摘 要.....	I
ABSTRACT.....	II
目 录.....	III
第 1 章 绪 论.....	1
1.1. 理论基础.....	1
1.1.1 量子计算理论基础.....	1
1.1.2 部分量子门.....	2
1.1.3 门库运算的封闭性.....	4
1.2. 量子计算研究历史简述.....	6
1.3. 量子计算仿真意义.....	6
1.4. 国内外研究现状.....	7
1.5. 论文组织结构.....	8
第 2 章 量子计算仿真平台架构.....	9
2.1. 信息文件格式定义.....	9
2.1.1 RevLib 量子电路信息文件.....	10
2.1.2 QCS 量子门库信息文件.....	11
2.1.3 QCS 量子门规则信息文件.....	13
2.2. QCS 架构.....	18
2.2.1 功能.....	18
2.2.2 设计思想.....	18
2.2.3 逻辑设计.....	20
第 3 章 QCS 类库设计.....	21
3.1. gate 模块.....	21
3.1.1 class Gate.....	21
3.1.2 class SimpleGate.....	22
3.1.3 class ComplexGate.....	24
3.1.4 class NullGate.....	26

3. 2. gatelib 模块.....	26
3. 2. 1 class GateLib.....	26
3. 3. circuit 模块.....	29
3. 3. 1 struct GateInCircuit.....	29
3. 3. 2 struct TruthInCircuit.....	29
3. 3. 3 class Circuit.....	29
3. 4. application 模块.....	33
3. 4. 1 class QuAlgorithm.....	34
3. 4. 2 说明.....	34
第 4 章 QCSUI 功能分析与测试.....	35
4. 1. QCSUI.....	35
4. 1. 1 功能分析与测试.....	35
4. 1. 2 利用 QCSUI 测试 QCS 运算正确性.....	40
4. 2. QCSShell.....	43
4. 2. 1 QCSShell 指令集.....	43
4. 2. 2 指令功能分析与测试.....	44
4. 3. 粒子概率分布灰度仿真图.....	47
第 5 章 总 结.....	50
附 录 一 量子电路相关文件示例.....	51
附 录 二 QCSUI 操作说明书.....	53
附 录 三 与本文相关的成果.....	59
参考文献.....	60
致 谢.....	61

第 1 章 绪 论

1.1. 理论基础

1.1.1 量子计算理论基础

量子计算是一种新型计算方式，它以量子力学为理论基础。在量子力学中，一个物理体系的状态由波函数表示，波函数的任意线性叠加仍然代表该体系的一种可能状态，这可以描述微观粒子的运动。将这个性质应用到计算逻辑中，利用量子状态作为存储和计算的工具，量子的叠加性和相干性作为超高效计算的技术基础。

在传统逻辑中，一个比特位（bit）的信息被确定地归结为 0 或 1，在经典计算机中表现为电气信号。一个 n 比特位的经典存储单元一次只能存储 0 到 2^n-1 中某个确定的真值，如果计算完整的真值表则必要进行 2^n 次经典运算。而在量子逻辑中，作为基本算子的是量子位（qubit），量子位的叠加态不是确定性的而是概率性的。一个量子位是一个双态量子系统，同一时刻单个量子位可以处于两个标准正交态的任意线性叠加。也可以看做以双态为基矢张成的二维 Hilbert 空间。

量子位通常用 Dirac 记号 $|\psi\rangle$ 标记，双态通常表示为：

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

用这种标记方法可以表示一个任意量子位，表示如下：

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad |\alpha|^2 + |\beta|^2 = 1 \quad (1.1)$$

其中， α, β 均为复数，代表计量该量子状态时，该量子状态会以 $|\alpha|^2$ 的概率坍缩到 $|0\rangle$ 态，以 $|\beta|^2$ 的概率坍缩到 $|1\rangle$ 态。这样的量子状态通常用布洛赫球面（Bloch sphere）来进行几何表示，见图 1-1。

单量子位的性质可以进行推广。一个 n 位的量子寄存器可以同时保存和处理 2^n 个 n 位正交态，记作 $|\chi_1\chi_2\cdots\chi_n\rangle$ 。矩阵表示为 n 个量子位的态的张量积(tensor product)。也可以看做以双态为基矢张成的 n 维 Hilbert 空间。

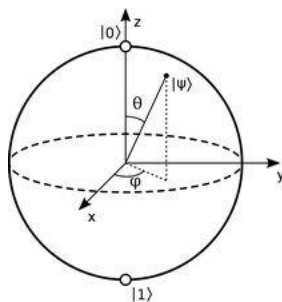


图 1-1 布洛赫球面 (Bloch sphere)

举例 2 位量子寄存器的态向量表示如下：

$$\begin{aligned}
 |00\rangle &= |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} & |01\rangle &= |0\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \\
 |10\rangle &= |1\rangle \otimes |0\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} & |11\rangle &= |1\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}
 \end{aligned}$$

量子计算的物理特性使其能够一次性对量子态中的所有叠加分量完成变换，并将结果按一定的概率振幅叠加起来，再根据实际需要进行测量结果，这体现了量子计算的内在并行性。

1.1.2 部分量子门

一个量子门是一个基本的，操作一个小数量量子位的量子变换。它是构成量子电路的基础部件。量子门是可逆的，它通常使用矩阵表示，它的特点是：输入和输出的量子位数量必须相等。量子门的操作可以用代表量子门的矩阵与代表量子位状态的向量作相乘来表示。

1) H 门 (Hadamard gate, 阿达马门)

仅对一个量子位操作，使 $|0\rangle$ 态变为 $\frac{|0\rangle + |1\rangle}{\sqrt{2}}$ ，使 $|1\rangle$ 态变为 $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$ 。H 变换是么正变换。变换

矩阵表示如下：

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

2) 相位偏移门 (Phase shift gates)

指的是一系列对单个量子位进行操作的门，保留 $|0\rangle$ 态，使 $|1\rangle$ 态变为 $e^{i\theta}|1\rangle$ 。变换矩阵表示如下：

$$R(\theta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$$

3) NOT 门 (量子非门)

与经典逻辑中的非门作用类似，对单个量子位实现量子态反转。变换矩阵表示如下：

$$N = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

4) CNOT 门 (C-NOT 门, cnot 门, 受控非门)

改进自经典逻辑中的异或门，即将原本舍弃的垃圾位保留。cnot 门控制两个量子位，有一个控制位和一个目标位。当控制位为 0 态时时，目标位的值保持不变；当控制位为 1 态时，目标位的值进行翻转。变换矩阵表示如下：

$$U = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

5) V 门和 V+门

V 和 V+门通常指具有一个控制位和目标位的受控 V 门和受控 V+门，只有当控制位的输入为 1 态时，目标位才会发生变换；V 门与 V+门的目标位功能矩阵是共轭的关系，相乘后的结果是一个单位矩阵。

V 门变换矩阵表示如下：

$$V = \frac{1+i}{2} \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}$$

V+门变换矩阵表示如下：

$$V_+ = \frac{1-i}{2} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}$$

6) SWAP 门（交换门）

SWAP 门是非基本量子门，对两个量子位进行操作，表现为对两个位线上的输入矢量进行互换。SWAP 门常用于实现量子电路线性最近邻化。SWAP 门由三个基本量子门 cnot 门级联成。

7) Toffoli 门（toffoli 门，多控制位控制非门）

对多个量子位进行操作，具有多个控制位和一个目标位，只有当所有的控制位的输入都是 1 态时，目标位才会发生变换，变换操作是取反操作。通常所说的狭义 Toffoli 门特指双控制位 Toffoli 门。理论上仅由 Toffoli 门就可以完成任意经典电路的设计。Toffoli 门也不是基本量子门，对 Toffoli 门的量子构造方法研究曾经是研究的热门，目前流行的 Toffoli 构造方法是由 5 个基本门构成。

Toffoli 门变换矩阵表示如下：

$$U = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

1.1.3 门库运算的封闭性

关于量子逻辑门库的严格定义如下：在量子计算中，如果一组量子基本逻辑门对量子计算是通用的，则这组逻辑门就被称为是量子逻辑门库。换言之，量子逻辑门库是可以组成能够任意逼近任何酉运算的量子线路的，参见参考文献[5]。

另一个概念关于量子纠缠，如果多个量子位组成的态矩阵可以表示为各个量子位的态的张量积，则称这个状态为直积态；而如果多量子位的态矩阵无法直接拆分成张量积，则称这些量子位处于纠缠态。处于纠缠态中的量子位需要至少两个量子态进行叠加才能表示。如：

$$|\beta_{00}\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \quad |\beta_{01}\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}} \quad |\beta_{10}\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}} \quad |\beta_{11}\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}$$

经典二值逻辑运算中的真值只有‘0’或‘1’，但是量子纠缠可能使量子位出现纠缠态的情况，理论上态的集合是个不可数集合。如果门库中门的数量是有限的，门库中的所有门都是

基于同一个态集合的置换操作，那么这个态的集合是有限的。这是量子门库运算的封闭性。

证明如下：假设一个量子门库中门数量有限。如果有一个量子门的置换会产生新的纠缠态却没有对应的酉运算，则不满足量子门库对通用性的要求。如果门库中态的集合无限，则产生对应的态的门也是无限的，不满足门库中门有限的前提。综上得证。

根据态的有限性，可以将纠缠态也认为是广义真值的一种，这种真值不能出现在电路的输入和输出位置，但可以在电路运算中出现。

以 NCV 门库举例。标准 NCV 门库由 not 门、cnot 门、v 门、v+门组成。所有可能出现的单量子位真值态包括：

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$|\alpha\rangle = \begin{pmatrix} \frac{1-i}{2} \\ \frac{1+i}{2} \end{pmatrix}$$

$$|\beta\rangle = \begin{pmatrix} \frac{1+i}{2} \\ \frac{1-i}{2} \end{pmatrix}$$

NCV 门库的运算封闭性示意图如图 1-2。

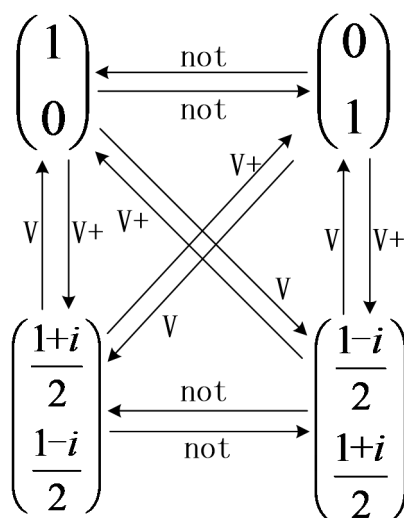


图 1-2 NCV 门库运算封闭性示意图

由此可见，定义完整的门库其运算结果集合应该是封闭的，而该门库中的非基本门都应由基本门构成，并且可以循环自举以构建拥有更复杂逻辑的非基本门。

1.2. 量子计算研究历史简述

Turing 在 1936 年发明可编程计算机的思想，他认为一台装有足够资源的计算机能够实现任何合理的算法。然而 Feynman 在 1982 年提出，Turing 机无法描述 n 个粒子之间相互作用的量子力学关系，即 NP-Hard 问题，而其实 Turing 机也无法解决 NPC 问题。能够模拟 n 粒子关系的只有 n 粒子本身。这引出了一种新型的计算逻辑：经典计算逻辑通常设计一个函数 $f(x)$ ，并对参数 x 进行定义，依据 x 的不同定义得到不同的输出结果，经典计算适合处理低维数的“简单”问题；而量子计算特性使其一次性存储参数 x 的所有定义，并填满所有输出结果，需要进行的只有测量预期的输出结果，适合解决高维数大规模的问题。

1961 年，Landauer 提出计算过程的能量消耗与计算的可逆性有着必然的联系，他指出能耗的主要产生于计算过程中的不可逆操作，信息在清除时会产生热量。为了避免这种逻辑上的不可逆性，Landauer 认为可以保留计算后无用的信息位，比如将异或门改造成有两个输出位的“控制非门”。在此基础上，20 世纪 70 年代，Fredkin, Toffoli, Bennett 等人发现可逆操作与量子计算具有紧密联系，他们希望利用量子实现可逆计算机以达到降低功耗的作用。1982 年 Benioff 和 Feynman 证明可以用量子力学系统来描述可逆计算机，即量子计算机的可行性。1985 年 Deutsch 进一步提出了通用量子计算机的构想，并提出了第一个量子算法。1993 年 Bernstein, Vazirani 和 Yao 研究了量子计算的复杂性，证明了量子计算比经典计算更强大。

1994 年 Shor 提出了大数质因分解的量子算法。作为量子计算研究的标志性事件，Shor 算法第一次将量子计算用于解决实际问题，并引申出了量子解密、加密等领域^[8]。之后，1996 年，Grover 提出了量子搜索算法。此后量子计算机逐渐开始成为主要研究目的，而可逆操作研究作为量子电路设计的一个中介。先将量子门线性组合成有实际逻辑意义的可逆门，再将可逆门级联成量子电路，成为研究量子计算的核心思路。

1.3. 量子计算仿真意义

目前实现量子计算机最主要的障碍是无法制作多量子位寄存器，设计出的量子计算机不仅价格高，稳定性差，更令人遗憾的是与经典计算机相比计算性能优势不明显。量子计算的物理实现仍处于初步阶段，仿真是解决理论研究和物理研究进展差距过大的最好方法。

量子计算仿真平台基于量子电路设计，是量子计算机电路、量子计算以及新的量子算法的

开发研究的一个有用工具。其中量子计算的仿真体现在量子位经过量子门进行运算，量子算法的仿真则体现在通过设计量子电路实现算法功能并验证。

在量子电路仿真方法中，需要向模拟量子寄存器的接口输入量子位的初始状态，可以得到的输出包括依次经过每一个量子门计算后得到的量子位状态。量子电路仿真建立在经典计算机平台上，因此所有的量子计算都由经典计算逻辑来仿真进行，复杂度规模是量子计算规模的 2 的指数次方倍。根据 Feynman 证明的经典图灵机的 NP 不可解特点，量子仿真也不可能完全模拟粒子状态。

虽然不能仿真量子计算的规模性，仿真方法仍具有可以计算并模拟难以测量量子状态的优点。经过量子门变换后量子状态可能呈纠缠态而无法准确测量，仿真方法则不存在这个问题。通常用矩阵计算仿真量子计算，计算结果矩阵即使无法分解也可以很好地保留并继续计算。通过对每一步得到的仿真计算结果分析，量子计算仿真平台还可以反推出实际应该呈现的量子状态，这种状态可以通过“粒子概率分布灰度仿真图”“量子计算阶段变换仿真图”^[4]等模拟演示。

对特定功能量子电路的综合方法也是仿真平台测试的功能之一，常用的综合方法仍然或多或少地借用枚举，复杂度都不低。在高复杂度情况下，高效的数据结构及其尽可能减少不必要步骤的基础操作是十分必要的。相关研究人员往往会自己设计简化的实验验证程序，这种程序也是广义上的仿真平台。然而单一功能的仿真平台难以满足多元的实验要求，同时也缺少统一的度量模型。一个统一的、功能全面的、度量模型丰富且可替换度高的量子计算仿真平台对量子逻辑研究十分重要。

1.4. 国内外研究现状

2005 年，Ioannis G. Karafyllidis 及其团队通过其仿真平台举例了三种仿真平台的应用，包括解释 Deutsch 的量子电路并行性证明算法，量子傅里叶（Fourier）变换和量子纠缠态的表现。演示方法包括“粒子概率分布灰度仿真图”和“量子计算阶段变换仿真图”^[4]。

University Bremen 的量子研究团队在 2008 年推出了 RevLib，是一个存储量子电路的平台库，并提供了一套完整的存储规格。此后这套存储格式逐渐成为业内标准，即 RevLib Benchmark。此外还提供了一个 RevKit 组件，该组件是一个完整意义上的量子电路仿真平台，可以对电路进行构造，更改，逻辑计算等等。RevKit 基于 C++ 编写核心，基于 Python 编写交

互，可以编写自定义的量子算法^{[2][3]}。

IBM 公司于今年 5 月 4 日（2016 年 5 月 4 日）推出了量子计算云平台。该平台的后台是一个真正的 5 量子位处理器。虽然该平台目前仅能用于教学与测试量子计算稳定性，但前景不言而喻。

1.5. 论文组织结构

面向量子计算的辅助逻辑设计及仿真平台，是用软件方法实现的用于辅助量子逻辑研究的工具。其实质是一种辅助量子电路设计研究的通用编程架构。

第一章主要介绍了量子计算的相关概念、仿真平台的设计意义和研究历史。本文对架构的正式介绍从第二章开始，共三大章 9 小节。

第二章介绍了面向量子计算的辅助逻辑设计，通俗地说即仿真平台架构的逻辑设计。量子电路相关信息文件作为仿真平台中使用的数据结构的介绍，也归于本章中。

第三章介绍了架构的量子电路编程类库部分，包括类库的详细设计和使用解释。类库是架构物理实现的核心部分，对类库的介绍按照逻辑结构进行，分别介绍每个逻辑层中的具体物理实现。

第四章通介绍架构的应用与演示部分。架构下的仿真演示插件基于架构设计并且递归属于仿真平台架构。

第 2 章 量子计算仿真平台架构

本课题中设计的量子电路仿真平台最大的特点在于电路的分层构造：将量子门按照基本门和非基本门的分类存储，基本门负责存储运算方法，非基本门负责构造电路，非基本门由基本门级联构成。

本课题进行的所有开发统称为“量子计算仿真平台 ver.1”(Quantum Computation Simulation ver.1, QCSv1)，核心是开发了“量子计算仿真架构”(QCS 架构)，可以辅助设计量子算法以及进行量子计算仿真。平台为 Windows 操作系统环境设计，主要包括一个 C++ 的量子电路设计类库 (QCS 类库) 和一系列平台插件。

逻辑上，QCS 架构包括 gate 层，circuit 层，application 层。

物理上，QCS 类库为核心构件，包括量子门库模块，属于 gate 层；量子电路模块，属于 circuit 层；扩展功能模块，属于 application 层；QCS 文件格式满足 gate 层和 circuit 层的存储需求；QCSUI 基于类库设计，是将功能模块的部分功能集成的一个演示平台，属于逻辑 application 层。

在 QCS 架构中，相同度量模型下的量子门集合以门库为存储结构，量子门库与门库接口程序共属于量子门库模块。以量子电路的构造为核心功能的模块属于量子电路模块。扩展功能模块用于存储高级的可以自定义的量子电路算法，或者是进行各种仿真演示。

架构详细规定了文件格式，量子电路的保存采用 RevLib 的量子电路 (Circuits) 格式标准，量子电路信息文件后缀为 “.real”。有别于 RevLib 的操作思想，本课题对门分类，并依据分类不同存储不同的门信息，因此量子门信息的格式独立设计，文件后缀为 “.gaterule”。门按组织存储在门库目录文件中，门库中有负责统计量子门的文件，后缀为 “.gatelib”。真值格式没有硬性标准，当前架构没有采用 RevLib 格式，也没有专门设计详细格式，后缀为 “.tr”，仅用于存储真值。

本章主要介绍包括文件逻辑格式在内的 QCS 架构的逻辑设计。

2.1. 信息文件格式定义

QCS 架构内部通信基于文件通信，规格化的文件是 QCS 不可或缺的组成部分，文件格式由 QCS 架构规定。如表 2-1 所示，QCS 中需要用到的文件包括量子电路信息文件 (.real)，量

子门库及门库表信息文件（.gatelib），量子门规则信息文件（.gaterule），称之为 QCS 文件。在符合文件规则的条件下，用户可以自定义文件内容。

表 2-1 QCS 相关文件及格式对照表

信息文件	文件格式
量子电路信息文件	.real
量子门库表信息文件	.gatelib
量子门规则信息文件	.gaterule

2.1.1 RevLib 量子电路信息文件

采用 RevLib 的量子电路（Circuits）格式标准。该文件有两个版本，版本一（version 1.0）发布于 2008 年；版本二（version 2.0）发布于 2010 年，支持 RevLib 定义的可逆电路硬件描述语言（SyReC）和 PLA 格式真值信息文件，见参考文献[3]。一个电路定义在一个 ASCII 编码文件中，后缀为“.real”，文件包括两个部分，量子电路报头和量子电路准确描述。文件中字段分标识字段和信息字段，如“.version”是标识字段，而“2.0”就是信息字段，信息字段是标识字段标识的对象。标识字段与信息字段、信息字段与信息字段之间用空格字符分隔；两条完整的信息（标识字段或标识字段加信息字段）之间用换行符分隔。该信息文件实例见附录一。

1) 电路信息文件的报头信息

量子电路信息文件的报头信息解释见表 2-2。

2) 电路信息文件的内容信息

在报头后定义的是对量子电路组成的描述，从“.begin”字段开始，到“.end”字段结束。门描述按电路的计算逻辑有序，若两个门之间没有逻辑先后关系则可以顺序不计。在两个字段的每一行描述代表一个门实例信息。每条描述中的第一个字段表示门的名称，为字符串；之后的字段标识了该门实例的所跨越位线，每一个字段都是字符串，对应所跨越位线在“.variables”字段对象中的名称（而非“inputs”字段和“outputs”字段标识的别称）。

单条门实例信息的语法如下：

```
<gate-instance>::=<gate-name>{' '<gate-position>}
<gate-name>::=<string>
<gate-position>::=<string>
<string>::=<char>{'<char>}
<char>::=('A'|...|'Z'|'a'|...|'z'|'0'|...|'9'|'_')
```

单一的一条门实例信息中没有任何体现门逻辑信息，甚至没有包括门的控制位和目标位位数，该门所跨越位数也是隐含表达。这使得门信息极为精简。实际上门信息存储的 RevLib 的硬件描述文件格式（HDL 及 SyReC）中，本课题没有采用这个格式，并在独立设计的量子门文件格式中同样加入了这些字段。

表 2-2 量子电路信息文件报头信息对照表

标识字段	逻辑名称	对象字段信息
#	Comments	提供设计者信息，电路描述等可读信息，该字段不处理。
.version	Version	版本号，RevLib 会不定期更新，更新内容根据版本号标识，该字段不处理。
.numvars	Number of Variables	变量数，非负整型，指.variables 字段对象数量，同时也是.inputs 和.outputs 字段的对象数量。
.variables	Variables	变量，字符串序列，标识电路所用到的所有逻辑量子位。
.inputs	Inputs	输入变量，字符串序列，.variables 中可以获得输入的量子位，可以用别称，主要用于真值表字段名。
.outputs	Outputs	输出变量，字符串序列，.variables 中可以读取输出的量子位，可以用别称，主要用于真值表字段名。
.constants	Constant Inputs	常量输入，字符串，字符串中的每一位代表.inputs 字段中相应位序的变量是否为常量。可选字符 ‘0’ 代表常量 0，‘1’ 代表常量 1，‘-’ 代表该位不是常量位。
.garbage	Garbage Outputs	垃圾输出，字符串，字符串中的每一位代表.outputs 字段中相应位序的变量是否为垃圾位，即输出无实意。可选字符 ‘1’ 代表该位是垃圾位，‘-’ 代表该位不是垃圾位。

2.1.2 QCS 量子门库信息文件

量子门库格式独立设计，其实例是包含了本门库中所有门描述的目录文件，以及一个在目录文件中的对门库中所有门进行统计的门库表信息文件。门库表信息文件 ASCII 编码，文件后缀为 “.gatelib”，简称门库表。一个门库文件中应该有且仅有一张门库表。门库表也有简单的报头，其中包含门库名称信息。标识字段与信息字段之间用冒号分隔；信息字段与信息字段之间用空格字符分隔；两条完整的信息（标识字段或标识字段加信息字段）之间用换行符分隔。

1) 量子门库表信息文件的报头信息

量子门库表信息文件的报头信息解释见表 2-3。其中门库名称不能唯一标识门库是指：可能存在两个逻辑不同的门库文件，但它们具有相同的门库名称。

表 2-3 量子门库表信息文件报头信息对照表

标识字段	逻辑名称	对象字段信息
gateLib- Name	门库名称	字符串，虽然不具有有唯一标识门库作用，但具有一定匹配作用。
gateNum	门数量	门库中门规则的数量，非负整型，等于表具体门信息中的数量。

2) 量子门库表信息文件的内容信息

在报头后定义的是对门规则类型的描述。从标识字段“libTable”和分隔号冒号开始，共有“gateNum”所标识数量的规则描述，规则描述不分先后。每条描述共有两个字段，第一个字段代表门规则的名称（即门名称）；第二个字段代表门规则的类型。QCS 平台中，可选的门类型共有三种，‘0’代表该门为基本门，‘1’代表该门为非基本门的简单门，‘2’代表该门为非基本门的复杂递归门。具体的门规则介绍见 2.2.3 节量子门规则信息文件介绍。

门库表中单条门规则信息的语法如下：

```
<gate-rule>::=<gate-name>' '<gate-type>
<gate-name>::=<string>
<gate-type>::=('0'|'1'|'2')
<string>::=<char>{<char>}
<char>::=('A'|...|'Z'|'a'|...|'z'|'0'|...|'9'|'_')
```

3) 门库信息文件结构解释与补充

如上所述，一个量子门库的实例是包含了本门库中所有门规则信息文件以及门库表的目录文件。门规则信息文件描述门的计算逻辑规则且一个门规则信息文件仅能描述一个门。门库目录文件也可以封包成门库信息文件。

量子门库中，每一个门名称都是唯一的，也是同门库中门的唯一标识。如果出现设计的门库中有同名的门，按照 Windows 文件查找规则，仅会读取同名的第一个文件，即使这两个门文件规则不同甚至分类不同。只有在不同的量子门库中门的名称可以相同。

量子门库表中，门库名（“gateLibName”）字段是对门库的代表与称呼，但不是对门库的唯一标识，这会在下一段进行解释。门库表中没有专门设置对门文件位置的索引，门文件存储在与门库表相同目录下，索引按照门名称进行。门名称与门库中门文件的文件名一一对应，

并且按照门库表进行索引而非对门库目录文件遍历索引。门库表中的门名称同样是对门的唯一标识，QCS 类库加载门库时会根据门库表内容在内存中生成结构表（`std::map` 而非 `std::multimap`），若有多条同名信息则会以最后一条为准，若有门文件未在门库表中注册则不会被读取。

量子门库不止一个，甚至可以这样定义，有不同度量模型的门库具有不同的量子门库信息文件。如上一段所述，“gateLibName” 字段不是对门库的唯一标识。本版本 QCS 中设计，用户在加载门库时根据文件路径加载，并在加载的同时为该门库取一个别称，这个别称将作为运算时门库的唯一标识，并在卸载门库时收回。

甚至会出现这样的情况，在一个目录文件中存放了多个门库，体现在一个目录文件中有多张门库表，并根据门库表所注册的在相同目录下存放完整的门文件，甚至这个目录文件中有很多完全无关紧要的文件！这种做法是可行的但是强烈不推荐的！

QCS 架构要求，一个清晰的逻辑应该是为一个门库建立一个门库目录文件；目录文件名称是门库名；目录文件中存放一个门库表信息文件（.gatelib），表中“gateLibName” 字段与目录文件名相同，“gateNum” 字段与门库表中门规则数量相同；门库目录文件中的门文件与“libTable” 字段后的门信息一一对应，没有多余或重复的文件。

2.1.3 QCS 量子门规则信息文件

量子门规则信息文件格式同样为独立设计，是体现 QCS 计算思想的核心，与 RevKit Tool 有着本质区别。门规则定义在 ASCII 编码文件中，使用“.gaterule” 后缀。文件包括量子门规则报头和量子门规则描述两个部分。标识字段与信息字段之间用冒号分隔；信息字段与信息字段之间用空格字符分隔；两条完整的信息（标识字段或标识字段加信息字段）之间用换行符分隔。

1) 量子门类型的设计思想

通常的设计思想中，cnot 门、V 门、V+门等等狭义的量子门负责量子计算实现，而 toffoli 门等狭义的可逆门负责逻辑计算的实现。自顶向下解释，若要对某电路进行真值运算，则需要先将电路中的非基本门全部分解成基本门，再对完全基本由基本门组成的同逻辑的电路进行运算。QCS 的设计遵循这种思想。向更广义的思想推广，由一部分门，称之为基本门负责底层运算，而非基本门由基本门线性构成，基本门可以不是狭义的基本量子门。在操作电路时以非基

本门为主，偶尔插入基本门来进行。

将这种设计具象到存储结构设计，基本门存储真值变换等演算逻辑，非基本门存储构成规则，每一条规则的描述都基于基本门。跟完整存储真值表相比，这种设计极大地节约了空间。跟 HDL 等标准相比，这种设计减少了规则编写难度，省去了编译时间，而且读取非常简单快捷。

这种设计实质上是将存储复杂度转换为运算时间复杂度，增加了电路转换操作，在计算电路真值时需要先将电路转换为全部由基本门组成的电路后才能够进行计算。

门类型字段为‘0’代表该门为基本门，门类型为‘1’或‘2’代表该门为非基本门。其中由于某些非基本门的构成规则是递归构成的，比如 toffoli 门系列的多控制位 toffoli 门，一种分解方法是将 toffoli_n 门分解为 2 个 toffoli_(n-1)门和一个 v 门一个 v+门的组合，并依次递归分解直到完全分解为基本门，如图 2-1，摘自参考文献[7]。

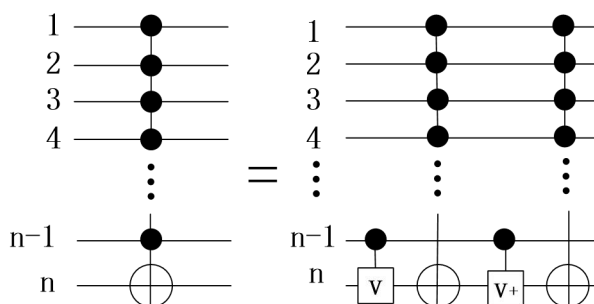


图 2-1 一种 Toffoli 门的递归分解方法示意图

为了便于分解和最大利用存储信息，QCS 将非基本门类型再分类为不需要递归的简单非基本门（`gateType = 1`），和需要进行递归分解的复杂非基本门（`gateType = 2`）。若电路转换操作时读取到门类型为‘2’，则会对该门进行递归分解，其余情况不递归。

详细定义为：简单非基本门（`gateType = 1`）为由且仅由基本门（`gateType = 0`）构成的门。递归非基本门（`gateType = 2`）为至少包含一个递归非基本门（`gateType = 2`）的门，只要符合逻辑，这个递归门可以是其自身也可以是其他递归门。

2) 量子门规则信息文件的报头信息

量子门规则信息文件的报头信息解释见表 2-4。

关于跨越位数不定的非基本门，目前认为目标位与控制位必须至少有一个的位数固定的。事实上这也是符合逻辑的，如果一个门的控制位和目标位位数都不确定，那么它的逻辑规则就完全无从谈起。而只要有其中一个确定，那么就可以推算出另一个的位数。QCS 根据这个规则

设计。

如果有用户自定义的复杂门甚至无法区分控制位和目标位，那么推荐将这些位都归为目标位。

在报头后定义的是对量子门规则的描述，根据逻辑需要，对基本门与非基本门规则的描述必然是不同的。

表 2-4 量子门规则信息文件报头信息对照表

标识字段	逻辑名称	对象字段信息
gateName	门名称	字符串，具有唯一标识门的作用，该字段与文件名相同。
gateLib- Name	门库名称	字符串，虽然不具有有唯一标识门库作用，但具有一定查错作用。
gateType	门类型	可选字符 ‘0’ 代表基本门； 可选字符 ‘1’ 代表不递归非基本门； 可选字符 ‘2’ 代表递归非基本门。
relate- Gate	递归关联门	仅供 gateType = 2 的递归非基本门使用，字符串，唯一标识该递归门的递归出口，递归出口可以是任何 “gateType” != ‘2’ 的门，以该门的门名称标识。
control- Num	控制位数量	存储控制位数量的信息， 若 gateType = 0，为非负整型，代表控制位数量； 若 gateType = 1，为非负整型或 ‘-1’，其中非负整型代表控制位数量，‘-1’ 代表该门控制位数量不确定； 若 gateType = 2，为整型，其中非负整型代表控制位数量，负整型代表该门将递归分解到控制位数量等于 “controlNum” 信息字段的绝对值为止。
target- Num	目标位数量	存储控制位数量的信息， 若 gateType = 0，为正整型，代表目标位数量； 若 gateType = 1，为正整型或 ‘-1’，其中正整型时代表目标位数量，‘-1’ 时代表该门目标位数量不确定； 若 gateType = 2，为非零整型，其中正整型代表目标位数量，负整型代表该门将递归分解到目标位数量等于本信息字段的绝对值为止。
quantum- Cost	量子代价	仅供 gateType = 0 的基本门使用，整型，标识该基本门的量子代价。

3) 基本量子门规则信息文件的内容信息

基本门规则。从标识字段 “truthTable” 和分隔号冒号开始，之后每一行代表一个真值变换

情况。根据 1.1.3 节介绍的门库的封闭性可知，有限门库的量子门运算中出现的叠加态是有限的。因此 QCS 推荐为每个独立的叠加态设置一个别称，这些叠加态也属于广义的真值。每条描述中第一个字段为输入真值序列；第二个字段是输出真值序列，中间用“->”字符串隔开。

基本门规则中的单条真值变换规则的语法如下：

```
<truth-struct>::=<truth-control>('?:')<truth-target>('->')<truth-control>('?:')<truth-target>
<truth-control>::=(<truth>?){<truth>}
<truth-target>::=(<truth>?){<truth>}
<truth>::=<char>
<char>::=('A'|...|'Z'|'a'|...|'z'|'0'|...|'9'|'_')
```

4) 非基本量子门规则信息文件的内容信息

非基本门规则格式不区分简单非基本门和递归非基本门。从标识字段“ruleTable”和分隔号冒号开始，之后每一行代表一条构成该门的子规则信息。每条描述中第一个字段为字符，指出该条子规则的模板；第二个字段是门名称，是门的唯一标识；第三个字段为字符串序列，指出子规则说明的分解门的分解方法，其中控制位和目标位以冒号分隔。

非基本门规则中的单条子规则的语法如下：

```
<rule-struct>::=<rule-name>'<gate-name>'<rule-control>('?:')<rule-target>
<rule-control>::=(<rule-position>?){'<rule-position>'}
<rule-target>::=(<rule-position>?){'<rule-position>'}
<rule-name>::='N'|'E'|'X'|'Y'
<gate-name>::=<string>
<rule-position>::=('c'|'t')('b'|'e')<int>
<string>::=<char>{<char>}
<char>::=('A'|...|'Z'|'a'|...|'z'|'0'|...|'9'|'_')
<int>::=('0'|...|'9'){('0'|...|'9')}
```

子规则指示在分解时应在该位置根据规则模板（“rule-name”）插入的门或门系列。本系统 QCS 中设计的规则模板有 3 个，即“rule-name”标识的字段有 3 个可选字符‘N’、‘E’、‘X’。子规则的顺序为逻辑有序，与量子电路信息文件中门实例内容顺序类似。

子规则的第三个字段分解方法字段（“rule-control”或“rule-target”）中的每一个子字段都分别表示添加的分解门的位置规则（“rule-position”）。每个字段又由三部分组成：第一部分可选字符‘c’或‘t’，分别指示被分解的非基本门的控制位（control）和目标位（target）；第二部分可选字符‘b’或‘e’，分别指示字段第一部分所指的控制位或目标位的起始（begin）或结束（end）位置，同时指示若为‘b’则第三部分位数为正向，若为‘e’则第三部分位数为负向；第三部分为正整型，指示从前两部分共同定位的非基本门线位开始，按第二部分所指

的方向读取的位数，该位数从 0 开始计。

见图 2-2，以该图为例，需要被分解的是一个三控制位 toffoli 门（t4），量子位线共有 a、b、c、d 四个，其中 a、b、c 是控制位，d 是目标位。控制位中，从 a 到 c 方向称之为控制位正向（“cb”），从 c 到 a 方向称之为控制位负向（“ce”）。a 是控制位正向的起点，即“cb0”，c 是控制位正向的终点，即“cb2”，同时 c 也是控制位负向的起点，即“ce0”。可见一条位线可能有多个指示名称，这不会影响 QCS 实际运作，同时希望用户在编写自定义的复杂门规则时能找到最适合自己的指示方法。同理，可见目标位的唯一位线 d 既是“tb0”也是“te0”。除此之外，任何指示了该段所属控制位或目标位位线以外的情况都将在实际运行时被 QCS 提示报错，比如上例中不能指示不存在的“cb3”或“tb1”等等，这种错误称为“添加越界”错误。

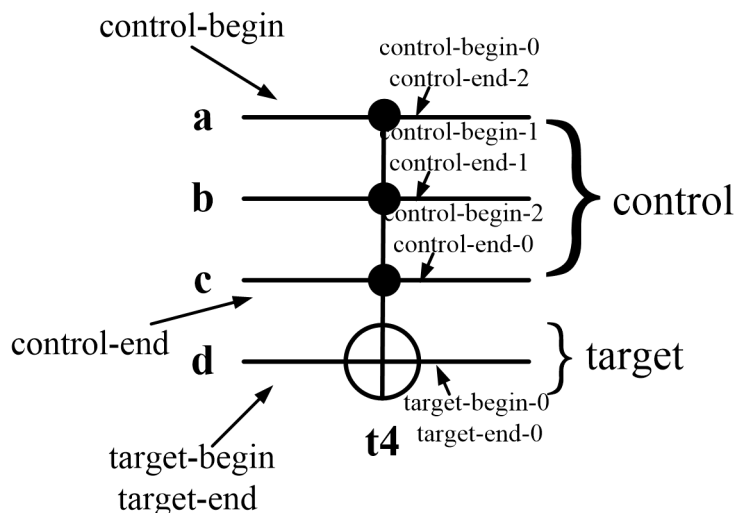


图 2-2 非基本门信息文件子规则第三字段实例图

有了可以指示被分解门位线的基础，再解释子规则中第一字段“rule-name”的几个可选模板。

模板‘N’（no rule），表示在该位添加一个门，门名称为子规则第二字段（“gate-name”），添加的门的位线见第三字段，第三字段分隔符冒号之前指示的位线是添加门的控制位，冒号之后指示的位线是添加门的模板位。位线按顺序依次添加。

模板‘E’（except），该模板主要用于递归门分解，表示在该位添加一个门，门的名称仍是第二字段（“gate-name”），而门的位线为除了指示位线之外的被分解门的所有位线。如图 2-1，摘自参考文献[7]。该门分解规则见附录一。

模板‘X’，该模板主要用于非递归且不固定位线的非基本门设计，添加多个门，门的名

称仍是第二字段（“gate-name”），添加的第一个门的位线为第三字段指示的位线，之后每个添加的门的位线均为前一个门的位线按方向进一位，前进的方向依据一二字段的指示，直到继续添加会出现添加越界情况为止。如图 2-3，摘自参考文献[7]。其门分解规则见附录一。

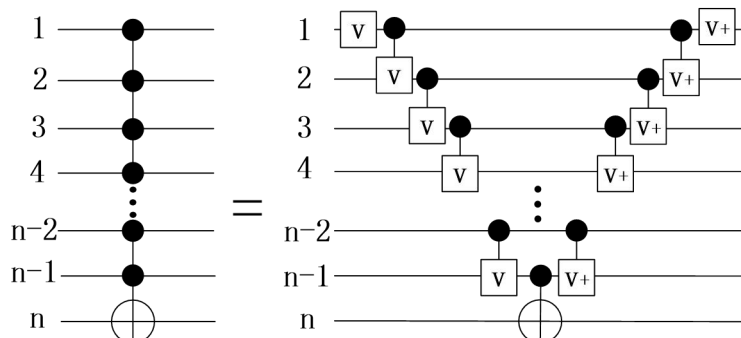


图 2-3 一种 Toffoli 门的非递归分解方法示意图

2.2. QCS 架构

QCS 架构是在 C++ 编译环境下，导入并使用 QCS 类库定义的方法，加载 QCS 定义的量子电路相关文件，利用包括 QCSUI 在内可以读取相关文件的演示插件，进行程序设计的一种轻量级框架。简单来说，QCS 架构的物理组成包括 QCS 类库、QCS 文件、QCS 演示插件等。

2.2.1 功能

QCS 架构的实现功能包括：

- 1) 构造量子电路；
- 2) 计算量子电路；
- 3) 编写量子电路算法；
- 4) 利用所提供参数比对算法；
- 5) 量子电路演示。

2.2.2 设计思想

QCS 逻辑设计从存储、计算、控制三个设计方面考虑。最终的逻辑结构中，存储与计算设计主要演变为 gate 层，控制设计演变为 circuit 层以及功能层 application 层。

存储设计包括对量子电路的存储、电路运算结果的存储，这两类存储是存储设计的核心，

甚至根据“真值表相同的电路逻辑相同”原则，仅有电路运算结果就可唯一标识电路。这里有意混淆了量子电路与量子门（主要指非基本门）的关系，两者归根到底都是由门库的基本门构成的，两者最大的逻辑区别在于非基本量子门有清晰的控制位和目标位区分，而量子电路没有。但也可以将量子电路认为是全部位都是目标位的非基本门。也因此，之前版本 QCSv0.0 设计的非基本量子门与量子电路格式相同甚至可以相互转换，这种设计有一定实用性但完全无法推广。权衡之下，在 QCSv1 版本设计中的量子电路信息文件部分选用了广为接受的 RevLib 电路信息文件版本。电路运算（真值）的存储结构取决于电路运算方法的设计，计算采用矩阵运算和模式匹配运算的结果分别是矩阵和字符串。

计算设计特指对量子门的计算，不考虑整个量子电路运算流程的控制。核心是计算方法的选择。量子电路的运算方式通常用矩阵表示，这符合量子运算本质的 Hilbert 空间逻辑。矩阵运算存储的信息完整，但张量矩阵的分解是一个数学问题，且矩阵运算量极大，最终的物理设计中还是选择了模式匹配运算，理论基础是 1.1.3 节介绍的门库的封闭性。不过如同逻辑设计所提到的，逻辑设计中不需要确定运算的方式，但需要在给出运算方式的同时给出规格化方法。换句话说，运算结果存储的文件的格式唯一，即使是矩阵运算也需要将向量结果分解为各个量子位向量的张量积存储。

控制设计包括对量子电路运算的控制和对量子电路构造的操作控制。核心思想便是“将量子电路分解成完全由基本门组成的量子电路，再对完全由基本门组成的同逻辑的电路进行运算。”对量子电路的操作是设计量子电路算法不可缺少的部分，量子电路算法中有关量子电路构造的操作最终都可以分解为以下操作的逻辑组合：添加一位量子位线、删除一位量子位线、添加一个门、删除一个门。这些操作也被归纳为控制操作。控制设计还为存储和计算设计提供上层调用。

算法设计、相关插件、或基于平台设计的附加功能都归于 application 层。

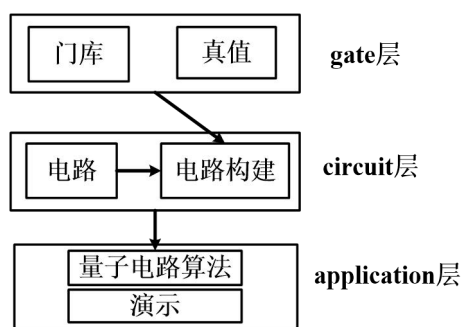


图 2-4 QCS 架构逻辑设计示意图

2.2.3 逻辑设计

逻辑结构分三层：门层（gate）、电路（circuit）层和应用（application）层。如图 2-4 是 QCS 架构的逻辑设计示意图。

1) gate 层

该层中定义了有关运算的所有基本信息，以及对应的规格化方法：

- a. 量子门库的存储结构；
- b. 量子门库的规格化方法；
- c. 量子门的信息存储方法；
- d. 基本门计算信息的规格化方法；
- e. 非基本门规则信息的规格化方法；
- f. 真值的存储方法；
- g. 真值的规格化与逆规格化方法与纠缠态判断。

2) circuit 层

该层是 QCS 架构的核心，向下设计对 gate 层的规格化后的信息读写接口，向上提供所有规格化的操作。该层包括量子电路的存储、量子电路的构造方法、对量子电路运算、统计方法：

- a. 量子电路的存储结构；
- b. 量子电路的规格化方法；
- c. 量子电路的构造方法；
- d. 接受 gate 层提供门库的操作方法；
- e. 接受 gate 层提供真值的操作方法；
- f. 量子电路的真值计算方法；
- g. 量子电路的信息统计方法。

3) application 层

该层用于存储所有基于 circuit 层设计的高级方法或实用插件。包括：

- a. 量子电路算法的存储结构；
- b. 接受 circuit 层整合的所有操作方法；
- c. 量子电路算法的存储；
- d. 量子电路演示方法。

第 3 章 QCS 类库设计

QCS 类库，是以 QCS 架构的逻辑设计为根据进行的具体物理设计。类库可供用户直接使用，因此本章将提供类库设计的详细介绍，包括类的大多数定义细节（数据定义、方法定义）。类库基于 C++11 设计，主要采用标准 stl（Standard Template Library）库，少量使用 Windows 提供的系统操作，暂未使用 boost 库。

本章按模块分节，按类介绍，对每个类的介绍会按照类的数据结构定义、关联结构或内部类、特殊方法、符号重载、get/set 方法的顺序进行。类的数据结构定义与 2.2.节相关信息文件格式介绍中的文件信息结构大致对应，除了有特殊规格化需要的数据结构，其余不再详细介绍。对关联结构或内部类的介绍会着重介绍该结构设计的原因。类的特殊方法指除了 get/set 方法和符号重载方法之外的定义方法，将逐条详细介绍。最后，符号重载、get/set 方法会作为两个模块列举，由于其功能简单易懂，不进行详细说明。

3.1. gate 模块

门物理结构，包括虚基类 Gate 类，和三个 Gate 派生类 SimpleGate 类、ComplexGate 类、NullGate 类。这些内容定义于文件“Gate.h”中。

3.1.1 class Gate

Gate 类是虚基类，是 gate 模块中所有类的基类。用于读取、缓存、提供门规则信息接口。数据结构与量子门规则信息文件（.gaterule）中的信息结构大致对应，参见 2.1.3 节。

1) 数据结构定义

```
string gateName;//门名称，唯一标识
string gateLibName;//门库名称
int gateType;//门类型
int contNum//控制位数量
int targNum;//目标位数量
bool useEnable;//是否可使用
```

“useEnable”字段标识标识了该门是否可使用，可选字段‘1’代表可使用，‘0’代表不可使用，未定义完全的门和无逻辑门（class NullGate）都是不可使用的门。另外，基类中存在的数据结构在派生类中不再提及。

2) read 方法

```
virtual bool read(const string&) = 0;
```

功能：从指定路径读取门规则信息文件内容。

参数：参数一为文件路径。

返回值：返回值为读取成功（1）或失败（0）。

基类不解释具体的读取方法，读取方法在派生类解释。

3) get/set 方法

```
virtual string get_gateName(void) = 0;
virtual string get_gateLibName(void) = 0;
virtual int get_gateType(void) = 0;
virtual bool get_useEnable(void) = 0;
```

基类不要求 set 方法存在。派生类根据具体情况设置 set 类。

3.1.2 class SimpleGate

作为 Gate 类的派生类，代表基本门（gateType = 0）。数据结构与基本门规则信息文件（.gaterule）中的内容结构大致对应，核心是缓存真值表（truthTable）。该类是逻辑结构 gate 层中设计的“基本门计算信息规格化方法”的物理实现。

1) 数据结构定义

```
map<string, string> truthTable;//真值表
```

2) 内部类 iterator

仿照 stl 的 iterator 结构，QCS 类库中的大多数类结构都有各自的迭代器（iterator），用于访问该类的核心存储内容，迭代器的实质是 stl 提供的数据结构迭代器的封装。本迭代器中封装的是 map<string, string>::iterator，用于迭代读写 truthTable 内容。

```
class iterator{
private:
    map<string, string>::iterator truthTableIter;//map::iterator
private:
    friend class SimpleGate;
public:
    SimpleGate::iterator& operator = (const SimpleGate::iterator&);
    SimpleGate::iterator& operator ++ ();
    const SimpleGate::iterator operator ++ (int);
    bool operator == (const SimpleGate::iterator&);
```

```
bool operator != (const SimpleGate::iterator&);
string input(void);
string output(void);
};
```

其中 `input` 方法和 `output` 方法中，

```
string input(void);
```

返回 `truthTable` 中一条键值对的键字段，实质为 `map::iterator.first`。

```
string output(void);
```

返回 `truthTable` 中一条键值对的值字段，实质为 `map::iterator.second`。

3) read 方法

```
bool read(const string&);
```

功能：从指定路径读取门规则信息文件内容。读取成功后 `useEnable` 置 1。

参数：参数一为文件路径。

返回值：返回值为读取成功（1）或失败（0）。

4) find 方法

```
string find(const string&);
```

功能：根据输入的模式串从 `truthTable` 中寻找匹配的输出生。即，使用模式匹配方式的逻辑真值运算。

参数：参数一为输入模式串。

返回值：返回值为输出模式串。如果无法再真值表中寻找到匹配规则，则返回输入串。

5) begin 方法

```
SimpleGate::iterator begin(void);
```

返回值：返回一个指向 `truthTable` 中的第一条信息的迭代器。

6) end 方法

```
SimpleGate::iterator end(void);
```

返回值：返回一个指向 `truthTable` 中最后一条信息之后位置的迭代器。类似 `stl` 中的 `end` 方法设计。

7) 关于遍历

```
SimpleGate x;
SimpleGate::iterator i;
for(i = x.begin(); i != x.end(); ++i) loop();
```

8) 符号重载

```
bool operator == (const SimpleGate&);
```

9) get/set 方法

```
int get_gateType(void);
bool get_useEnable(void);
string get_gateName(void);
string get_gateLibName(void);
int get_contNum(void);
int get_targNum(void);
int get_quantumCost(void);
```

3.1.3 class ComplexGate

作为 Gate 类的派生类，代表非基本门（gateType = 1|2）。数据结构与非基本门规则信息文件（.gaterule）中的内容结构大致对应，核心是缓存分解规则表（ruleTable）。该类是逻辑结构 gate 层中设计的“非基本门规则信息规格化方法”的物理实现。

1) 数据结构定义

```
string relateGate;//关联门
vector<RuleStruct> ruleTable;//规则表
```

2) 内部类 iterator

迭代器，封装的是 vector<RuleStruct>::iterator，用于迭代读写 ruleTable 内容。

```
class iterator{
private:
    vector<RuleStruct>::iterator ruleTableIter;//vector::iterator
private:
    friend class ComplexGate;
public:
    ComplexGate::iterator& operator = (const ComplexGate::iterator&);
    ComplexGate::iterator& operator ++ ();
    const ComplexGate::iterator operator ++ (int);
    bool operator == (const ComplexGate::iterator&);
    bool operator != (const ComplexGate::iterator&);
    const string get_ruleName(void);
    const string get_gateName(void);
    const vector<string> get_contPosi(void);
    const vector<string> get_targPosi(void);
};
```

3) 相关结构体 RuleStruct

该结构是组成 ComplexGate::ruleTable 的元结构。结构类似非基本门规则信息文件中的每

条子规则信息的组织结构。有四个字段：规则模板名（ruleName），对应文件中子规则的“rule-name”；门名称（gateName），对应子规则的“gate-name”；控制位信息（contPosi），对应子规则的“rule-control”；目标位信息（targPosi），对应子规则的“rule-target”。另外，ComplexGate 类中只存储子规则的规则信息，对规则模板的解释则属于 Circuit 类，逻辑上归为计算设计而非存储设计。

```
struct RuleStruct{
    string ruleName;
    string gateName;
    vector<string> contPosi;
    vector<string> targPosi;
};
```

4) read 方法

```
bool read(const string&);
```

功能：从指定路径读取门规则信息文件内容。读取成功后 useEnable 置 1。

参数：参数一为文件路径。

返回值：返回值为读取成功（1）或失败（0）。

5) begin 方法

```
ComplexGate::iterator begin(void);
```

返回值：返回一个指向 ruleTable 中的第一条信息的迭代器。

6) end 方法

```
ComplexGate::iterator end(void);
```

返回值：返回一个指向 ruleTable 中最后一条信息之后位置的迭代器。

7) 符号重载

```
bool operator == (const ComplexGate&);
```

8) get/set 方法

```
int get_gateType(void);
bool get_useEnable(void);
string get_gateName(void);
string get_gateLibName(void);
string get_relateGate(void);
int get_contNum(void);
int get_targNum(void);
```


3.1.4 class NullGate

无逻辑门，作为 Gate 类的派生类，无实意。该类的设计为了应对多态情况下可能发生的类型问题。比如，若门库表中存有某个门文件的信息，但实际路径下不存在该门，则门库查找函数（GateLib::find()）会返回一个 NullGate。

```
class NullGate : public Gate{
public:
    bool read(const string&);//返回 1
    bool read(void);//返回 1
    string get_gateName(void);//返回 "-"
    string get_gateLibName(void);//返回 "-"
    int get_gateType(void);//返回 -1
    bool get_useEnable(void);返回 0
};
```

NullGate 类中，gateType 规定为 -1，useEnable 规定为 0 即不可使用。其余字段以及函数仅作实现与重载用，无实意。

3.2. gatelib 模块

门库物理结构，基于 gate 模块，仅包括一个类 GateLib。该结构定义于文件“GateLib.h”。

3.2.1 class GateLib

该类是逻辑结构 gate 层中设计的“量子门库规格化方法”的物理实现。核心是缓存门库中的所有门规则信息的门库信息表（gateLibTable），其数据结构与门库表信息文件（.gatelib）中的结构大致对应，参见 2.1.2 节。该类不直接显式存储门库表的内容，而是依据门库表依次读取并缓存具体的门规则。

1) 数据结构定义

```
string gateLibName;//门库名称，标识
int gateNum;//门库中门数量
bool useEnable;//是否可使用
map<string, Gate*> gateLibTable;//门库表
```

2) 内部类 iterator

```
class iterator{
private:
    map<string, Gate*>::iterator gateLibTableIter;
private:
```

```

        friend class GateLib;
    public:
        GateLib::iterator& operator = (const GateLib::iterator&);
        GateLib::iterator& operator ++ ();
        const GateLib::iterator operator ++ (int);
        bool operator == (const GateLib::iterator&);
        bool operator != (const GateLib::iterator&);
        string get_gateName(void);
        Gate* get_gate(void);
};

```

迭代器，封装内容为 `map<string, Gate*>::iterator`，用于迭代读写 `gateLibTable` 内容。

其中 `get` 方法返回当前迭代器指向的门规则的指针。

```
Gate* get_gate(void);
```

该函数返回的是指针而非引用指针。

3) read 方法

```
bool read(const string&);
```

功能：如图 3-1，从指定目录路径下，寻找并读取门库表信息文件，再依据门库表中每个门的类型信息新建相应类型缓存，并调用相应的 `Gate::read` 方法从相同目录路径中按名读取量子门规则信息文件。门名称、量子门规则将分别作为键、值成对存入量子门库表。读取成功后 `useEnable` 置为 1。

参数：参数一为目录文件路径。

返回值：返回值为读取成功（1）或失败（0）。

4) find 方法

```
Gate* find(const string&);
```

功能：根据输入的门名称从 `gateLibTable` 中寻找匹配的量子门规则。

参数：参数一为量子门名称。

返回值：返回值为指向该门规则的指针。如果当前门库中没有该门，则返回一个无逻辑门（`NullGate`）。可以在调用此方法时通过测试返回的门的门类型（`gateType`）来判断，对返回指针指向的对象调用 `get_gateType` 方法，返回 -1 则表示寻找失败。

5) begin 方法

```
GateLib::iterator begin(void);
```

返回值：返回一个指向 `gateLibTable` 中的第一条信息的迭代器。

6) end 方法

```
GateLib::iterator end(void);
```

返回值：返回一个指向 gateLibTable 中最后一条信息之后位置的迭代器。

7) 关于遍历

```
GateLib x;  
GateLib::iterator i;  
for(i = x.begin(); i != x.end(); ++i) loop();
```

8) unload 方法

```
void unload();
```

功能：主动释放门库内存，清除所有门规则，卸载此门库。useEnable 置为 0。

9) get/set 方法

```
string get_gateLibName(void);  
int get_gateNum(void);  
bool get_useEnable(void);
```

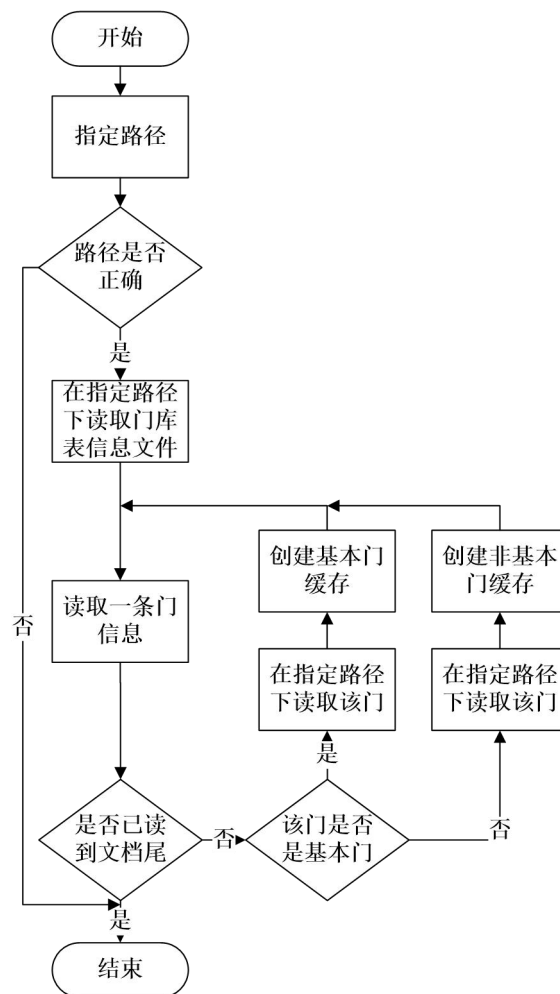


图 3-1 read 方法运行流程图

3.3. circuit 模块

电路物理结构，基于 `gatelib` 模块，仅包括一个类 `Circuit`。另有两个结构体：电路中的门结构（`GateInCircuit`）、电路运算的真值结构（`TruthInCircuit`）。这两个结构体实际上是 `Circuit` 类的关联结构，但由于这两个结构有独立的实际意义，且经常在基于类库编程时直接使用，因此独立介绍。这些内容定义于文件“`Circuit.h`”中。

3.3.1 struct GateInCircuit

该结构是缓存电路实例中的门实例的结构，也是组成 `Circuit::gateList` 的元结构。组织结构类似量子电路信息文件中每个门实例信息的结构。有两个字段：门名称（`gateName`），对应文件中的“`gate-name`”；位信息（`posi`），对应文件中的“`gate-position`”。位信息不显式区分控制位和目标位，但按照目标位在前控制位在后的顺序排列。该结构同样属于 `gate` 层设计的“量子门信息规格化方法”的物理实现。

```
struct GateInCircuit{
    string gateName;//实例门名称
    vector<string> posi;//实例门跨越位线
};
```

3.3.2 struct TruthInCircuit

该结构是缓存真值的结构，是计算过程中缓存真值表的完整结构。若存储于文件中，后缀为“`.tr`”，文件无规定格式。该结构是 `gate` 层设计的“真值的规格化与逆规格化方法”的物理实现。

```
struct TruthInCircuit{
    map<string, string> truthTable;//真值表
};
```

核心为真值表 `truthTable`，键为输入真值，值为模式匹配运算得到的输出真值。

该结构与基本门（`SimpleGate`）类结构的核心存储结构 `SimpleGate::truthTable` 相同。这为电路与门的转换留下了余地。

3.3.3 class Circuit

该类是逻辑结构 `circuit` 层中设计的“量子电路的规格化方法”的物理实现，同时也包括“量

子电路的构造方法”、“接受 gate 层提供门库的操作方法”、“接受 gate 层提供真值的操作方法”、“量子电路的真值计算方法”、“量子电路的信息统计方法”的物理实现。Circuit 的数据结构与量子电路信息文件 (.real) 中的结构大致对应, 参见 2.1.1 节, 核心是缓存电路中的门实例信息表 gateList。Circuit 作为 QCS 类库的核心类, 既是对其下层功能的总结, 也是可以向上层提供的唯一接口集合。

1) 数据结构定义

```
string circuitName;//量子电路名称
string gateLibName;//门库名称
int gateNum;//门数量
int quantumCost;//电路量子代价
int variNum;//变量数
vector<string> variables;//变量
vector<string> inputs;//输入变量别称
vector<string> outputs;//输出变量别称
string constants;//常量位
string garbage;//垃圾位
list<GateInCircuit> gateList;//门实例信息表
```

为了强调电路中门序列的线性逻辑, gateList 选用 list 结构。

2) 内部类 iterator

```
class iterator{
private:
    list<GateInCircuit>::iterator gateListIter;
private:
    friend class Circuit;
public:
    Circuit::iterator& operator = (const Circuit::iterator&);
    Circuit::iterator& operator ++ ();
    const Circuit::iterator operator ++ (int);
    bool operator == (const Circuit::iterator&);
    bool operator != (const Circuit::iterator&);
    string get_gateName(void);
    vector<string> get_posi(void);
    int get_posiNum(void);
};
```

迭代器, 封装的是 list<GateInCircuit>::iterator, 用于迭代读写 gateList 内容。

3) read 方法

```
bool read(const string&);
```

功能：从指定路径读取量子电路信息文件内容。读取成功后 `useEnable` 置为 1。

参数：参数一为文件路径。

返回值：返回值为读取成功（1）或失败（0）。

4) `print` 方法

```
void print(void);
```

功能：按照量子电路信息文件（.real）格式，向标准输出流打印当前电路全部信息。

5) `begin` 方法

```
Circuit::iterator begin(void);
```

返回值：返回一个指向 `gateList` 中的第一条信息的迭代器。

6) `end` 方法

```
Circuit::iterator end(void);
```

返回值：返回一个指向 `gateList` 中最后一条信息之后位置的迭代器。

7) `back` 方法

```
Circuit::iterator back(void);
```

返回值：返回一个指向 `gateList` 中最后一条信息的迭代器。似 `stl` 中的 `back` 设计，`end` 方法返回指向 `back` 方法返回值之后的位置的迭代器。

8) 关于遍历

```
Circuit x;
Circuit::iterator i;
for(i = x.begin(); i != x.end(); ++i) loop();
```

9) `insert_gate` 方法

```
bool insert_gate(Circuit::iterator&, GateInCircuit&);
```

功能：向迭代器指向的位置插入一个门，原迭代器指向的门及之后的门都后移一位。

参数：参数一为迭代器；参数二为门的实例。

返回值：返回值为插入成功（1）或失败（0）。

10) `erase_gate` 方法

```
bool erase_gate(Circuit::iterator&);
```

功能：从迭代器指向的位置删除一个门，原迭代器指向的门及之后的门都前移一位。

参数：参数一为迭代器。

返回值：返回值为删除成功（1）或失败（0）。

该函数不再返回所删除的门的实例，是因为迭代器已经指向了该门，完全可以在删除前对

其进行所需操作。另外，`erase_gate` 方法实际是调用了 `list::erase`，尽管 `stl` 中 `erase` 方法不会清理原有数据，门实例的数据还在，但保险的做法是对该实例进行复制后对副本进行操作。

11) `insert_line` 方法

```
bool insert_gate(bool, string&);
```

功能：向指向的位置插入一个量子位线，插入位置只能是变量首位前或变量末位后。

参数：参数一代表插入位置，首位前为 0，末位后为 1；参数二为插入位线的变量名。

返回值：返回值为插入成功（1）或失败（0），变量重名返回失败。

12) `erase_line` 方法

```
bool insert_gate(bool);
```

功能：从指向的位置删除，删除位置只能是变量首位或变量末位。

参数：参数一代表插入位置，首位为 0，末位为 1。

返回值：返回值为删除成功（1）或失败（0）。

13) `changegate` 方法

```
bool changegate(Circuit::iterator&, ComplexGate*);
```

功能：在量子电路中，将迭代器指向的非基本门根据非基本门规则分解成基本门。

参数：参数一为指向非基本门的迭代器；参数二为该非基本门的分解规则。

返回值：返回值为分解成功（1）或失败（0）。若参数一的迭代器指向的不是非基本门，或者参数二给出的分解规则不适用于迭代器指向的门，都会返回分解失败。

14) `matchgate` 方法

```
bool matchgate(GateLib&);
```

功能：该方法是 QCS 架构量子电路真值计算的第一步，通过对非基本门调用 `changegate` 方法，根据门库规则将电路中所有非基本门转换成基本门。该方法还会更新 `quantumCost` 值。

参数：参数一为量子门库。

返回值：返回值为分解成功（1）或失败（0）。门库不支持该电路时返回失败。

15) `calculate` 方法

```
bool calculate(GateLib&, TruthInCircuit&);
```

功能：该方法是 QCS 架构量子电路真值计算的第二步，按照二值真值表真值顺序计算中小规模量子电路。中小规模是指参与运算的量子电路的位数限制为 31 位及以下。

参数：参数一为量子门库；参数二为用于保存计算结果的真值表。

返回值：返回值为计算成功（1）或失败（0）。若进行计算的量子电路中有非基本门，或

者门库不支持该电路，或者电路位数超过 31 位时返回失败。

由于位数的增加对计算规模的影响是指数级的， n 位量子位真值表规模为 2^n ，这是经典计算逻辑的自然限制。出于运算量的考虑，本版本中用一个 `unsigned int` 类型参数限制运算规模，32 位（32 位编译器），理论上最多可以进行含 32 位量子位电路的运算。保险起见将量子位上限定为 31 位，对包括 32 位及以上量子位的电路不能运算。

16) calculate_value 方法

```
bool calculate_value(GateLib&, string&, string&);
```

功能：calculate 的加强方法。没有位数限制，但一次只能计算一条真值。

参数：参数一为量子门库；参数二为输入真值模式串；参数三为保存计算结果的输出真值模式串。

返回值：返回值为计算成功（1）或失败（0）。若进行计算的量子电路中有非基本门，或者门库不支持该电路时返回失败。

17) refresh 方法

```
void refresh(void);
```

功能：刷新参数。本版本中主要用于刷新量子代价。

18) 符号重载

```
Circuit& operator += (const Circuit&);  
Circuit& operator = (const Circuit&);
```

19) get/set 方法

```
int get_variNum(void);  
int get_quantumCost(void);  
int get_gateNum(void);  
string get_circuitName(void);  
string get_gateLibName(void);
```

3.4. application 模块

application 模块属于 application 逻辑层，在本系统中是一个测试结构，基于 circuit 模块，定义于“QCSApp.h”中。该结构有一个基类 QuAlgorithm，设计目的是为量子算法提供一个通用的评估方法，而不仅仅是通过计算复杂度或统计运行时间。但由于这个想法还处于雏形，本系统中提供的方法不一定准确。

3.4.1 class QuAlgorithm

QuAlgorithm 类是虚基类，被设计为用户自定义量子算法的基类，即用户自定义的算法应是 QuAlgorithm 的派生类，且一个派生类代表一个算法。该基类中有两个记录信息，分别是运行时间（runTime）和运算量（instNum）。

1) 数据结构定义

```
clock_t runTime;//运行时间
int instNum;//算法运算量
```

对算法运算量的设计为：由于定义的所有可供调用的电路操作都在 Circuit 类中，因此对 Circuit 类中的基本操作赋予一定的权值，并在派生算法类的实例运行时统计 instNum。统计公式为权值求和，见公式 3.1。

$$\text{instNum} = \sum (\text{weight} \times \text{cycle}) \quad (3.1)$$

2) get/set 方法

```
virtual string get_runTime(void) = 0;
virtual string get_instNum(void) = 0;
```

3.4.2 说明

逻辑设计中将用户调用类库设计的可重用功能或算法全归为 application 结构。但用户可以跳过不稳定的 QuAlgorithm 基类，直接设计功能。实际编程时，无论是否使用 QuAlgorithm 类，都推荐包含“QCSApp.h”头文件而不是“Circuit.h”头文件。

```
#include <QCSApp.h>
```

第 4 章 QCSUI 功能分析与测试

广义的 QCSUI 指任何通过读取 QCS 文件，调用类库方法，输出或演示量子电路、量子真值表等等仿真信息，进行量子计算仿真的演示插件。QCSUI 在架构中属于逻辑设计的 application 层，是逻辑功能“量子电路演示方法”的物理实现。本章中 QCSUI 特指 application 层的一个基于 QT 开发的量子计算应用软件。

本章分三个部分介绍：QCSUI、QCSShell 和粒子概率分布灰度仿真图。其中，QCSUI 是一个集合了 QCS 架构所提供计算功能的应用软件；QCSShell 是一个集合了计算功能和简单电路操作功能的控制台软件；粒子概率分布灰度仿真图是显示量子电路逻辑真值的一个可视方法。

4.1. QCSUI

QCSUI 设计为：QCS 框架下一个专门用于进行中小规模量子电路运算的可视平台。其中中小规模量子电路是指位线在 31 位及以下的量子电路。QCSUI 在加载电路和门库后先根据门库规则将电路转换成基本门组成的量子电路，再对基本门量子电路进行逻辑运算，最终得到真值表。基本门量子电路以及真值表都表现为信息文件，信息文件可以通过专门的可视平台转换成仿真图。

QCSUI 调用了 circuit 层提供的电路缓存、门库缓存、基本门转换（matchgate）和对中小规模基本门量子电路计算真值表（calculate）等方法。

QCSUI 集合了 5 个功能：打开 QCSShell 功能、加载量子门库功能、加载并显示量子电路信息功能、量子电路基本门转换功能以及小规模量子电路计算功能。除打开 QCSShell 功能外，其余四个功能运行时有部分逻辑先后顺序，推荐的操作顺序流程如图 4-1，且由于该操作流程仅是推荐，操作的主体是使用用户。QCSUI 运行界面如图 4-2。

4.1.1 功能分析与测试

1) 打开 QCSShell 功能

QCSShell 是一个控制台界面，相较 QCSUI 还集合了 QCS 架构中部分对电路构造的操作功能，对 QCSShell 的介绍详见 4.2.节。打开 QCSShell 功能的运行效果如图 4-3。

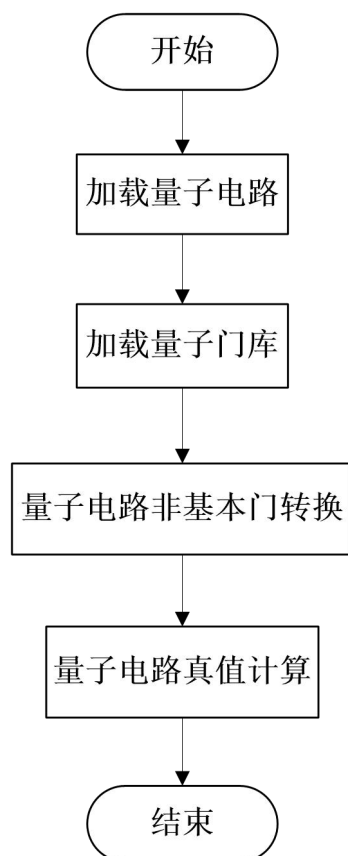


图 4-1 QCSUI 推荐操作顺序流程图

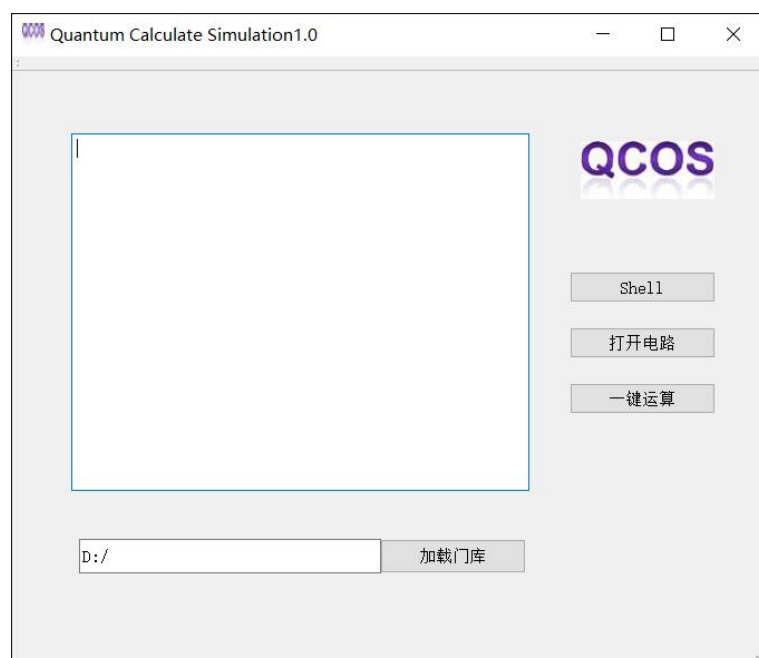


图 4-2 QCSUI 运行界面

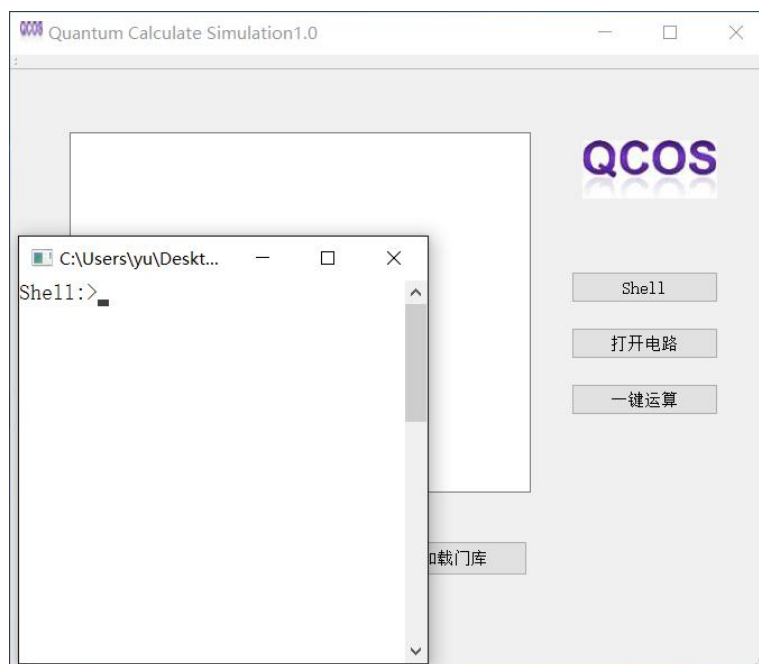


图 4-3 打开 QCShell 功能运行效果图

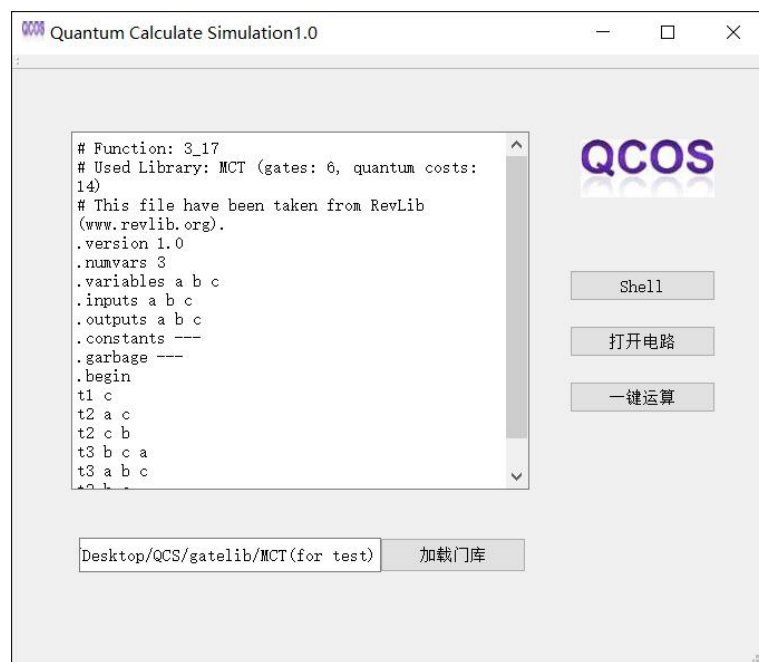


图 4-4 加载电路及加载门库功能运行效果图

2) 加载并显示量子电路信息功能

简称加载显示电路功能，其中加载量子电路是进行量子电路运算的必要前提之一。无论仿真实现或者物理实现量子电路，都是根据某个合理的量子门库提供的量子门构造量子电路。在 QCS 架构以及 QCSUI 中，这种逻辑简化为，选择加载一个量子电路和一个相对应的量子门库是进行量子电路运算的必要前提。

该功能包括打开量子电路信息文件（.real）以及将电路信息文件打印在显示框中，实际上通过 `circuit` 层调用了 `gate` 层的量子电路缓存方法。如图 4-4 所示，该图中加载并显示的电路是 `revLib` 提供的 3 线 6 门量子电路 “3_17”，该电路信息保存在量子电路信息文件（.real）中，文件名没有实际意义，修改也不会有任何影响。显示的量子电路信息与量子电路信息文件（.real）格式相同，主要包括量子电路名称、构造类库名、门数量、变量即量子位个数以及量子电路内容信息等等，具体参见 2.1.1 节 `RevLib` 量子电路信息文件。

`QCSUI` 中一次仅能加载并缓存一个量子电路。

3) 加载量子门库功能

加载门库同样是进行量子电路运算的必要前提之一。在流程图图 4-1 中，该功能在加载量子电路之后执行，由于加载并显示电路功能会在显示框中显示电路的构造门库名，可以根据门库名选择门库文件，因此在推荐操作流程中将加载门库放置在加载电路后进行，但其实这两步的执行顺序可以不分先后。

该功能包括打开量子门库信息目录文件以及将门库文件路径打印在门库路径显示框中，实际上通过 `circuit` 层调用了 `gate` 层的量子电路门库缓存方法。如图 4-4 所示，该图所示案例中不但加载并显示了量子电路 “3_17”，还加载了门库 “MCT(for test)”。“MCT(for test)” 门库是独立总结的且仅用作测试的，它符合门库要求并符合 `MCT` 门库的所有逻辑设计。门库目录文件名为 “MCT(for test)”，显示在门库路径显示框中，文件名没有实际意义，修改也不会有任何影响。门库目录文件中，门库表信息文件的门库名字段标识了门库，在进行量子电路运算时会先测试加载门库名以及量子电路的构造门库名是否匹配，匹配成功方可进行运算。

`QCSUI` 中一次仅能加载并缓存一个量子电路门库。

4) 量子电路基本门转换功能

量子电路基本门转换属于量子电路运算的第一步。将电路中所有非基本门转换成基本门然后进行计算，是 `QCS` 架构的运算设计。该功能需要一个缓存的量子电路和一个缓存的量子门库作为输入，输出是进行基本门转换后的量子电路信息。该功能实际上调用了 `circuit` 层的基本门转换方法（`circuit::matchgate`）。

如图 4-5 所示，该案例是对量子电路 “3_17” 根据门库 “MCT(for test)” 中的非基本门规则进行基本门转换，显示框中显示的量子电路全部由门库中的基本门构成，且逻辑与原电路相同，门数量（`gates`）变为 14 个，量子代价（`quantum cost`）保持不变。另外，由于电路名不是

对量子电路的唯一标识，所以基本门转换后的量子电路名（Function）保持不变，图 4-5 中基本门转换后的量子电路名称仍是“3_17”。

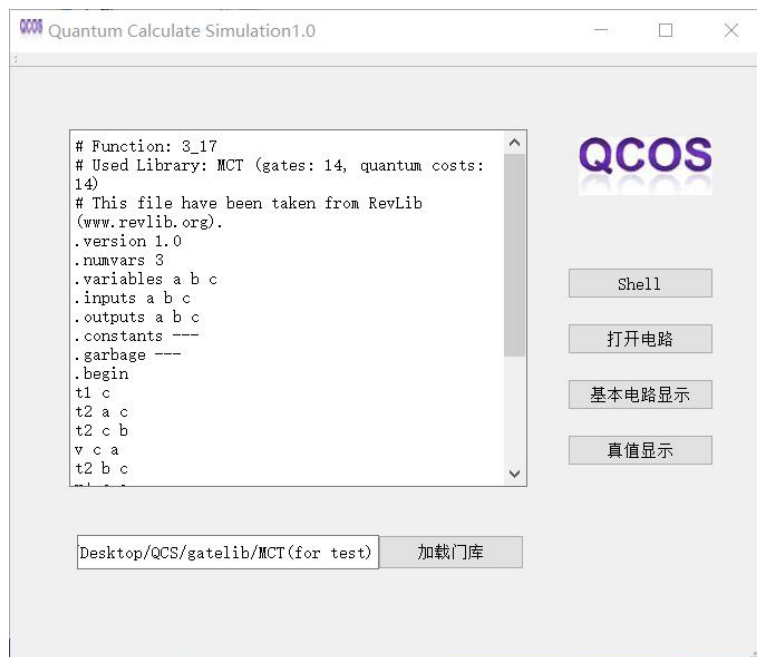


图 4-5 基本门转换功能运行效果图

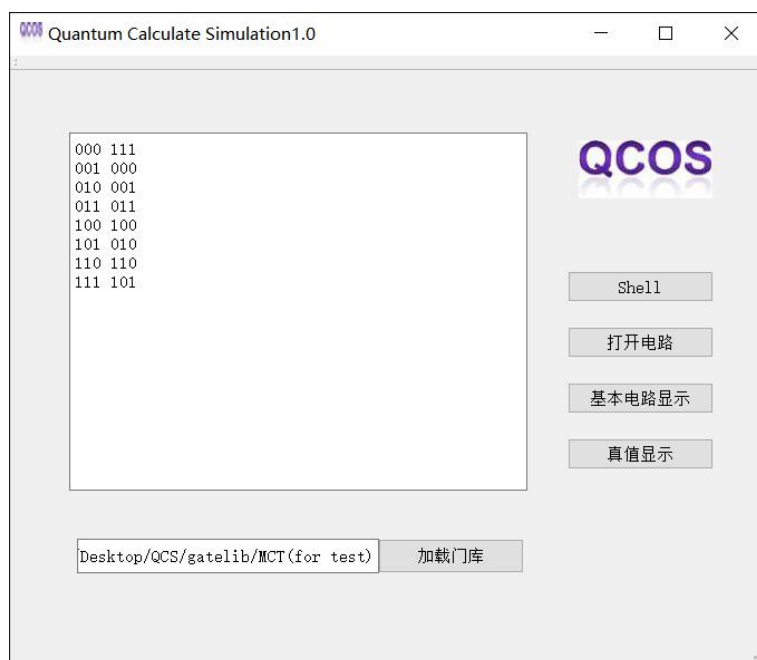


图 4-6 量子电路计算功能运行效果图

5) 小规模量子电路计算功能

简称量子电路计算功能，如 4.1.1 节引言介绍，小规模量子电路是指 31 位及以下的量子电路。该功能是量子电路运算的第二步即对基本门电路进行运算。该功能需要一个经过量子电路基本

门转换后的基本门电路和一个相应的量子门库作为输入，输出是量子电路真值表信息。该功能实际上调用了 `circuit` 层的量子电路真值表计算方法 (`circuit::calculate`)。真值中若出现叠加态，将用门库信息文件中为叠加态设置的别称进行显示。

如图 4-6 所示，该案例是对已经基本门转换的量子电路 “3_17” 根据门库 “MCT(for test)” 中的基本门规则进行基本门电路真值表计算。该功能将量子电路计算得到的真值信息打印在显示框中，打印格式同真值信息文件 (`.tr`) 格式。显示框中每一行即真值表的变换信息，空格前代表输入的真值，空格后代表计算后的真值结果。

4.1.2 利用 QCSUI 测试 QCS 运算正确性

本小节将利用 QCSUI 的计算功能测试 QCS 运算的正确性，逻辑上展现为在 `application` 层中测试 `circuit` 层及 `gate` 层功能。本测试不能完全证明 QCS 运算正确性。

1) 测试设计

设计门库 “MCT(toffoli)” 和门库 “MCT(for test)”。

其中 “MCT(toffoli)” 门库仅由一个 2 控制位 1 目标位的量子门 “t3” 门构成，该门设计为该门库中的唯一基本门，量子门库表信息文件如图 4-7 所示，“t3” 门的门类型 (`gateType`) 为 0 表示其为基本门。

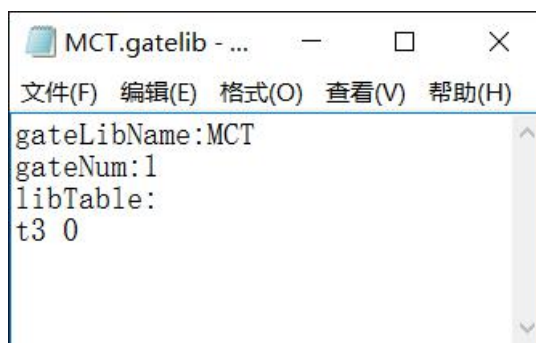


图 4-7 “MCT(toffoli)” 门库表信息文件

“MCT(for test)” 门库认为 “t3” 门是非基本门，包含完整的可分解 “t3” 门的非基本门分解规则。“MCT(for test)” 门库表信息文件如图 4-8 所示，“t3” 门的门类型 (`gateType`) 为 1 表示其为非基本门。

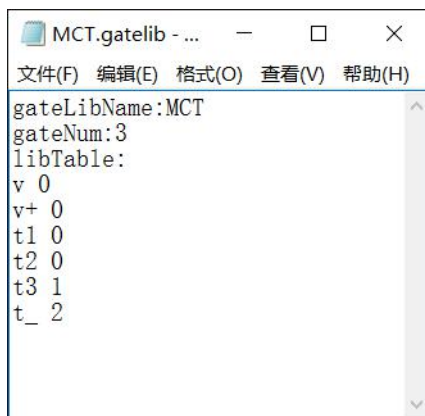


图 4-8 “MCT(for test)” 门库表信息文件

构建一个仅由一个“t3”门组成的量子电路“t3”，如图 4-9。在 QCSUI 中先后加载“MCT(t3)”门库以及“MCT(for test)”门库对“t3”电路进行量子电路运算。若根据两个门库规则进行计算得到的电路真值表结果相同，则可以认为 QCS 架构的量子电路运算功能在本案例中得到验证。

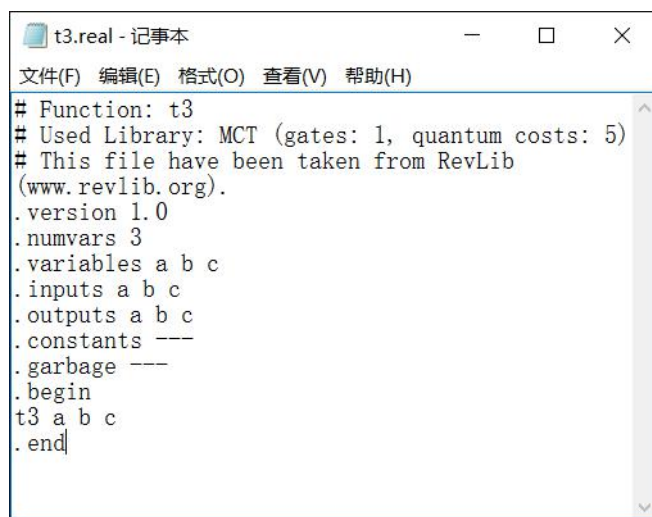


图 4-9 “t3” 量子电路信息文件

2) 测试结果

加载“t3”电路以及门库“MCT(toffoli)”并对其进行量子电路运算，由于该门库认为“t3”是基本门，不存在基本门转换，因此仅展示真值表运算结果，如图 4-10 所示。

加载“t3”电路以及门库“MCT(for test)”并对其进行量子电路运算，基本门转换的量子电路如图 4-11 所示，真值表运算结果如图 4-12 所示。

3) 测试分析

比较图 4-10 和 4-12 运算所得真值表，两张真值表完全相同，验证成功。



图 4-10 对“t3”用“MCT(toffoli)”门库运算所得真值表

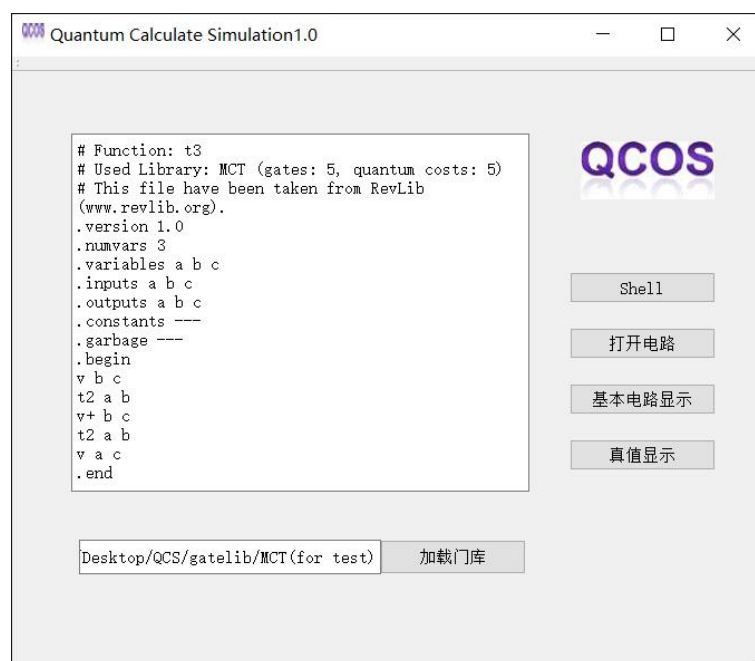


图 4-11 对“t3”用“MCT(for test)”门库转换所得基本门电路

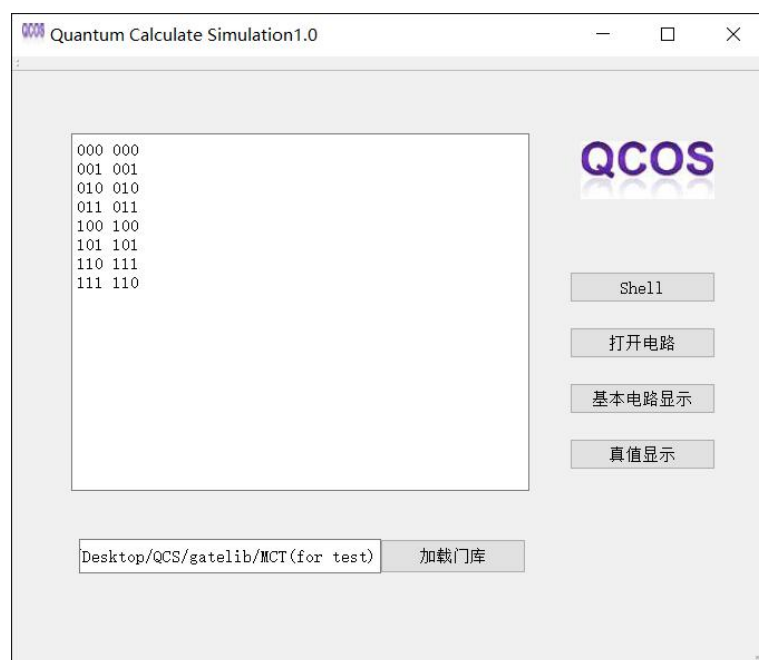


图 4-12 对“t3”用“MCT(for test)”门库运算所得真值表

4. 2. QCSShell

QCSShell 设计为：QCS 框架下 application 层中，在 QCSUI 包含功能基础以外集成了简单的电路构造操作功能的控制台平台。除了拥有与 QCSUI 一样的量子电路运算功能，QCSShell 还拥有删除或者添加末位量子门的功能。它的功能比 QCSUI 更加全面，还体现在可以选择多种输入输出方式上。QCSShell 调用了 circuit 层提供的电路缓存、门库缓存、基本门转换（matchgate）、对中小规模基本门量子电路计算真值表（calculate）、添加门（insert_gate）、删除门（erase_gate）等方法。

QCSShell 的运行界面如图 4-13 所示。在 QCSShell 中操作不当仅会提示指令使用失败（“failed”），不会影响之前进行的操作。

4. 2. 1 QCSShell 指令集

QCSShell 是一个通过指令控制电路操作的控制台平台。在这个平台上不仅可以进行量子计算，还可以完成部分量子电路操作的功能，功能列表如表 4-1 所示。

与 QCSUI 中功能执行有先后顺序类似，QCSShell 中部分指令之间有潜在的逻辑先后关系：CREC 和 LOAGL 指令执行后才可以执行 MATCH、CALCULATE_C 指令；CREC 指令执行之

后才可以执行 INSG、ERAG、PRINT_C、PRINT_F 指令；HELP、CLEAR 指令没有执行要求。

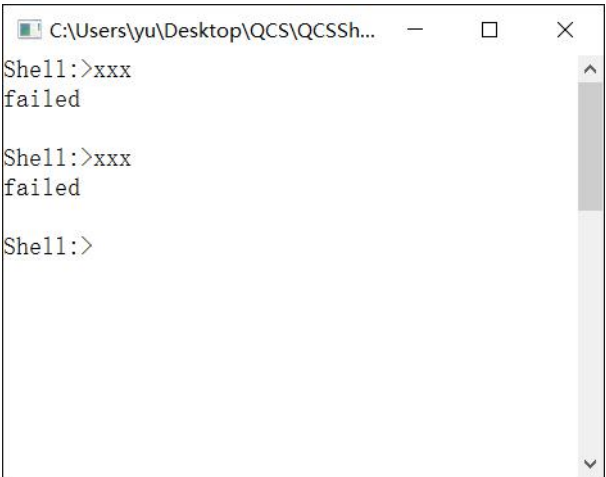


图 4-13 QCSShell 运行界面

表 4-1 QCSShell 指令集功能表

指令全称	指令格式	指令功能
Create Circuit	CREC 电路名 (换行)电路读取路径	根据指定路径读取并创建电路
Load Gatelib	LOAGL 门库名 (换行)门库读取路径	根据指定路径读取门库
Insert Gate	INSG 电路名 门名 门线位	在指定电路的末位添加门
Erase Gate	ERAG 电路名	在指定电路的末位删除门
Print on CMD	PRINT_C 电路名	在控制台上打印指定电路
Print on File	PRINT_F 电路名 (换行)电路读取路径	在指定路径打印指定电路
Match Gate	MATCH 电路名 门库名	对指定电路根据指定门库进行门匹配
Calculate	CALCULATE_C 电路名 门库名	在控制台上打印对指定电路根据指定门库进行门计算结果
Help	HELP	帮助
Clear	CLEAR	清屏

4.2.2 指令功能分析与测试

1) CREC 指令

从量子电路信息文件（.real）中读入并缓存量子电路，原理与 QCSUI 加载量子电路功能相同。指令执行与测试结果如图 4-14。

QCSShell 中可以缓存多张量子电路，在缓存量子电路时为每个量子电路绑定一个可唯一标识的别名。如图 4-14 中，为缓存电路绑定别名为“a”。

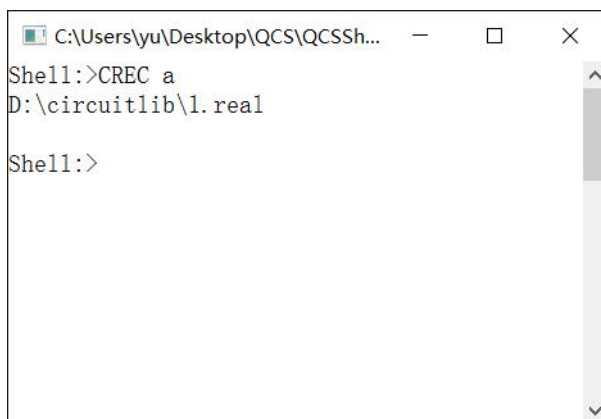


图 4-14 “CREC” 创建电路指令

2) LOAGL 指令

从量子门库信息文件中读取并缓存量子门库，原理与 QCSUI 加载量子电路功能相同。指令执行与测试结果如图 4-15。

同样，QCSShell 可以缓存多个量子门库，并在缓存门库时为每个门库绑定一个可唯一标识的别名。如图 4-15 中，为缓存门库绑定别名为“a”，门库别名与电路别名可以重名。

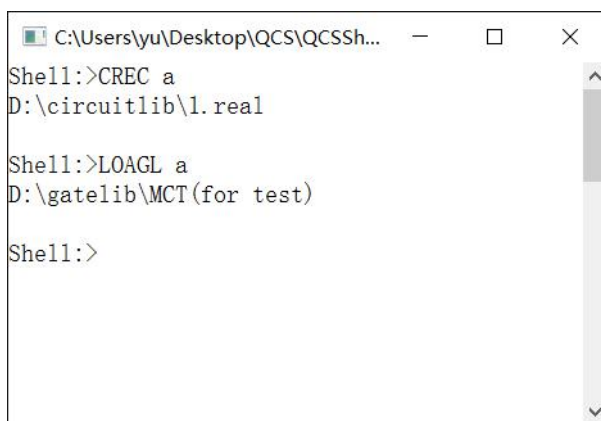
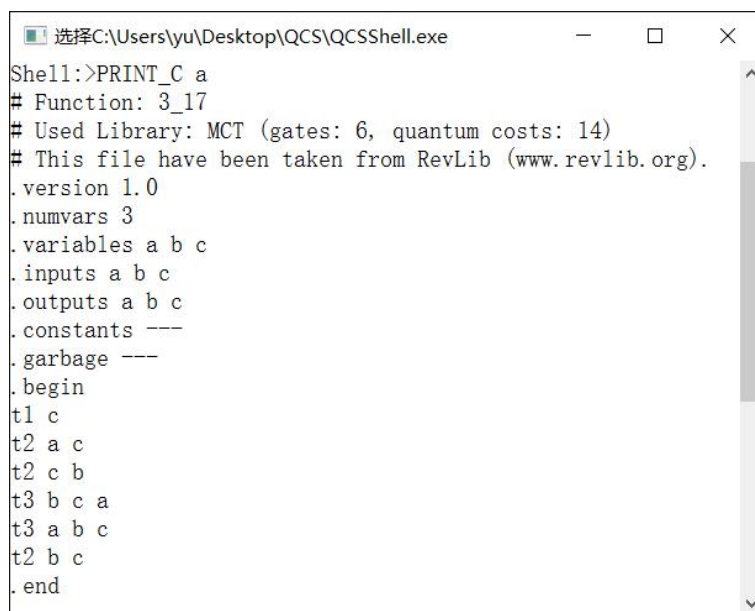


图 4-15 “LOAGL” 加载门库指令操作图

3) PRINT_C 指令

在 QCSShell 控制台上打印电路，指令参数是于缓存量子电路绑定的别名。指令执行与测试结果如图 4-16。



```

选择C:\Users\yu\Desktop\QCS\QCSShell.exe
Shell:>PRINT_C a
# Function: 3_17
# Used Library: MCT (gates: 6, quantum costs: 14)
# This file have been taken from RevLib (www.revlib.org).
.version 1.0
.numvars 3
.variables a b c
.inputs a b c
.outputs a b c
.constants ---
.garbage ---
.begin
t1 c
t2 a c
t2 c b
t3 b c a
t3 a b c
t2 b c
.end

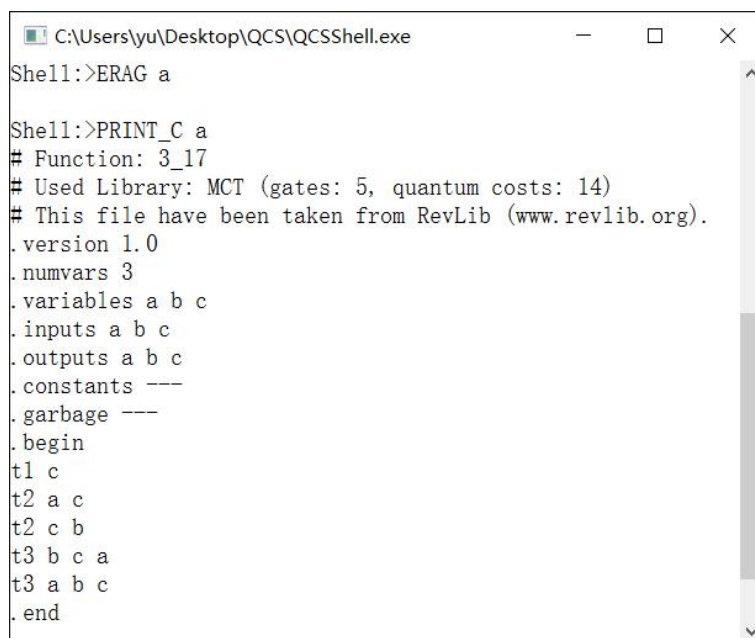
```

图 4-16 “PRINT_C” 控制台打印电路指令

4) ERAG 指令

删除指定电路的末位门，指令参数是与缓存量子电路绑定的别名。指令执行与测试结果如图 4-17，该案例中删除了量子电路“a”的末位门“t2 b c”。

QCSShell 仅提供小规模电路构造操作，无论是删除门或者添加门仅提供对末位门操作。



```

C:\Users\yu\Desktop\QCS\QCSShell.exe
Shell:>ERAG a

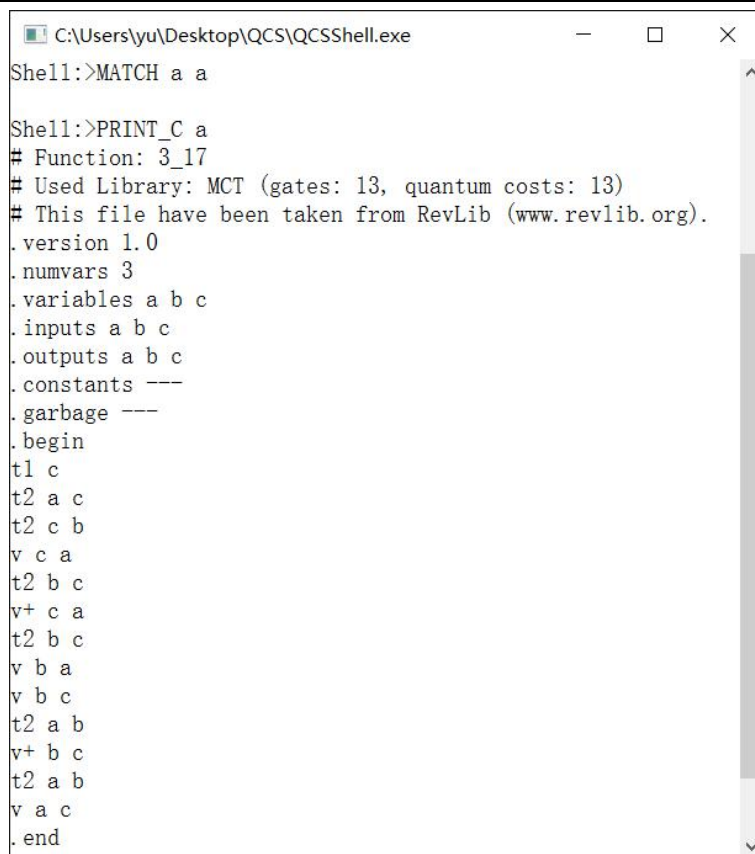
Shell:>PRINT_C a
# Function: 3_17
# Used Library: MCT (gates: 5, quantum costs: 14)
# This file have been taken from RevLib (www.revlib.org).
.version 1.0
.numvars 3
.variables a b c
.inputs a b c
.outputs a b c
.constants ---
.garbage ---
.begin
t1 c
t2 a c
t2 c b
t3 b c a
t3 a b c
.end

```

图 4-17 “ERAG” 删除末位门指令

5) MATCH 指令

对指定的缓存电路使用指定的缓存门库规则进行非基本门转换。指令执行与测试结果如图 4-18，该案例中对量子电路“a”采用了量子门库“a”的规则进行了转换。



```

C:\Users\yu\Desktop\QCS\QCShell.exe
Shell:>MATCH a a

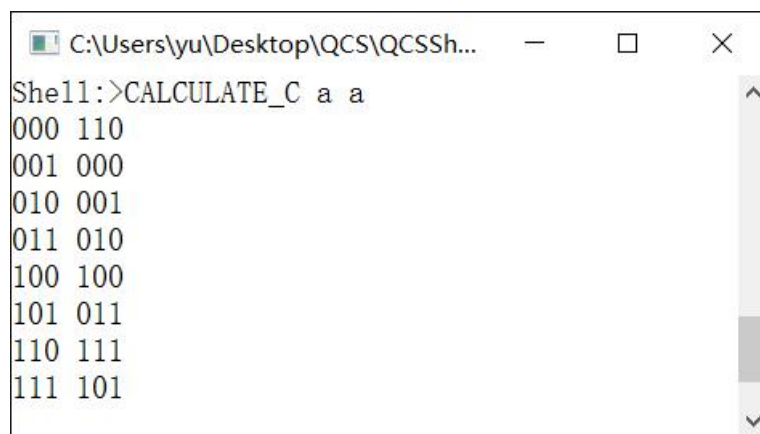
Shell:>PRINT_C a
# Function: 3_17
# Used Library: MCT (gates: 13, quantum costs: 13)
# This file have been taken from RevLib (www.revlib.org).
.version 1.0
.numvars 3
.variables a b c
.inputs a b c
.outputs a b c
.constants ---
.garbage ---
.begin
t1 c
t2 a c
t2 c b
v c a
t2 b c
v+ c a
t2 b c
v b a
v b c
t2 a b
v+ b c
t2 a b
v a c
.end

```

图 4-18 “MATCH” 电路基本门化指令操作图

6) CALCULATE_C 指令

中小规模电路真值表计算，并打印在控制台上。指令执行与测试结果如图 4-19，该案例中对量子电路“a”采用了量子门库“a”的规则进行了量子计算。



```

C:\Users\yu\Desktop\QCS\QCSh...
Shell:>CALCULATE_C a a
000 110
001 000
010 001
011 010
100 100
101 011
110 111
111 101

```

图 4-19 “CALCULATE_C” 计算并控制台打印真值指令

4.3. 粒子概率分布灰度仿真图

对量子计算真值进行灰度仿真的概念源自 IG Karafyllidis^[4]。如图 4-19 为“粒子概率分布

灰度仿真图”，图的右边是灰度仿真对比条，模拟了每个状态位置真值的平方，按由从小到大灰度增加。以图的左下角为原点，横轴数据表示的是量子电路图中初始位置和经过每个门的状态位置以及最终位置的真值区域，纵轴表示的是真值的输入情况以二进制转化成十进制形式表示。对于整个框图的每一个位置区域表示的即使量子电路真值的灰度表示。以横轴的真值变化来看便是每一种输入真值状态在电路每经过一个量子门位置后的真值变化，以纵轴的真值变化来看便是所有的真值情况经过每一个量子门位置后的概率分布变化。

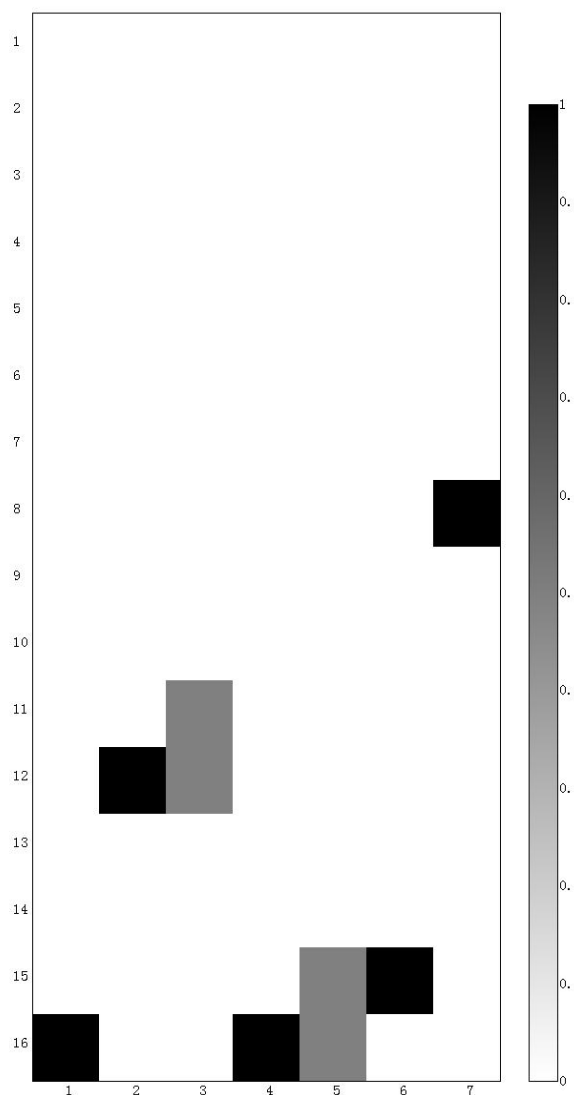


图 4-20 粒子概率分布灰度仿真图

以图 4-20 所示案例为例，图上纵轴共有 16 行，由于二值逻辑真值表的输入真值不存在纠缠态，因此该图的 16 个态必由 4 条量子位线构成。最初的输入态只有一个，体现在图中为第一列只有第 16 行（左下）为黑，该态表示为 $|1111\rangle$ ，代表全部粒子都处于该态状态。随着横轴

发展，粒子状态开始出现变化：经过第一个门后，所有粒子状态变为第 12 行的状态，即 $|1011\rangle$ ；经过第二个门后，一半粒子处于第 11 行状态即 $|1010\rangle$ ，一半粒子处于第 12 行状态即 $|1011\rangle$ ，这就是纠缠态的表现；经过第三个门后，所有粒子又全变为 $|1111\rangle$ 状态；以此类推。最终全部粒子回归到第 8 行状态，即 $|0111\rangle$ ，这也是该量子电路的逻辑变换结果。

第 5 章 总 结

1) 这是“量子电路仿真平台”的第二个版本。前一个版本“量子电路仿真平台 ver.0.0”为测试版，课题“基于量子电路的量子计算仿真平台”为国家级大学生创新创业训练计划项目，软件申请了专利“一种量子电路仿真平台”和软件著作权“量子电路仿真平台”。

2) “量子电路仿真平台 ver.0.0”相比，“量子计算仿真平台 ver.1”最大区别在于量子电路信息文件的完整设计：舍弃了原有的电路信息文件设计而采用“.real”文件格式；设计了量子门库表信息文件，从而使量子门库结构更加稳定。唯一的遗憾是仍然没有设计真值存储文件。

3) 所有的模拟计算方式都采用模式匹配方式。即缓存所有的输入输出可能性，并通过字符串模式匹配计算。这种方式不如矩阵计算有逻辑，但十分符合量子计算的特征——一次性完成所有计算。实际上，曾经测试过用矩阵方式计算，然而普通计算机通常止步于 6 条逻辑位线，最后只能不了了之。

4) QCS 架构可供设计的部分集中在 application 层，而可供修改内容的部分还包括 gate 层的门库。门库设计在文件中就是为了便于用户修改。QCSv1 仅提供了 MCT、NCV 两个门库。而根据规则，用户甚至可以设计一个以多位线门为基本门，提供真值逻辑，以甚多位线门为非基本门的门库，以满足计算需要。

5) QCSv1 设计了满足大规模电路计算的方法 calculate_value。

附录一 量子电路相关文件示例

1) 量子电路信息文件示例

```
.version 2.0
.numvars 3
.variables x0 x1 x2
.inputs 1 a b
.outputs f g1 g2
.constants 1--
.garbage -11
.begin
t1 x0
t3 x1 x2 x0
.end
```

2) 量子门库表信息文件示例

```
gateLibName:MCT
gateNum:6
libTable:
v 0
v+ 0
t1 0
t2 0
t_2
```

3) 非基本门规则信息文件示例

```
gateName:t3
gateLibName:MCT
gateType:1
controlNum:2
targetNum:1
ruleTable:
N v cb1:tb0
N t2 cb0:cb1
N v+ cb1:tb0
N t2 cb0:cb1
N v cb0:tb0
```

4) 基本门规则信息文件示例

```
gateName:v
gateLibName:MCT
gateType:0
controlNum:1
```

```
targetNum:1
quantumCost:1
truthTable:
0:0->0:0
0:1->0:1
0:a->0:a
0:b->0:b
1:0->1:a
1:1->1:b
1:a->1:1
1:b->1:0
```

5) 对图 2-1 用模板 ‘E’ 分解的规则

```
N v ce0:tb0
E t _ ce0:
N v+ ce0:tb0
E t _ ce0:
```

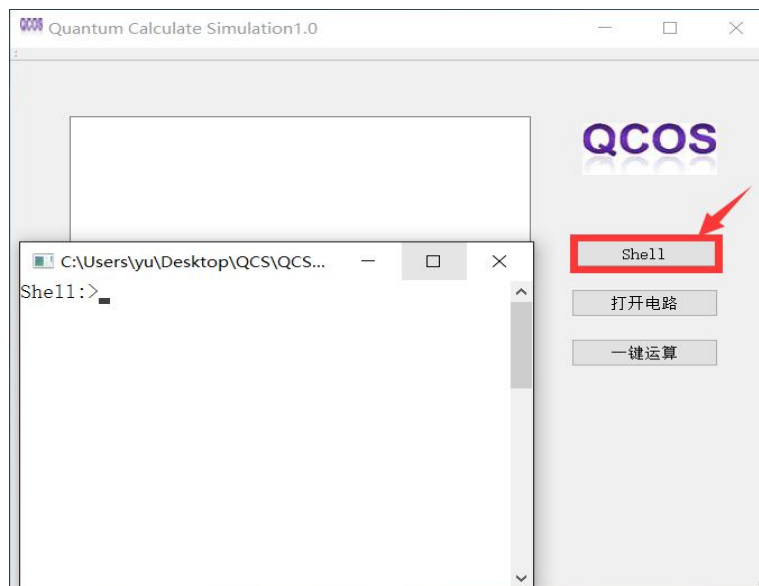
6) 对图 2-3 用模板 ‘X’ 分解的规则

```
N v :cb0
X cv cb0:cb1
N cnot :tb0
X cv+ ce1:ce0
N v+ :cb0
```

附录二 QCSUI 操作说明书

1) 打开 QCSShell

单击“Shell”按钮打开 QCSShell，QCSShell 使用见 4.2.节。

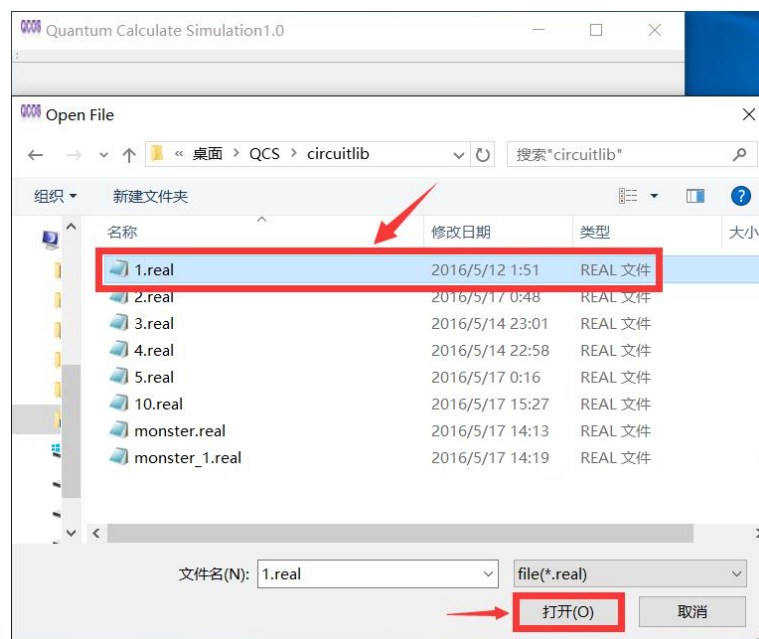


2) 打开电路

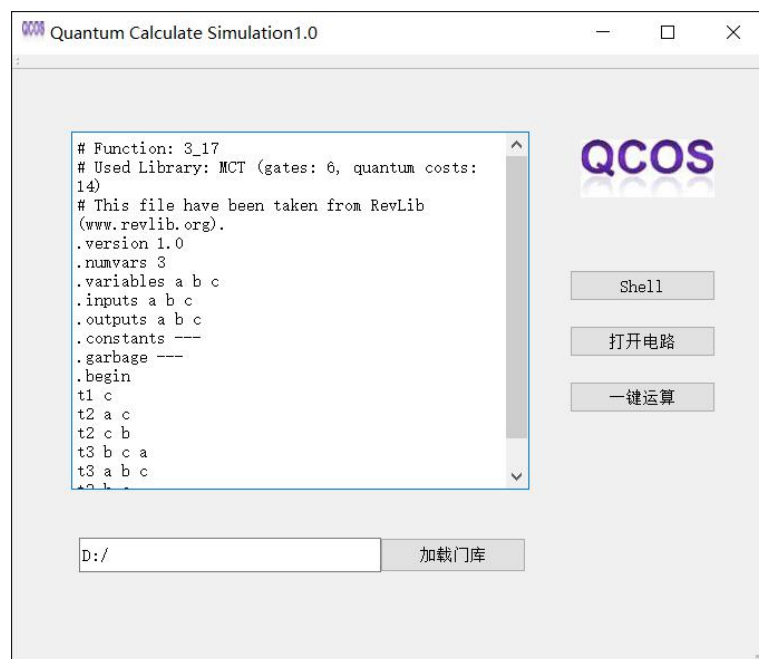
单击“打开电路”按钮，跳出量子电路选择对话框。



量子电路选择对话框中只允许选择“.real”文件。

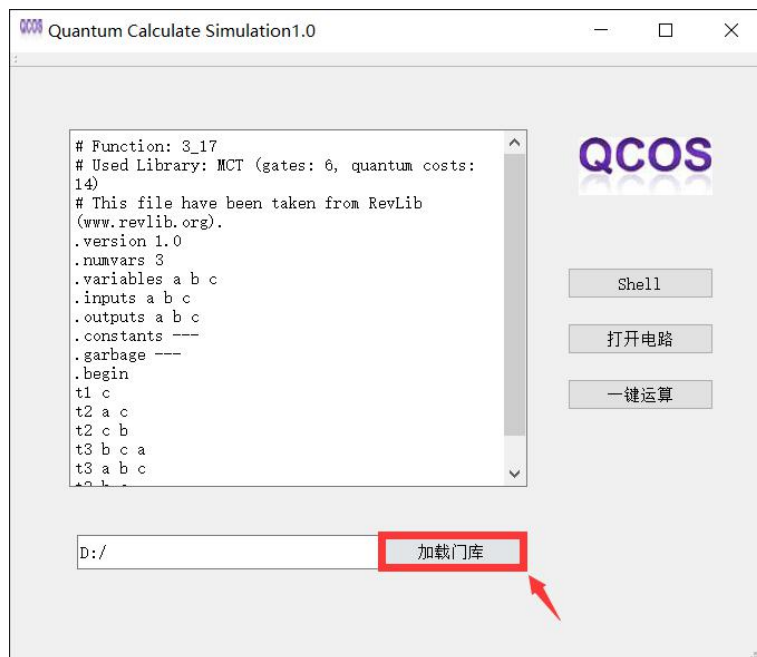


电路会立即显示在显示框中。

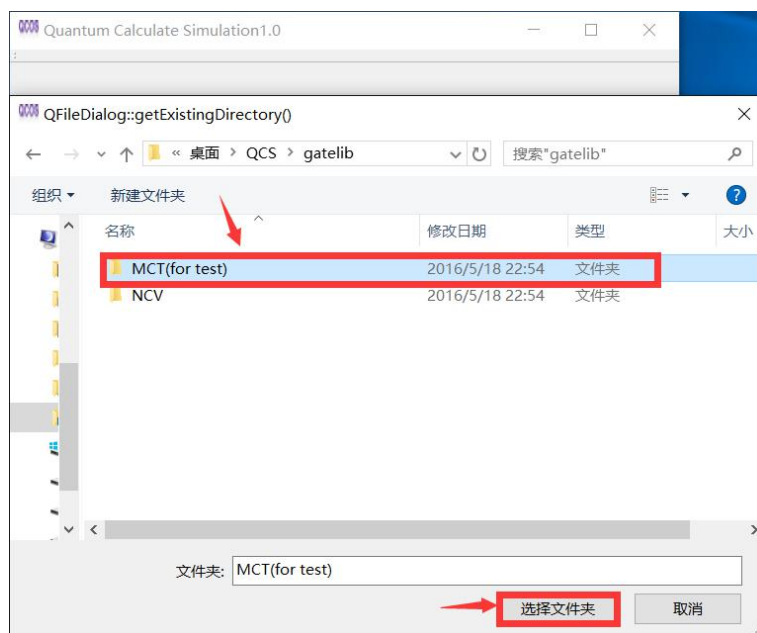


3) 加载门库

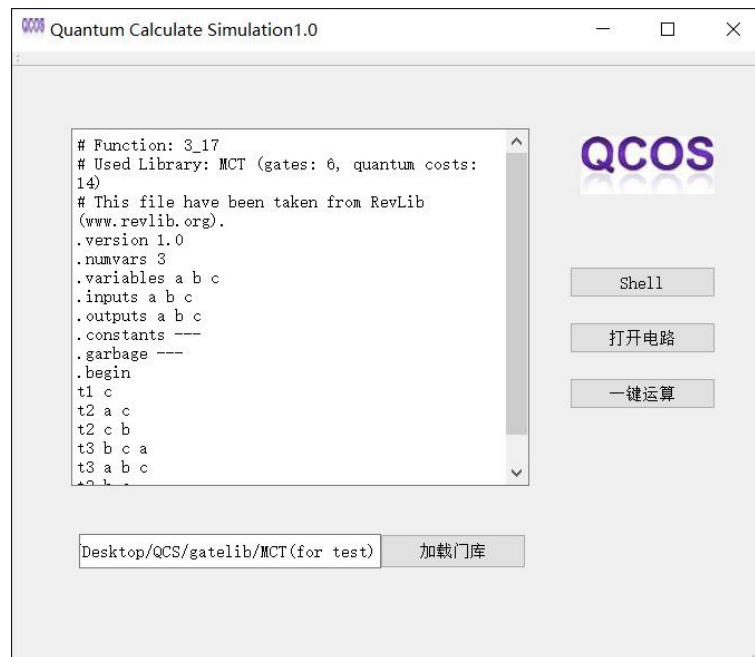
单击“加载门库”按钮，跳出门库文件选择对话框。



门库文件选择对话框中只允许选择文件夹。

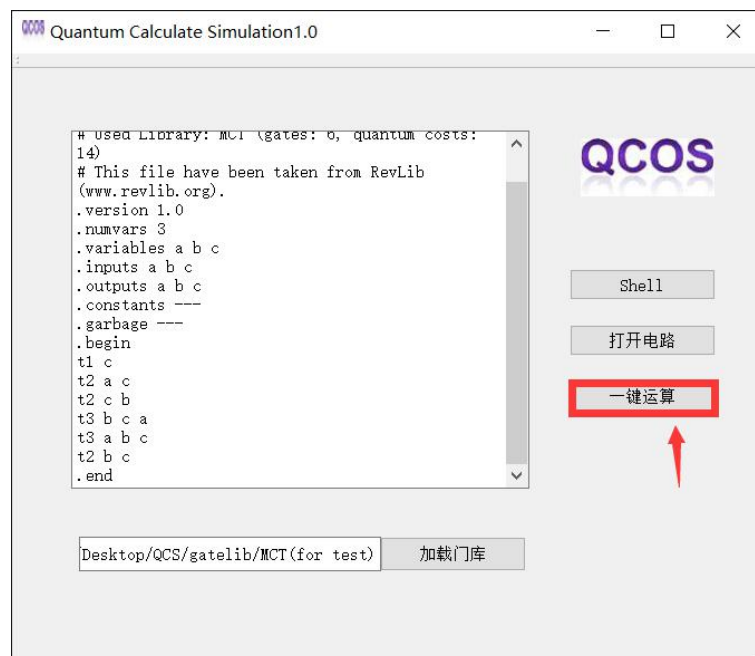


“加载门库”按钮左侧的门库路径显示框中会显示选择的门库路径。



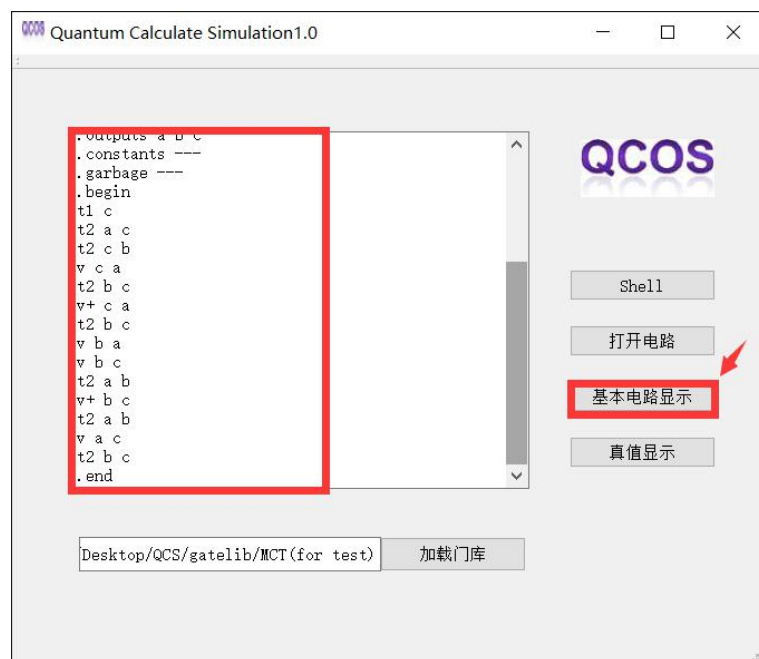
4) 一键运算

“一键运算”即量子电路运算功能，点击“一键运算”按钮后该按钮小时，同时触发“基本电路显示”和“真值显示”按钮显示。重新加载电路或门库会重新触发“一键运算”按钮显示。



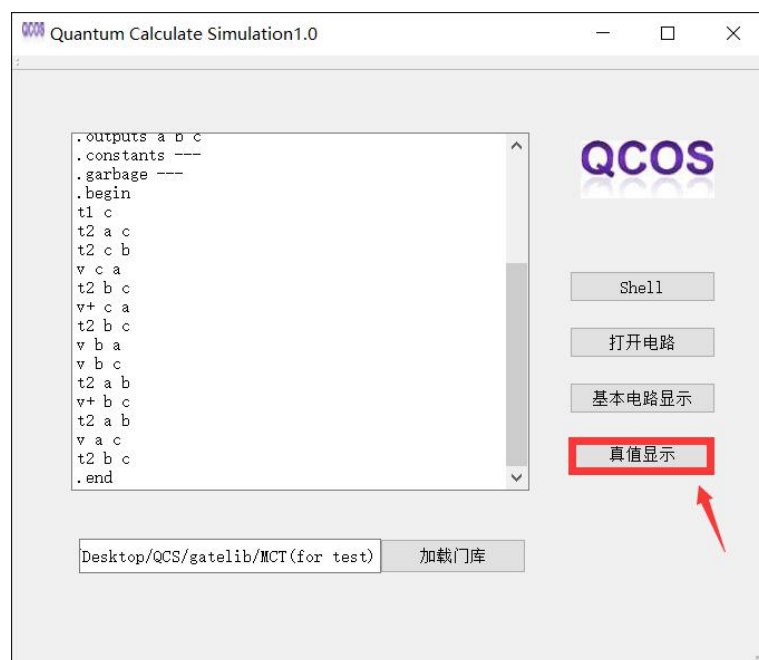
5) 基本电路显示

单击“基本电路显示”按钮，显示框中将按照量子电路信息文件（.real）格式显示电路。

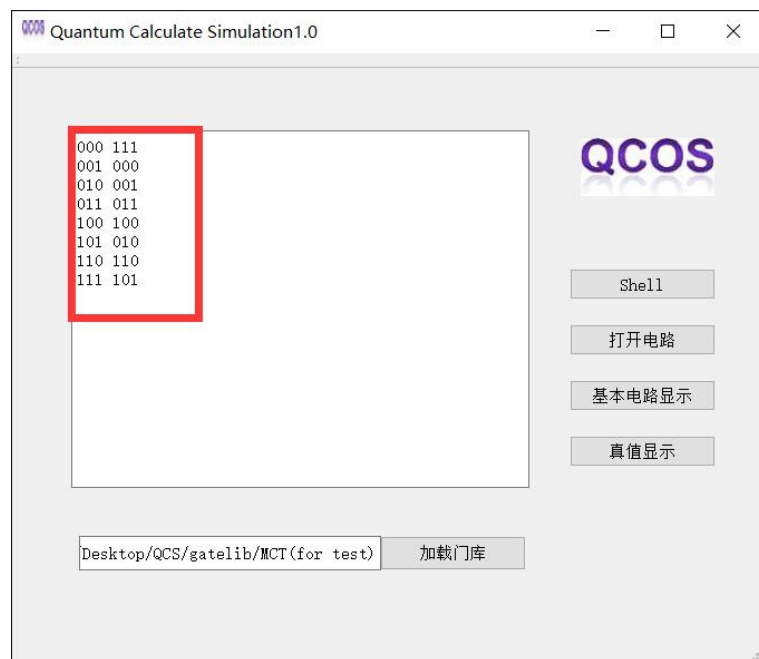


6) 真值显示

单击“真值显示”按钮。



显示框中将按照真值文件（.tr）格式打印真值表。



附录三 与本文相关的成果

A. 论文

- [1] 何金凤, 管致锦, 郁可人,等. 基于 NCV 门的量子电路故障的检测与定位[J]. 量子电子学报, 2015, 32(2):161-169.
- [2] 曲艺, 管致锦, 郁可人,等. Synthesis Algorithm Based on the Pre-evaluation of Quantum Circuits for Linear Nearest Neighbor Architectures[J]. Journal of Circuits, Systems, and Computers(收录).

B. 专利

- [1] 线性最近邻量子电路生成器. 管致锦, 鹿玉, 郁可人等. 申请号:20140745167.
- [2] 一种量子电路仿真平台. 郁可人, 管致锦, 潘雨坤等. 申请号:201510182366.

C. 软件著作权

- [1] LNN 架构的量子电路综合及其代价优化软件. 管致锦, 曲艺, 潘雨坤等. 登记号:2015SR097553.
- [2] 量子电路仿真平台. 郁可人, 管致锦, 姚林霞等. 登记号:2015SR097547.

D. 竞赛

- [1] 2014 年中国大学生计算机设计大赛江苏省级赛.软件应用与开发, 科学计算类, 本科组, 一等奖.

E. 项目

- [1] 大学生创新创业训练计划项目:线性最近邻可逆逻辑设计及其代价研究, 2013.05-2014.04. 项目编号 201310304028Z.
- [2] 大学生创新创业训练计划项目:基于量子电路的量子计算仿真平台, 2014.05-2015.04. 项目编号 201410304024Z.

参考文献

- [1] 管致锦.可逆逻辑综合[M].北京, 科学出版社, 2011.2.
- [2] revkit_user_manual[EB/OL].Available: www.revkit.org.
- [3] revlib_2_0[EB/OL].Available: www.revkit.org.
- [4] IG Karafyllidis.Quantum Computer Simulator Based on the Circuit Model of Quantum Computation[J].IEEE Transactions on Circuits & Systems I: Regular Papers, 2005, 52(8):1590-1596.
- [5] 李晓瑜.量子计算与量子信息中若干问题的研究[D].电子科技大学, 2014, pp. 19-40, 78-88.
- [6] Sasanian Z, Wille R, Miller D M.Realizing reversible circuits using a new class of quantum gates[C].Design Automation Conference.ACM, 2012:36-41.
- [7] 吕洪君, 郭俊旺, 彭斐, 吴天昊, 解光军.用基本两位量子逻辑门实现n位量子逻辑门的研究[J].量子电子学报, 2010, 27(1):26-30.
- [8] P.W.Shor.Algorithm for quantum computation:discrete logarithm and factoring[J]. Symposium on Foundations of Computer Science, 1994, pp. 124-134.
- [9] 马颖.基于量子计算理论的优化算法研究[D].西北工业大学, 2014, pp. 1-20.
- [10] 王晓晗, 曹怀信, 吴水艳.量子并行性及其应用[J].咸阳师范学院学报, 2008, 23(6):12-14.

致 谢

我从大一下学期开始加入管致锦老师带领的量子逻辑研究课题组，到现在大四即将毕业，除去半年的考研复习时间，大概在这个领域“玩”了三年有余。量子逻辑研究是个极其复杂的领域，横跨数学、物理、电子信息、计算机等等学科。经过三年多的学习研究，我才刚刚触及这个学科的边缘。

感谢管致锦老师当年接纳我进入课题组，让我接触这个领域的同时也激起了我学习其他课业的兴趣。同时感谢管老师多年来从学习、生活等各方面给予我的帮助。感谢程学云老师、朱鹏程老师在专业方面给予的极大帮助。感谢鲁灿刚、何金凤、徐海、鹿玉等学长学姐。感谢姚林霞、王伟煜、龚雨濛、严杨扬等课题组的学弟学妹。感谢课题组这个大家庭中的所有成员。最后感谢一直以来一起奋斗的曲艺、潘雨坤、徐蓓蓓同学，如果没有大家的互相扶持，绝对无法取得今天的成果。