

Gustavo Eulalio Miranda Cabral

*Uma Ferramenta para Projeto e Simulação
de Circuitos Quânticos*

Campina Grande, PB, Brasil

2004

Gustavo Eulalio Miranda Cabral

*Uma Ferramenta para Projeto e Simulação
de Circuitos Quânticos*

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Informática do Centro de Ciências e Tecnologia da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciências (MSc).

Orientadores:

Aécio Ferreira de Lima
Bernardo Lula Jr.

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE – UFCG

Campina Grande, PB, Brasil

2004

Ficha Catalográfica

CABRAL, Gustavo Eulalio Miranda

C117F

Uma Ferramenta para Projeto e Simulação de Circuitos Quânticos

Dissertação (mestrado), Universidade Federal de Campina Grande, Centro de Ciências e Tecnologia, Coordenação de Pós-Graduação em Informática, Campina Grande, Paraíba, Julho de 2004.

77 p. Il.

Orientadores: Aécio Ferreira de Lima
Bernardo Lula Jr.

Palavras-chave:

1. Computação Quântica
2. Simulação
3. Circuitos Quânticos

CDU – 519.71

**“UMA FERRAMENTA PARA PROJETO E SIMULAÇÃO DE
CIRCUITOS QUÂNTICOS”**

GUSTAVO EULÁLIO MIRANDA CABRAL


DISSERTAÇÃO APROVADA EM 21.07.2004



**PROF. BERNARDO LULA JÚNIOR, Dr.
Orientador**



**PROF. AÉRCIO FERREIRA DE LIMA, Dr.
Orientador**



**PROF. HERMAN MARTINS GOMES, Ph.D
Examinador**



**PROF. RUBENS VIANA RAMOS, Dr.
Examinador**

CAMPINA GRANDE – PB

Resumo

A Computação Quântica surgiu como uma promissora tecnologia alternativa capaz de contornar os problemas previstos para acontecerem em breve com a saturação da tecnologia atual (Computação Clássica). Estes problemas acontecem quando se trabalha com componentes muito pequenos, da ordem de poucos nanômetros, quando os efeitos da Mecânica Quântica começam a tornar-se mais fortes, gerando resultados aleatórios. A Computação Quântica toma proveito desses efeitos, resultando numa Computação não somente mais rápida, como um novo tipo de Computação, capaz de realizar tipos novos de computação impossíveis, em princípio, de serem realizadas classicamente. Embora a teoria da Computação Quântica venha se desenvolvendo com rapidez, a parte experimental ainda enfrenta grandes dificuldades. Embora implementações de algoritmos simples e com poucos qubits já tenham sido realizadas, ainda há problemas sérios de escalabilidade, descoerência e controlabilidade a serem resolvidos. Devido a esses problemas, torna-se imperativo o uso de simuladores tanto para o desenvolvimento de novos algoritmos quanto para o estudo dos algoritmos existentes. A simulação, porém, apresenta um problema, no que um sistema quântico só pode ser simulado eficientemente por outro sistema quântico. Um sistema quântico de n qubits levará um tempo $O(2^n)$ para ser simulado por um computador clássico. A simulação, entretanto, mostra-se a alternativa mais viável em vista da falta de um hardware quântico. Este trabalho apresenta o processo de concepção e implementação de um simulador universal de circuitos quânticos, denominado *Zeno*. O modelo de circuitos quânticos foi escolhido por ser amplamente difundido e de fácil assimilação. O trabalho apresenta uma introdução aos conceitos de Computação Quântica usados na implementação do simulador, os requisitos gerais desejáveis para um novo simulador de circuitos quânticos, tanto do ponto de vista funcional quanto do não-funcional, o processo de desenvolvimento utilizado e uma análise dos resultados obtidos com a implementação do simulador.

Abstract

Quantum Computation appeared as a promising alternative technology capable of turn around the problems predicted to happen soon with the saturation of the current technology (Classical Computation). These problems occur when one works with very small components, in the order of a few nanometers, when the Quantum Mechanics effects start to become stronger, generating random results. Quantum Computation takes advantage of those effects, resulting in a Computation not only faster, but a new kind of Computation, capable of realizing new types of computations impossible, in principle, to be realized classically. Although the Quantum Computation theory has been developing fast, the experimental part still faces great difficulties. Though implementations of simple algorithms, with a few qubits, have already been done, there are still serious problems of scalability, decoherence and controllability to be solved. Due to these problems, the use of simulators becomes mandatory either for the development of new algorithms and for the study of existing algorithms. Simulation, however, has a problem, in that a quantum system can only be efficiently simulated by another quantum system. A quantum system of n qubits will take time $O(2^n)$ to be simulated by a classical computer. Simulations, however, results the most available alternative due to the lack of a quantum hardware. This work presents the process of conception and implementation of a universal quantum circuit simulator, named *Zeno*. The quantum circuit model was chosen due to it being an easily assimilated and widely known model. The work presents an introduction to the concepts of Quantum Computation used in the simulator's implementation, the general requisites wanted for a new quantum circuit simulator, as from the functional point of view as from the non-functional, the development process used and an analysis of the results obtained with the simulator's implementation.

Sumário

Lista de Tabelas

Lista de Figuras

1	Introdução	p. 13
1.1	Computação Quântica	p. 13
1.1.1	O Computador Quântico	p. 14
1.2	Simuladores	p. 15
1.2.1	Simuladores Quânticos	p. 15
1.2.1.1	Problemas da Simulação	p. 16
1.2.2	Estado Atual	p. 16
1.2.3	Deficiências	p. 17
1.2.3.1	Funcionalidade	p. 17
1.2.3.2	Usabilidade	p. 17
1.2.3.3	Documentação do Desenvolvimento	p. 17
1.3	Objetivo	p. 18
1.4	Relevância	p. 18
1.5	Metodologia de Trabalho	p. 18
1.5.1	Metodologia de Desenvolvimento	p. 19
1.5.2	Análise de Resultados	p. 19
1.6	Estrutura da Dissertação	p. 19
2	Fundamentação Teórica – Computação Quântica	p. 21
2.1	Representação da Informação	p. 21
2.2	Processamento da Informação	p. 23
2.3	Portas de Um Qubit	p. 23
2.4	Rotações	p. 24

2.4.1	Decomposição em Rotações	p. 25
2.5	Portas Controladas de Dois Qubits	p. 25
2.6	Computação Reversível	p. 26
2.7	Computação Quântica	p. 27
2.8	Portas Quânticas Universais	p. 29
2.8.1	Universalidade de Hadamard e as Portas de Fase	p. 29
2.8.2	Decomposição de $C(U)$	p. 29
2.8.3	Decomposição de $C^2(U)$	p. 30
2.8.4	Decomposição em Produto de Matrizes de Dois Níveis	p. 31
2.8.5	Universalidade por Aproximação	p. 31
2.8.5.1	Universalidade de <i>Hadamard</i> , $\pi/8$ e <i>CNOT</i>	p. 32
2.9	Medição	p. 33
2.9.1	Medição Projetiva	p. 33
2.9.2	POVM	p. 34
2.10	Estados Mistos	p. 34
2.10.1	Traço Parcial	p. 35
2.10.2	Canais Quânticos	p. 36
3	Requisitos Gerais para um Novo Simulador	p. 38
3.1	Recursos do Sistema	p. 38
3.1.1	Recursos de Funcionalidade	p. 38
3.1.2	Recursos de Usabilidade	p. 39
3.1.3	Recursos de Documentação	p. 39
3.1.4	Outros Requisitos	p. 40
4	Processo de Desenvolvimento	p. 41
4.1	Princípio da Independência do Diálogo	p. 41
4.2	Processo Unificado Simplificado	p. 42
4.2.1	As Quatro Fases de um Processo	p. 43
4.2.2	Fluxos de Atividades	p. 43
4.3	MEDITE	p. 45
4.4	Análise da Tarefa	p. 46
4.4.1	Aquisição de Dados sobre a Tarefa e sobre o Usuário	p. 46

4.4.2	Descrição da Tarefa Utilizando o TAOS	p. 46
4.5	Desenvolvimento do Módulo Funcional	p. 49
4.5.1	Fase de Planejamento	p. 49
4.5.2	Fase de Elaboração	p. 50
4.5.2.1	Decisões de Projeto	p. 50
4.5.3	Fase de Construção	p. 51
4.5.4	Fase de Transição	p. 51
4.6	Desenvolvimento do Módulo de Interação	p. 52
4.6.1	Identificação de Janelas e Objetos Visuais	p. 52
4.6.2	Implementação	p. 52
4.7	Integração	p. 52
4.7.1	Testes	p. 52
5	Análise de resultados	p. 53
5.1	Recursos Planejados <i>vs.</i> Implementados	p. 53
5.1.1	Recursos de Funcionalidade	p. 53
5.1.2	Recursos de Usabilidade	p. 55
5.1.3	Recursos de Documentação	p. 57
5.2	Testes de Correção e Desempenho	p. 57
5.2.1	Somador de Dois Bits	p. 58
5.2.2	Transformada de Fourier Quântica (QFT)	p. 59
5.2.3	Circuito para Calcular o Autovalor	p. 62
5.2.4	Testes com a Porta de Medição	p. 64
5.3	Resultados dos Testes	p. 66
6	Conclusão	p. 67
6.1	Objetivos Alcançados	p. 67
6.2	Contribuições	p. 68
6.3	Perspectivas Futuras	p. 68
	Referências	p. 70
	Apêndice A – Guia de como Usar o Simulador Zeno	p. 73

A.1	Menus	p. 74
A.2	Portas	p. 75
A.3	Barra de Tarefas	p. 76
A.4	Circuitos	p. 76

Lista de Tabelas

1	Atividades de cada fluxo de atividades ^[13]	p. 44
2	Atividades modificadas	p. 44
3	Descritor — tarefa ‘gerenciar circuitos’	p. 48
4	Descritor — pós-situação de ‘gerenciar circuitos’	p. 48
5	Descritor — ação ‘definir número de colunas’	p. 48
6	Descritor — requisito de ‘definir número de colunas’	p. 48
7	Descritor — pós-situação de ‘definir número de colunas’	p. 49
8	Tempo de execução do circuito somador de dois bits	p. 58
9	Resultados de simulação do circuito somador de dois bits	p. 59
10	Tempo de execução do circuito QFT para 3 qubits	p. 60
11	Tempo de execução do circuito QFT para 6 qubits	p. 60
12	Tempo de execução do circuito para calcular o autovalor de um operador quântico	p. 63
13	Resultados de simulação do circuito QFT de 3 qubits com leitura de todos os qubits no final	p. 66

Lista de Figuras

1	Portas NOT e AND, respectivamente	p. 23
2	Circuito que computa a função adição módulo 2.	p. 23
3	Porta CNOT	p. 26
4	Porta U-controlada	p. 26
5	Porta <i>Toffoli</i>	p. 27
6	Circuito que põe registrador no estado $\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} x\rangle$	p. 28
7	Computando valor de f para todos os valores de x	p. 29
8	Decomposição de porta U controlada	p. 30
9	Decomposição de porta U controlada por 2 qubits	p. 30
10	Construção de operador unitário de dois níveis U	p. 31
11	Canal quântico unitário	p. 36
12	Canal quântico linear	p. 36
13	Software interativo	p. 42
14	Tarefa “Utilizar uma ferramenta de software para projetar e simular circuitos quânticos”	p. 47
15	Tarefa “Configurar ferramenta”	p. 47
16	Arquitetura	p. 49
17	Modelo conceitual do módulo funcional	p. 50
18	Circuito representado no simulador Zeno	p. 55
19	Visualização do estado no simulador Zeno como ket	p. 56
20	Visualização do estado no simulador Zeno como matriz-densidade	p. 56
21	Visualização do estado no simulador Zeno como histograma	p. 56
22	Montagem do somador de números de 2 bits no simulador Zeno	p. 58
23	Montagem do somador de números de 2 bits no simulador jaQuzzi.	p. 59
24	Circuito da QFT para 3 qubits no simulador Zeno.	p. 60
25	Circuito da QFT para 3 qubits no simulador jaQuzzi.	p. 61
26	Resultado da simulação para a entrada $ 000\rangle$ no circuito QFT para 3 qubits no simulador Zeno.	p. 61

27	Resultado da simulação para a entrada $ 000\rangle$ no circuito QFT para 3 qubits no simulador jaQuzzi.	p. 62
28	Resultado do mesmo circuito no simulador Zeno com estados mistos.	p. 62
29	Circuito para calcular o autovalor de um operador unitário no simulador Zeno.	p. 63
30	Circuito para calcular o autovalor de um operador unitário no simulador jaQuzzi.	p. 63
31	Resultado da simulação do circuito para cálculo de autovalor no simulador Zeno.	p. 64
32	Resultado da simulação do circuito para cálculo de autovalor no simulador jaQuzzi.	p. 64
33	Circuito da QFT para 3 qubits com portas de medição nos simuladores Zeno (esquerda) e jaQuzzi.	p. 65
34	Circuito da QFT para 3 qubits com portas de medição nos simuladores Zeno (esquerda) e jaQuzzi.	p. 65
35	As partes da janela principal do simulador Zeno	p. 73

1 *Introdução*

Este trabalho apresenta o processo de concepção e implementação de um simulador universal de circuitos quânticos denominado Zeno. Um simulador é uma ferramenta importante para a análise de comportamento e projeto de sistemas, principalmente quando há sérias dificuldades ou mesmo impossibilidade de se construir o sistema fisicamente, como é o caso dos computadores quânticos atualmente.

Neste capítulo apresentamos a importância e o cenário atual dos simuladores quânticos, identificamos as deficiências dos principais simuladores existentes, apresentamos os objetivos do trabalho, a metodologia utilizada para a construção de um novo simulador e a estrutura da dissertação.

1.1 *Computação Quântica*

O surgimento do computador causou grande impacto no desenvolvimento da sociedade. Os primeiros computadores foram construídos utilizando-se válvulas e relés como seus componentes principais, e realizavam cálculos várias vezes mais rápido que qualquer ser humano. Esses computadores porém ocupavam muito espaço físico e se provaram muito lentos para as necessidades de cálculos que surgiriam. O advento dos transistores e dos circuitos integrados (CI's) possibilitou uma diminuição considerável no tamanho dos circuitos e no seu gasto de energia, e um aumento na sua velocidade de processamento.

O número de transistores por CI dobrou a cada ano entre 1958 e 1979, e desde então até agora vem aumentando a uma taxa de aproximadamente 2 vezes a cada 18 meses. Isso significa que as dimensões de um transistor estarão brevemente passando da faixa de μm (micrômetros) para a faixa de nm (nanômetros), ou seja, estaremos entrando na *nano-eletrônica*, na escala atômica. Segundo previsões da associação da indústria de semicondutores americana (NSIA) para a tecnologia CMOS, o tamanho dos transistores passará da faixa de $0,35\mu m$ ($350nm$) atuais para a faixa de $0,07\mu m$ ($70nm$) até 2010. No entanto, a tecnologia CMOS pode progredir até o limite de $20nm$ sem sacrifício funcional. Abaixo desse limite (chamado limite de De Broglie), os transistores CMOS deixam de funcionar como esperado: os elétrons deixam de agir como partículas e passam a se comportar como ondas (i.e., segundo o princípio da dualidade das partículas subatômicas, formulado por De Broglie^[15]), obedecendo essencialmente às leis da Mecânica Quântica. Ao invés do efeito de chaveamento, se tornam dominantes efeitos como interferência, emaranhamento e superposição.

Como fica evidente, a tecnologia dominante está caminhando para a saturação. Dentre as alternativas que estão sendo consideradas para substituir a atual tecnologia, a que

aparece como mais promissora e inovadora é a que propõe o uso dos efeitos quânticos para se obter dispositivos e máquinas quânticas — um sistema quântico opera em muitos estados clássicos simultaneamente; seu estado quântico pode ser, em qualquer instante, uma superposição de estados clássicos. Esse paralelismo inerente aos sistemas quânticos leva à possibilidade de se conceber e construir máquinas com poder de processamento exponencialmente maior do que a atual tecnologia permite, i.e., máquinas capazes de resolver problemas que são considerados “intratáveis” por qualquer máquina atual. A disciplina que estuda o uso dos efeitos quânticos para realizar computação, chamada *Computação Quântica*, está no seu começo, impulsionada pela possibilidade extraordinária de um salto exponencial da capacidade de processamento.

1.1.1 O Computador Quântico

A abordagem quântica introduz mais do que a evolução do processo de miniaturização, ela torna possível um paralelismo computacional impossível através de outras técnicas e que nos leva às portas de uma revolução no mecanismo computacional. Como dizem Ekert et al.^[7]: “Então, se os computadores se tornarem menores no futuro, a nova tecnologia quântica deve substituir ou complementar a que nós temos agora. A questão é que a tecnologia quântica pode oferecer muito mais que a possibilidade de abarrotar o silício de bits aumentando assim a velocidade de processamento. Ela pode dar origem a uma forma de computação completamente nova com algoritmos baseados nos princípios quânticos.”

A Computação Quântica vai além, portanto, de ser uma tecnologia mais rápida e miniaturizada para a construção de computadores. Um computador quântico é uma máquina que usa unicamente efeitos quânticos, como a *interferência* e a *superposição*, para executar tipos totalmente novos de computação que seriam impossíveis, em princípio, de serem realizados com eficiência em qualquer computador clássico.

O conceito de computador quântico foi lançado por Feynman em um artigo de 1982^[23], no qual ele afirma a incapacidade de computadores clássicos simularem sistemas quânticos eficientemente, e lançou a idéia de um *computador quântico* como um simulador quântico universal. A idéia foi levada adiante por Deutsch^[17], que propôs um modelo para a Computação Quântica universal, onde mostrou que basta usar um sistema quântico de dois estados e definir um pequeno conjunto de operações simples para reproduzir a evolução unitária no tempo de qualquer sistema quântico. Este formalismo foi posteriormente refinado por Feynman^[24] e Deutsch^[18], o que levou a uma descrição da Computação Quântica muito similar àquela da computação clássica através de portas lógicas booleanas. Deutsch mostrou que uma máquina quântica universal é capaz de simular qualquer máquina de Turing.

Entretanto, até então, a Computação Quântica era apenas uma curiosidade acadêmica. Somente em 1994, quando Peter Shor publicou seu artigo^[40] com um algoritmo em tempo polinomial para fatoração de números primos — um problema que ainda hoje não tem solução clássica eficiente conhecida — é que se percebeu que a Computação Quântica realmente poderia ter alguma utilidade prática, i.e., poderia resolver problemas reais de forma mais eficiente do que a computação clássica. Outros algoritmos que surgiram desde então são o algoritmo de Grover^[29], que oferece um aumento quadrático em velocidade para buscas em bases de dados não-ordenadas, e o algoritmo de Shor^[41] para correção de

erros.

Enquanto a parte teórica da Computação Quântica vem se desenvolvendo com rapidez, a parte experimental ainda enfrenta grandes dificuldades, devido a enormes obstáculos ainda a transpor para se construir efetivamente uma máquina quântica. Diversas tecnologias vêm sendo propostas, entre elas, íons aprisionados^[12, 35, 39], eletrodinâmica quântica de cavidades (QED)^[44] e ressonância magnética nuclear (NMR), para implementar registradores e portas quânticas. A tecnologia NMR foi usada com sucesso para desenvolver sistemas de 2 e 3 *qubits*. Esses computadores quânticos minúsculos foram usados para implementar o problema de Deutsch^[31] e o algoritmo de Grover^[27, 32] e mostrar que esses algoritmos quânticos podem ser executados em hardware quântico.

Embora essas implementações básicas já tenham sido realizadas, ainda há problemas sérios de escalabilidade, descoerência e controlabilidade a serem resolvidos. Além disso, ainda não há hardware quântico fora dos laboratórios de pesquisa, e mesmo que houvesse, os sistemas construídos não seriam suficientes para uma exploração detalhada de alguns dos algoritmos quânticos que foram propostos. Até que se resolvam todos esses problemas, a simulação de computadores quânticos (em computadores clássicos) pode ser uma boa solução temporária, embora que limitada, como será visto a seguir.

1.2 Simuladores

A simulação digital é uma técnica largamente utilizada para se prever o comportamento de sistemas, sejam estes reais ou hipotéticos. Como, muitas vezes, não é possível se construir o sistema fisicamente (por este ser de difícil manuseio ou exigir recursos não disponíveis), ou mesmo sequer o sistema existe, faz-se uso de simulação para se prever o seu comportamento. Além disso, a simulação digital permite a repetição do experimento várias vezes, com valores diferentes e testando situações-limites sem que nenhum equipamento seja danificado ou precise de manutenção (ajustes, reposição de peças, etc.).

Não menos importante é a utilidade que a simulação tem no aprendizado da teoria sobre o sistema sendo simulado. Os simuladores, por permitirem repetição dos testes e variedade de valores, permitem uma melhor compreensão do funcionamento do sistema.

Um bom exemplo da utilidade da simulação é o planejamento de circuitos digitais^[2]. Apesar de ser barato e fácil de se construir um circuito digital hoje em dia, simuladores de circuitos digitais foram e ainda são largamente usados para o planejamento/desenvolvimento de circuitos digitais.

1.2.1 Simuladores Quânticos

Devido aos problemas com a implementação de um hardware quântico citados na seção anterior, torna-se imperativo o uso de simuladores na Computação Quântica para: explorar e desenvolver algoritmos quânticos, investigar e caracterizar erros intrínsecos (descoerência) e operacionais, investigar códigos de correção de erros, e, adicionalmente, para implementar e simular os algoritmos quânticos conhecidos, estimulando o aprendizado e, conseqüentemente, o desenvolvimento de novos algoritmos. Uma outra área im-

portante de aplicação de simuladores quânticos, ainda não explorada, seria na Informação Quântica, para a investigação, desenvolvimento e experimentação de modelos de canais quânticos.

Outra vantagem do uso de um simulador quântico é a possibilidade de se visualizar o estado quântico durante a computação, ou seja, nos passos intermediários da computação. Esta observação é proibida pelas leis da Mecânica Quântica (ocorreria descoerência na observação) e portanto, impossível de ser feita em uma máquina real. A observação desses estados intermediários permite o estudo dos efeitos causados pelos erros.

1.2.1.1 Problemas da Simulação

Como Feynman observou, a única maneira de se simular eficientemente um sistema quântico é usando outro sistema quântico. Um simulador de computadores quânticos é uma tentativa de se modelar um sistema quântico em um sistema clássico, e isso é impossível de ser feito eficientemente. O simulador quântico executado em uma máquina clássica terá que manter registro de um número exponencial (com relação ao tamanho da entrada) de forma a simular o sistema quântico com precisão. “Para simular um vetor de estados em um espaço de Hilbert 2^n -dimensional, um computador clássico precisará manipular vetores contendo na ordem de 2^n números complexos, enquanto que um computador quântico requer apenas n qubits, isto o torna mais eficiente em espaço de armazenamento.”^[42] Isso implica que a simulação de computadores quânticos requer recursos extraordinários não apenas de espaço, mas também de tempo, que cresce na mesma razão do espaço necessário para armazenar os dados.

1.2.2 Estado Atual

Até nossos dias, o mais abrangente e completo trabalho sobre simuladores de computadores quânticos é o apresentado por Julia Wallace^[46] que discute os simuladores de computadores quânticos e sua importância no estudo, na pesquisa e no desenvolvimento da Computação Quântica. Os diversos tipos e exemplares de simuladores até então existentes são descritos e comparados segundo seu desempenho na execução do algoritmo de Shor. A maioria dos simuladores quânticos lá descritos são simuladores específicos para apenas um ou outro algoritmo quântico (tal como o algoritmo de fatoração de Shor ou o algoritmo de busca de Grover). No entanto, em levantamento mais recente realizado pelo autor^[10], foram identificados novos simuladores mais elaborados que usam o modelo de circuitos quânticos de Deutsch e permitem que o usuário (projetista) monte um algoritmo quântico qualquer a partir das portas quânticas básicas disponíveis. Esses simuladores são ditos *universais* no sentido que disponibilizam portas quânticas universais suficientes para a implementação de qualquer algoritmo quântico. Nesta categoria estão o simulador QComputer, de D. Menscher^[34], o Quantum Computer Simulator da Senko Corp.^[1], o simulador jaQuzzi, desenvolvido por Felix Schürmann^[38] e o QCSim, desenvolvido por Paul Black e M. Cass^[11]. A tendência é que os simuladores evoluam nesta direção, ou seja, simuladores de circuitos quânticos que permitam que o usuário monte qualquer circuito desejado a partir de conjuntos universais de portas quânticas.

1.2.3 Deficiências

Apesar de representarem uma contribuição considerável no apoio à modelagem e simulação de algoritmos quânticos, esses novos simuladores citados acima ainda apresentam algumas deficiências, que podem ser classificadas em três categorias:

1. Funcionalidade
2. Usabilidade
3. Documentação do desenvolvimento

1.2.3.1 Funcionalidade

Apesar de já existirem simuladores universais, esses simuladores, em geral, ainda apresentam deficiências com respeito a funcionalidades desejáveis tanto do ponto de vista da Computação Quântica (modelagem simulação de algoritmos) quanto da Informação Quântica (modelagem e simulação de canais quânticos). Entre essas deficiências, pode-se citar: o uso exclusivo de estados puros (a maioria) ou de estados mistos (QCSim); a impossibilidade de medição em bases diferentes da base computacional; a ausência da operação de traço parcial (à exceção do QCSim); a não geração de erros operacionais e de descoerência no circuito (à exceção do jaQuzzi); entre outras; a utilização apenas de portas pré-definidas (QCSim); a impossibilidade de criação de bibliotecas de portas personalizadas.

1.2.3.2 Usabilidade

Uma boa interface com o usuário é essencial para uma boa e fácil utilização de um sistema computacional pois ela proporciona facilidade de aprendizagem, de uso e de entendimento do sistema.

Dos simuladores avaliados, o QCSim e o QComputer não apresentam uma interface segundo os padrões atuais de uma boa interface (interface gráfica). Suas interfaces são textuais, o que dificulta a visualização e edição de um circuito. O jaQuzzi apresenta interface gráfica, mas sem recursos de manipulação direta, a edição é feita totalmente por meio de menus e formulários. O Senko, por sua vez, apresenta uma interface gráfica com manipulação direta. A manipulação direta permite que se manipule os elementos gráficos sem que se tenha que recorrer a menus, formulários, etc., ou seja, usando apenas o *mouse*.

1.2.3.3 Documentação do Desenvolvimento

A documentação de desenvolvimento de um sistema computacional é importante para a compreensão de sua arquitetura e para que projetos como o presente trabalho possam aproveitar as idéias já pensadas por outros, economizando tempo de desenvolvimento.

Poucos dos simuladores listados acima apresentam uma documentação sólida de como foram desenvolvidos. O Senko não apresenta por ser um software comercial. O jaQuzzi

apresenta apenas o diagrama de classes e uma descrição sucinta dos seus módulos principais. O QCSim e o QComputer não apresentam documentação de projeto, mas apenas o código fonte do simulador.

1.3 Objetivo

Um novo simulador quântico deve implementar as principais funcionalidades dos simuladores já existentes, e superar as suas deficiências com relação à funcionalidade, à usabilidade e à documentação.

Assim, o presente trabalho tem como objetivo principal a concepção e o desenvolvimento de um simulador universal de circuitos quânticos. Este simulador deve:

1. oferecer recursos funcionais que permitam modelar e simular algoritmos e canais quânticos utilizando o modelo de circuitos quânticos;
2. oferecer recursos avançados de *manipulação direta* que permitam a edição e a montagem de circuitos de maneira fácil e intuitiva;
3. disponibilizar uma documentação apropriada tanto do processo de desenvolvimento quanto da ferramenta a ser construída.

1.4 Relevância

Um simulador de circuitos quânticos é um elemento valioso no ensino e na aprendizagem da Computação Quântica e da Informação Quântica, devido à facilidade de uso, a precisão e a velocidade de resposta que a ferramenta proporciona com relação a cálculos manuais, que é, em geral, a única alternativa disponível. Alunos e professores poderão validar os resultados dos cálculos, além de poder usar a ferramenta em exercícios para testar várias soluções.

Também na pesquisa da Computação Quântica e da Informação Quântica, o uso de um simulador é de grande importância, pois permite a validação de algoritmos e modelos, e o estudo dos erros e dos efeitos causados pelas diversas propriedades quânticas (interferência, superposição, etc).

Para o grupo local de estudos em Computação e Informação Quântica da UFCG, um simulador de circuitos quânticos será valioso devido às razões apresentadas acima, assim como ao estímulo à melhoria e ao desenvolvimento de disciplinas sobre o assunto, além de servir de veículo de divulgação do grupo fora da UFCG.

1.5 Metodologia de Trabalho

Este trabalho foi desenvolvido utilizando a seguinte metodologia:

1. realização de estudos sobre os fundamentos da Mecânica Quântica e da sua aplicação na Computação e na Informação Quânticas;
2. realização de estudos e levantamento das principais ferramentas de simulação de circuitos quânticos existentes atualmente;
3. definição dos requisitos gerais, ou recursos que a ferramenta deve oferecer;
4. seleção e utilização de métodos e técnicas de engenharia de software e de interface homem-máquina para levantamento de requisitos funcionais e não-funcionais;
5. realização do projeto, da implementação e de testes da ferramenta desenvolvida;
6. elaboração de artigos técnico-científicos para a divulgação do trabalho e aspectos deste.

1.5.1 Metodologia de Desenvolvimento

Para a parte funcional, foi usado um processo simplificado baseado no Processo Unificado^[33], e para a parte da interação com o usuário foi utilizada a metodologia MEDITE^[30] de concepção e desenvolvimento de interface baseada na análise e modelagem da tarefa.

1.5.2 Análise de Resultados

Os resultados do trabalho serão avaliados de duas formas: (i) a conformidade da ferramenta implementada com os requisitos planejados e (ii) o desempenho e a correção dos resultados da simulação da ferramenta desenvolvida em comparação com o desempenho e os resultados obtidos no simulador jaQuzzi.

No que se refere à conformidade com os requisitos planejados, serão comparadas as funcionalidades implementadas com as que foram propostas no início do desenvolvimento.

Quanto ao desempenho, serão comparados os tempos de simulação e os resultados obtidos nas simulações de alguns circuitos em ao menos dois simuladores (um deles sendo o simulador sendo desenvolvido neste trabalho).

1.6 Estrutura da Dissertação

Após esta introdução, é apresentado um capítulo (Capítulo 2) sobre os fundamentos teóricos da Computação Quântica, em especial sobre o modelo de circuitos quânticos.

O Capítulo 3 aborda a definição dos requisitos gerais, ou recursos que a ferramenta deve oferecer para seus usuários de forma a atender aos objetivos definidos na Seção 1.3.

O Capítulo 4 apresenta uma descrição do processo de desenvolvimento da ferramenta, e dos artefatos produzidos.

O Capítulo 5 apresenta uma análise dos resultados obtidos.

O Capítulo 6 apresenta uma discussão sobre os resultados e perspectivas futuras.

2 *Fundamentação Teórica – Computação Quântica*

A utilização de fenômenos quânticos para a representação e processamento de informação se apresenta hoje como uma das alternativas mais promissoras para futuras gerações de dispositivos computacionais. As vantagens teóricas advindas dessa utilização vêm atraindo atenção crescente do mundo científico e tecnológico/industrial.

Em 1985, Deutsch propôs um algoritmo, utilizando apenas operações quânticas, capaz de resolver um determinado problema matemático impossível de ser resolvido com operações ou métodos clássicos. O problema (chamado o *Problema de Deutsch*) consiste em determinar se uma função booleana $f : \{0, 1\} \rightarrow \{0, 1\}$ é constante ou balanceada¹ avaliando-a apenas uma vez^[17]. No entanto, esse algoritmo só alcançou notoriedade em 1989, quando Deutsch introduziu a noção de portas lógicas quânticas que poderiam ser conectadas umas às outras formando um circuito ou malha quântica^[18]. O algoritmo de Deutsch reescrito na nova linguagem teve a partir de então uma ampla repercussão, pois a linguagem dos qubits (análogo quântico ao *bit* clássico) e portas lógicas quânticas (análogo quântico às portas lógicas booleanas) era similar à linguagem de circuitos lógicos/digitais convencionais e poderia ser mais facilmente entendida por engenheiros eletricitistas e cientistas da Computação. Daí em diante, outros algoritmos quânticos foram desenvolvidos e difundidos utilizando-se o modelo de circuitos quânticos de Deutsch associado à linguagem matemática.

Neste capítulo, vamos apresentar as noções básicas e os conceitos fundamentais da Computação Quântica^[6, 21, 26] e da Informação Quântica^[28, 36, 37], através da apresentação do modelo de circuitos quânticos, fazendo sempre o contra-ponto com as noções e conceitos básicos da Computação Clássica já conhecidos.

2.1 Representação da Informação

Na Computação Quântica, ao invés da noção clássica de bit, temos a noção de *qubit* (*quantum bit*). Diferentemente de um bit de informação, que pode estar em (ou representar) apenas dois estados distintos, 0 ou 1, um qubit, além dos estados $|0\rangle$ e $|1\rangle$, pode

¹Uma função booleana é balanceada se metade de seu domínio leva a 0 e a outra metade leva a 1, e é constante se todo o domínio leva a 0 ou a 1.

estar em qualquer *superposição* de estados da forma:

$$|\psi\rangle = c_0|0\rangle + c_1|1\rangle \quad (2.1)$$

onde c_0 e c_1 são coeficientes complexos e tais que $|c_0|^2 + |c_1|^2 = 1$. As notações $|0\rangle$ e $|1\rangle$ representam, respectivamente, os vetores:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (2.2)$$

e a notação $|\psi\rangle$ representa o vetor:

$$|\psi\rangle = \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} \quad (2.3)$$

Um qubit pode ser realizado de diversas formas: por meio de uma partícula de *spin* $\pm 1/2$, onde os estados $|0\rangle$ e $|1\rangle$ corresponderiam, respectivamente, aos estados *spin-down* e *spin-up*; ou ainda por meio de caminhos seguidos pelo feixe (ou pelo fóton) ao passar por um interferômetro. Embora um qubit possa ser preparado em um número infinito de estados quânticos (c_0 e c_1 são complexos), quando medido, só podemos obter um bit de informação. A medição de um qubit inicialmente em um estado $c_0|0\rangle + c_1|1\rangle$ dará como resultado o bit 0, com probabilidade $|c_0|^2$, ou o bit 1, com probabilidade $|c_1|^2$, ou seja, num aparato de medição, os resultados 0 e 1 seriam registrados com probabilidades $|c_0|^2$ e $|c_1|^2$ nos respectivos detectores. Depois de uma medição, ao contrário de um bit clássico que mantém o seu estado, o estado de um qubit é alterado para o estado $|0\rangle$ ou $|1\rangle$, de acordo com as respectivas probabilidades. Ou seja, medir um qubit altera, em geral, seu estado.

Em um computador clássico, bits são agrupados em conjuntos chamados *registradores*. Então, se um bit pode armazenar um dos dois números 0 ou 1, um registrador de n bits pode armazenar 2^n números diferentes ($\{0, 1, \dots, 2^n - 1\}$), um por vez.

Em um computador quântico, um registrador de n qubits, por sua vez, pode armazenar 2^n números diferentes ao mesmo tempo. Por exemplo, com 2 qubits podemos representar o estado:

$$\begin{aligned} |\psi\rangle &= |\psi_1\rangle \otimes |\psi_2\rangle \\ &= (c_0^{(1)}|0\rangle + c_1^{(1)}|1\rangle) \otimes (c_0^{(2)}|0\rangle + c_1^{(2)}|1\rangle) \\ &= c_0^{(1)}c_0^{(2)}|0\rangle \otimes |0\rangle + c_0^{(1)}c_1^{(2)}|0\rangle \otimes |1\rangle + \\ &\quad c_1^{(1)}c_0^{(2)}|1\rangle \otimes |0\rangle + c_1^{(1)}c_1^{(2)}|1\rangle \otimes |1\rangle \\ &\equiv c_0^{(1)}c_0^{(2)}|0\rangle|0\rangle + c_0^{(1)}c_1^{(2)}|0\rangle|1\rangle + \\ &\quad c_1^{(1)}c_0^{(2)}|1\rangle|0\rangle + c_1^{(1)}c_1^{(2)}|1\rangle|1\rangle \end{aligned} \quad (2.4)$$

$$\equiv c_0|0\rangle + c_1|1\rangle + c_2|2\rangle + c_3|3\rangle \quad (2.5)$$

$$= \sum_{x=0}^3 c_x|x\rangle \quad (2.6)$$

onde $c_i^{(j)}$ indica o coeficiente c_i do qubit j , e a Equação (2.5) é obtida da Equação (2.4) pela mudança da notação binária pela decimal (por exemplo, $|1\rangle|1\rangle$ por $|3\rangle$).

2.2 Processamento da Informação

Em um computador clássico, o processamento da informação é realizado por dispositivos chamados de *circuitos lógicos*, que são agrupamentos de dispositivos mais simples conhecidos por *portas lógicas*. Uma porta lógica opera sobre o estado dos bits da entrada, obtendo um outro estado na saída correspondendo à sua “tabela da verdade”. Por exemplo, a porta lógica NOT inverte na saída o estado do bit da entrada ($\bar{a} = 1 - a$): se o bit da entrada (a) estiver no estado 0, então o bit 1 é gerado na saída (\bar{a}), e vice-versa. A porta AND tem dois bits na entrada (a e b) e gera uma saída, c , dada por $c = a \cdot b$. Com as portas lógicas NOT e AND é possível construir circuitos lógicos capazes de computar qualquer função $f : \{0, \dots, 2^m - 1\} \rightarrow \{0, \dots, 2^n - 1\}$ teoricamente computável^[36]. Representando as portas NOT e AND como na Figura 1

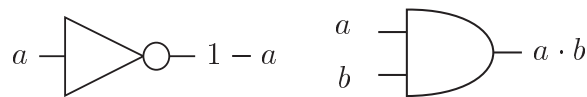


Figura 1: Portas NOT e AND, respectivamente

e conectando as portas umas às outras por fios, podemos representar os circuitos lógicos através de um diagrama. Por exemplo, o diagrama da Figura 2 representa um circuito lógico que realiza (ou computa) a função $f(a, b) = a \oplus b$, onde \oplus é a operação adição módulo 2 (**mod 2**)².

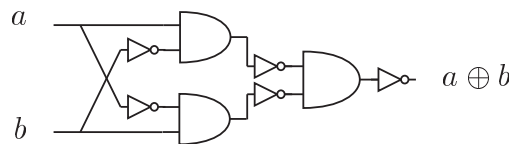


Figura 2: Circuito que computa a função adição módulo 2.

2.3 Portas de Um Qubit

Em um computador quântico o processo é similar, ou seja, podemos construir (teoricamente) circuitos quânticos, que são agrupamentos de dispositivos mais simples chamados *portas quânticas*, que realizam *operações unitárias* sobre um registrador quântico. Uma porta quântica simples aplica uma operação unitária U (definida por uma matriz unitária 2×2)³ sobre um qubit no estado $|\psi\rangle$ fazendo-o evoluir para o estado $U|\psi\rangle$. Por exemplo, a porta quântica X , descrita pela matriz:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad (2.7)$$

²Na aritmética modular, a operação módulo 2 é definida como o resto r da divisão euclidiana de um número m por 2: $m = k \cdot 2 + r$, onde k é um inteiro. Então, dizemos que $r = m \bmod 2$.

³Uma matriz unitária U é toda aquela que satisfaz à seguinte propriedade: $UU^\dagger = I = U^\dagger U$, onde \dagger é a operação transposta conjugada e I é a matriz identidade.

corresponde à porta lógica NOT:

$$X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} |0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle \quad (2.8)$$

$$X|1\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} |1\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle \quad (2.9)$$

A matriz que descreve a porta X pertence a um grupo especial de matrizes na Computação Quântica, as *matrizes de Pauli*. As matrizes de Pauli são quatro:

$$\begin{aligned} I = \sigma_0 &\equiv \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & X = \sigma_1 = \sigma_x &\equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \\ Y = \sigma_2 = \sigma_y &\equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} & Z = \sigma_3 = \sigma_z &\equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \end{aligned} \quad (2.10)$$

A importância das matrizes de Pauli será vista mais à frente.

Examinando-se as matrizes de Pauli, percebe-se que nem todas as portas possuem correspondentes clássicas. Outros exemplos de portas quânticas sem equivalentes clássicas são:

$$\begin{aligned} H &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} & S &= \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \\ T &= \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix} \end{aligned} \quad (2.11)$$

A porta H , chamada de *Hadamard*, tem o seguinte efeito sobre os estados da base computacional:

$$\begin{aligned} H|0\rangle &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \\ H|1\rangle &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}}, \end{aligned} \quad (2.12)$$

ou seja, leva um estado da base em uma superposição de estados e, se aplicada novamente, leva a superposição ao estado original.

2.4 Rotações

As *matrizes de Pauli* ($\sigma_x, \sigma_y, \sigma_z$) mais a matriz identidade (I) formam uma base para o conjunto dos operadores (matrizes) unitários de dimensão 2, ou seja, qualquer operador unitário U (2×2) pode ser expresso através de uma decomposição linear nessa base, a menos de uma fase global ($e^{i\alpha}$):

$$U = u_o I + i\vec{v} \cdot \vec{\sigma} \quad (2.13)$$

onde u_o é um escalar (real), $\vec{v} = (v_x, v_y, v_z)$, um vetor real, e o vetor $\vec{\sigma} \equiv (\sigma_x, \sigma_y, \sigma_z)$. Observe que:

$$\vec{v} \cdot \vec{\sigma} = v_x \sigma_x + v_y \sigma_y + v_z \sigma_z \quad (2.14)$$

é o produto escalar entre os vetores \vec{v} e $\vec{\sigma}$.

U pode ser reescrito como:

$$U = \cos(\theta/2)I + i \sin(\theta/2)(\vec{n} \cdot \vec{\sigma}) = e^{i\theta\vec{n}\cdot\vec{\sigma}}, \quad (2.15)$$

onde \vec{n} é um vetor real unitário paralelo a \vec{v} , e θ , o ângulo de \vec{n} com o eixo polar.

Então, de 2.13 e 2.15, podemos deduzir que qualquer operador unitário de dimensão 2 pode ser interpretado como uma rotação de um vetor de θ radianos em torno do eixo \vec{n} .

A partir da Equação 2.15, podemos destacar três casos especiais de rotação, que são as rotações em torno dos eixos x , y e z , chamadas de *operadores de rotação*, e dadas pelas seguintes equações:

$$R_x(\theta) \equiv e^{-i\theta X/2} = \cos \frac{\theta}{2} I - i \sin \frac{\theta}{2} X = \begin{bmatrix} \cos \frac{\theta}{2} & -i \sin \frac{\theta}{2} \\ -i \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{bmatrix} \quad (2.16)$$

$$R_y(\theta) \equiv e^{-i\theta Y/2} = \cos \frac{\theta}{2} I - i \sin \frac{\theta}{2} Y = \begin{bmatrix} \cos \frac{\theta}{2} & -\sin \frac{\theta}{2} \\ \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{bmatrix} \quad (2.17)$$

$$R_z(\theta) \equiv e^{-i\theta Z/2} = \cos \frac{\theta}{2} I - i \sin \frac{\theta}{2} Z = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix} \quad (2.18)$$

2.4.1 Decomposição em Rotações

Utilizando os operadores de rotação definidos acima, podemos mostrar que qualquer operador unitário U de dimensão 2 pode ser decomposto em uma seqüência de operadores de rotação, seguida de um deslocamento de fase:

$$U = e^{i\alpha} R_z(\beta) R_y(\gamma) R_z(\delta) \quad (2.19)$$

onde α , β , γ e δ são números reais.

Igualmente, também é possível fazer uma decomposição nos eixos x e y :

$$U = e^{i\alpha} R_x(\beta) R_y(\gamma) R_x(\delta) \quad (2.20)$$

2.5 Portas Controladas de Dois Qubits

As portas X e H são portas quânticas simples que aplicam operações quânticas sobre um único qubit. Apesar do conjunto de portas quânticas simples ser infinito (o número de matrizes unitárias 2×2 é infinito), esse conjunto não é *universal*, ou seja, não é suficiente para construir circuitos quânticos que representam operações quânticas sobre um número n qualquer de qubits (matrizes unitárias $n \times n$). Para tanto, é preciso a utilização de portas quânticas de múltiplos qubits cujo representante principal é a porta *NOT-controlada* ou, como é mais conhecida na literatura, porta *CNOT*. Esta porta aplica uma operação sobre 2 qubits a e b (qubit de *controle* e qubit *alvo*, respectivamente), conforme a Figura 3,

descrita pela matriz 4×4 mostrada abaixo:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (2.21)$$

e cuja atuação se processa da seguinte forma: se o qubit de controle a está no estado $|0\rangle$, o estado $|b\rangle$ do qubit alvo permanece inalterado. Se o qubit de controle a está no estado $|1\rangle$ o estado $|b\rangle$ do qubit alvo é alterado para $|b \oplus a\rangle$. A operação sobre o qubit alvo pode ser expressa como $X^a|B\rangle$, onde X é a porta X e o sobrescrito “ a ” é o valor do bit de controle. O diagrama da Figura 3 representa a porta *CNOT*.

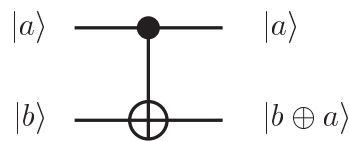


Figura 3: Porta CNOT

Podemos generalizar a porta *NOT-controlada* para uma porta *U-controlada* sobre dois qubits, onde U é uma operação unitária sobre um simples qubit, e cuja ação é descrita pelo diagrama da Figura 4.

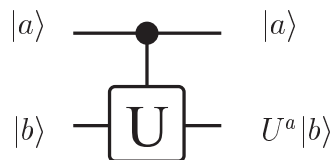


Figura 4: Porta U-controlada

2.6 Computação Reversível

No final dos anos 70, os cientistas começaram a questionar as implicações que a Física teria sobre a Computação. Inicialmente, a termodinâmica da computação clássica era o alvo principal dos esforços, com questões sobre o mínimo de energia necessária para realizar uma dada computação, ou quanto calor o apagamento de um bit dissiparia, entre outras^[47].

Os modelos de computadores resultantes dessas pesquisas serviram como experimentos teóricos ideais que revelaram algumas conexões intrigantes entre a reversibilidade lógica⁴ e a termodinâmica. Charles Bennett^[8], da IBM, mostrou que qualquer operação lógica irreversível teria que dissipar uma certa quantidade mínima de energia.

⁴Se uma operação é reversível, então pode-se obter o estado anterior à sua aplicação a partir do estado após a aplicação.

Em 1982, Ed Fredkin e Tommaso Toffoli provaram que era possível construir fisicamente uma porta lógica clássica universal e reversível^[25]. A porta apresentada por Fredkin e Toffoli levou o nome do segundo. A porta *Toffoli* é representada pela Figura 5.

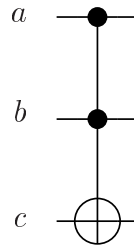


Figura 5: Porta *Toffoli*

A ação realizada pela da porta Toffoli pode ser descrita pela matriz abaixo:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (2.22)$$

A porta Toffoli pode ser usada para implementar todas as operações booleanas (AND, NAND, OR, NOT), assim como a operação FANOUT. Por exemplo, para computar a operação NAND, coloca-se os dois bits de entrada como as entradas a e b da porta Toffoli e prepara-se um bit no estado 1 associado à entrada c (bit *ancilla* ou *de trabalho*). A saída do NAND é lida na saída c do circuito. A operação FANOUT pode ser implementada preparando-se um bit de trabalho no estado 1 para a entrada a , um bit de trabalho no estado 0 na entrada c , e o bit que se quer copiar na entrada b . A saída c conterá a cópia.

Assim, um circuito clássico qualquer pode ser eficientemente simulado usando-se um circuito reversível consistindo apenas de portas Toffoli e bits de trabalho^[36].

Então, esses trabalhos mostraram que a Computação Clássica pode ser realizada de forma totalmente reversível, acarretando, portanto, um gasto mínimo de energia.

2.7 Computação Quântica

Como já foi dito na Seção 2.3, as portas quânticas implementam operadores unitários, ou seja, operadores que satisfazem à propriedade $UU^\dagger = I = U^\dagger U$. Isso implica que, se existe o operador unitário U , então existe um operador unitário U^\dagger que realiza a operação inversa. Ou seja, se U é aplicado a um estado $|\psi\rangle$, teremos um resultado $|\psi_o\rangle$ na saída. Aplicando, agora, o operador U^\dagger a $|\psi_o\rangle$, obteremos o estado $|\psi\rangle$ original.

Classicamente, um circuito para computar uma função n -ária⁵ f consistiria em um aglomerado de portas com n entradas x_1, \dots, x_n e uma saída $y = f(x_1, \dots, x_n)$. Entretanto, um circuito quântico não pode computar uma função dessa forma pois uma operação quântica é unitária e portanto reversível. Um computador quântico precisa de 2 registradores: um para guardar o estado da entrada e outro para o estado da saída, além de bits extras de trabalho. A computação de uma função seria determinada por uma operação unitária U_f que agiria sobre os dois registradores (a menos dos bits de trabalho), preservando a entrada:

$$U_f(|x\rangle|y\rangle) = |x\rangle|y \oplus f(x)\rangle \quad (2.23)$$

Podemos observar que se $y = 0$,

$$U_f(|x\rangle|0\rangle) = |x\rangle|0 \oplus f(x)\rangle = |x\rangle|f(x)\rangle \quad (2.24)$$

Suponha que se prepare um registrador quântico $|\psi\rangle$ de m qubits em uma superposição de todos os valores de entrada (2^m) utilizando m portas Hadamard, como na Figura 6. Aplicando U_f a $|\psi\rangle|0\rangle$, como na Figura 7, obtemos:

$$\begin{aligned} U_f(|\psi\rangle|0\rangle) &= U_f\left(\frac{1}{2^{m/2}} \sum_{x=0}^{2^m-1} |x\rangle|0\rangle\right) \\ &= \frac{1}{2^{m/2}} \sum_{x=0}^{2^m-1} |x\rangle|f(x)\rangle \end{aligned} \quad (2.25)$$

Ou seja, são computados todos os 2^m valores $f(0), f(1), \dots, f(2^m-1)$ ao mesmo tempo com uma única aplicação de U_f . Essa característica de poder calcular vários valores de $f(x)$ ao mesmo tempo é chamada de *paralelismo quântico*. O circuito da Figura 7 sintetiza a descrição acima. No entanto, este paralelismo por si só não se concretiza em vantagem pois ao medir a saída do circuito (o resultado da computação) obtemos apenas o valor da função em um ponto, que não reflete toda a informação contida na superposição. Uma maneira de se obter uma informação global sobre a função é fazendo uso do fenômeno da interferência, que, junto com o fenômeno da superposição, constitui a base dos atuais algoritmos quânticos.

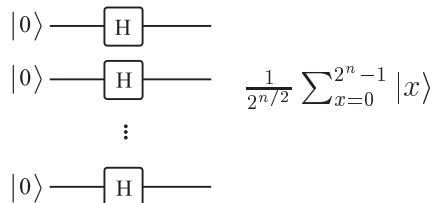


Figura 6: Circuito que põe registrador no estado $\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle$.

⁵Uma função n -ária é uma função cujo domínio tem dimensão n , ou seja, uma função com n argumentos.

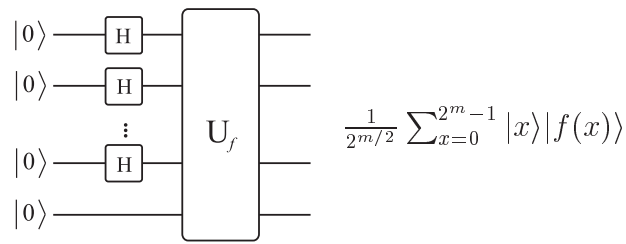


Figura 7: Computando valor de f para todos os valores de x .

2.8 Portas Quânticas Universais

Na Computação Clássica, existem conjuntos de portas lógicas a partir das quais é possível se construir um circuito para implementar qualquer função booleana. Esses conjuntos são chamados de *conjuntos de portas universais*. Por exemplo, o conjunto formado pelas portas *AND* e *NOT* mais a operação *fanout* (cópia) é universal. Na Computação Quântica, existem também conjuntos de portas universais, porém, nem sempre é possível se implementar a função desejada com exatidão. Às vezes é possível apenas obter uma aproximação da função. Entretanto, o grau de aproximação pode ser controlado arbitrariamente^[6, 36].

A seguir, serão mostrados alguns conjuntos de portas quânticas universais, a partir dos quais se pode decompor (aproximada ou exatamente) qualquer operação unitária de qualquer dimensão em uma seqüência de portas definidas por esses conjuntos.

2.8.1 Universalidade de Hadamard e as Portas de Fase

A porta Hadamard e as portas de fase ϕ definidas abaixo:

$$\phi = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix} \quad (2.26)$$

podem ser combinadas para implementar exatamente qualquer operação unitária sobre um qubit^[21]. Pode-se ver que as portas de fase implementam rotações sobre o eixo z (v. Equação 2.18).

2.8.2 Decomposição de $C(U)$

Existe uma forma de se decompor qualquer operação unitária U de um qubit controlada por outro qubit ($C(U)$) em uma seqüência de portas de um qubit e de portas *CNOT*. Nielsen e Chuang^[36] mostraram que, para um operador unitário qualquer U sobre um único qubit, existem operadores unitários A , B e C sobre um único qubit tais que $ABC = I$ e $e^{i\alpha}AXBXC = U$, onde α é um fator de fase global.

Fazendo-se

$$A = R_z(\beta)R_y(\gamma/2) \quad (2.27)$$

$$B = R_y(-\gamma/2)R_z(-(\delta + \beta)/2) \quad (2.28)$$

$$C = R_z((\delta - \beta)/2) \quad (2.29)$$

onde β , γ e δ são ângulos de rotação em torno do eixos z , y e z respectivamente. Temos então que:

$$ABC = I \quad (2.30)$$

e também:

$$e^{i\alpha}AXBXC = e^{i\alpha}R_z(\beta)R_y(\gamma)R_z(\delta) = U \quad (2.31)$$

pela equação 2.20.

Com este resultado, é possível se realizar a seguinte decomposição:

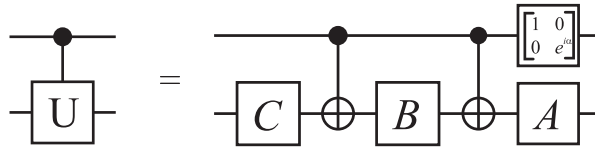


Figura 8: Decomposição de porta U controlada

Ou seja, qualquer operação sobre dois qubits do tipo $C(U)$ pode ser realizada exatamente por uma seqüência de 4 portas de um qubit mais 2 portas $CNOT$.

2.8.3 Decomposição de $C^2(U)$

Para se decompor qualquer operação unitária U de um qubit controlada por dois qubits ($C^2(U)$) usa-se um procedimento um pouco diferente. A operação $C^2(U)$ pode ser decomposta em portas $CNOT$ e portas de um qubit controladas por um qubit $C(V)$ e $C(V^\dagger)$, onde $V^2 = U$. A Figura 9 mostra essa decomposição:

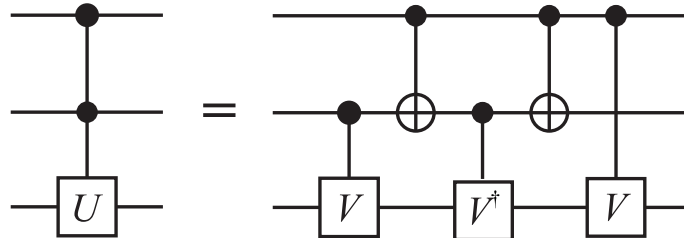


Figura 9: Decomposição de porta U controlada por 2 qubits

Ou seja, qualquer operação sobre três qubits do tipo $C^2(U)$ pode ser realizada exatamente por uma seqüência de três portas controladas do tipo $C(U)$ mais duas portas $CNOT$.

Assim, as portas Hadamard, as portas de fase e a porta CNOT formam um conjunto infinito e universal de portas. Pode-se mostrar que qualquer operação unitária de n qubits pode ser implementada exatamente com $O(4^n n)$ portas desse conjunto^[21].

2.8.4 Decomposição em Produto de Matrizes de Dois Níveis

Este procedimento se aplica a qualquer operação unitária de m qubits (para $m > 1$), e a decompõe em portas $C^{m-1}NOT$ e $C^{m-1}(\tilde{U})$.

O procedimento consiste no seguinte: a partir da matriz (de m bits) U , faz-se a decomposição desta em matrizes de dois níveis de forma que:

$$U_n \dots U_2 U_1 U = I \quad (2.32)$$

Em seguida, decompõe-se cada uma dessas matrizes U_i em circuitos compostos por $C^{m-1}NOT$ s e $C^{m-1}(\tilde{U}_i)$ s por um procedimento que não será detalhado aqui (v. Nielsen e Chuang^[36] para detalhes), resultando em um circuito como o da Figura 10.

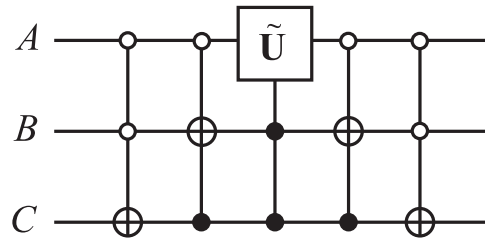


Figura 10: Construção de operador unitário de dois níveis U .

Repare que alguns controles são representados por bolas vazias. Isto indica que o controle é ativado pelo qubit $|0\rangle$ em vez do $|1\rangle$ usual.

Pode-se notar que, neste ponto, a única porta básica usada na decomposição dos circuitos tem sido a $C^n NOT$ (porta NOT controlada por n qubits). As outras portas usadas são portas $C^n(U)$ (porta unitária U controlada por n qubits) ou U , onde U opera sobre um qubit. Na seção que segue, será apresentada uma forma de se aproximar a implementação de uma porta quântica de um qubit usando-se apenas as portas quânticas básicas T e H .

2.8.5 Universalidade por Aproximação

Embora, como visto acima, possamos ter conjuntos infinitos universais de portas quânticas (e.g., CNOT e portas de um qubit), a implementação dessas portas não é livre de erros. Porém, podemos ter um conjunto discreto de portas que podem ser usadas para realizar computação quântica universal de uma maneira livre de erros utilizando códigos corretores de erros.

Suponha que U e V sejam dois operadores unitários de mesma dimensão. U é o operador unitário *alvo*, que se deseja implementar, e V é o operador unitário que se tem

disponível, implementado. Define-se o *erro* quando V é implementado em vez de U por

$$E(U, V) \equiv \max_{|\psi\rangle} \|(U - V)|\psi\rangle\| \quad (2.33)$$

onde o **max** é sobre todos os estados quânticos normalizados $|\psi\rangle$ no espaço de estados.

Se várias portas são aplicadas em seqüência, tem-se:

$$E(U_m U_{m-1} \dots U_1, V_m V_{m-1} \dots V_1) \leq \sum_{j=1}^m E(U_j, V_j) \quad (2.34)$$

2.8.5.1 Universalidade de *Hadamard*, $\pi/8$ e *CNOT*

Considere as portas que implementam as operações T e HTH . T é, com uma fase global, uma rotação de $\pi/4$ radianos sobre o eixo \hat{z} . Enquanto que HTH é uma rotação de $\pi/4$ sobre o eixo \hat{x} . Compondo estas duas operações resulta em (à diferença de uma fase global):

$$\begin{aligned} \exp\left(-i\frac{\pi}{8}Z\right) \exp\left(-i\frac{\pi}{8}X\right) &= \left[\cos\frac{\pi}{8}I - i\sin\frac{\pi}{8}Z\right] \left[\cos\frac{\pi}{8}I - i\sin\frac{\pi}{8}X\right] \\ &= \cos^2\frac{\pi}{8}I - i\left[\cos\frac{\pi}{8}(X+Z) + \sin\frac{\pi}{8}Y\right] \sin\frac{\pi}{8} \end{aligned} \quad (2.35)$$

que é uma rotação sobre um eixo $\vec{n} = (\cos\frac{\pi}{8}, \sin\frac{\pi}{8}, \cos\frac{\pi}{8})$, com vetor unitário correspondente \hat{n} , e por um ângulo θ definido por $\cos(\frac{\theta}{2}) \equiv \cos^2\frac{\pi}{8}$. Ou seja, usando apenas portas *Hadamard* e $\pi/8$ pode-se construir $R_{\hat{n}}(\theta)$.

A partir disso, tem-se que:

$$E(R_{\hat{n}}(\alpha), R_{\hat{n}}(\theta)^n) < \frac{\epsilon}{3}, \quad (2.36)$$

onde ϵ é o erro máximo permitido.

Agora já se pode verificar que qualquer operação de um qubit pode ser aproximada com precisão arbitrária usando apenas portas *Hadamard* e $\pi/8$. É fácil ver que, para qualquer α :

$$H R_{\hat{n}}(\alpha) H = R_{\hat{m}}(\alpha) \quad (2.37)$$

onde \hat{m} é um vetor unitário na direção $(\cos\frac{\pi}{8}, -\sin\frac{\pi}{8}, \cos\frac{\pi}{8})$, ou seja, perpendicular a \hat{n} . Disto, segue que:

$$E(R_{\hat{m}}(\alpha), R_{\hat{m}}(\theta)^n) < \frac{\epsilon}{3} \quad (2.38)$$

e, daí:

$$U = R_{\hat{n}}(\beta) R_{\hat{m}}(\gamma) R_{\hat{n}}(\delta) \quad (2.39)$$

com uma fase global, que pode ser desprezada (comparar com (2.19)). Combinando os resultados (2.36) e (2.38), resulta:

$$\begin{aligned} E(U, R_{\hat{n}}(\beta)^{n_1} H R_{\hat{n}}(\gamma)^{n_2} H R_{\hat{n}}(\delta)^{n_3}) \\ = E(U, R_{\hat{n}}(\beta)^{n_1} R_{\hat{m}}(\gamma)^{n_2} R_{\hat{n}}(\delta)^{n_3}) \\ < \epsilon \end{aligned} \quad (2.40)$$

Ou seja, dados um operador unitário U qualquer de um qubit, e uma taxa de erro $\epsilon > 0$, é possível aproximar U dentro de ϵ usando um circuito composto unicamente de portas Hadamard e $\pi/8$.

Assim, com a porta Hadamard, a porta $\pi/8$ e mais a porta $CNOT$, pode-se implementar qualquer operação unitária sobre um número qualquer de qubits com grau arbitrário de precisão. Um outro conjunto de portas universais equivalente é o conjunto formado pelas portas Hadamard, Fase (S), $CNOT$ e $Toffoli$. Todas essas portas podem ser implementadas de maneira tolerante a falhas^[36].

2.9 Medição

Em um circuito clássico, quando se deseja saber o valor de um bit, basta realizar uma leitura do bit especificado. Este é um processo simples, sem problemas, pois a leitura de um bit não altera seu estado. Porém, em um qubit em estado de superposição, por exemplo, $|\psi\rangle = a|0\rangle + b|1\rangle$, não há maneira de se saber os valores de a ou de b . A única forma de se obter alguma informação sobre o qubit é realizando uma *medição*. Realizar uma medição é fazer um teste no qubit, que resultará em uma modificação do seu estado para um dos estados da base, $|0\rangle$ ou $|1\rangle$ na base computacional, que é o resultado da medição (note-se que os resultados podem ser outros, se for usada uma base diferente da computacional na medição). Após a medição, não há como se saber os valores originais do qubit (valores de a e b).

As medições quânticas são descritas por um conjunto $\{M_m\}$ de *operadores de medição*. Esses operadores agem sobre o espaço de estados sendo medidos. O índice m se refere aos resultados das medições que possam ocorrer no experimento. Se o estado do sistema quântico for $|\psi\rangle$ imediatamente antes da medição, então a probabilidade de que o resultado m ocorra é dada por

$$p(m) = \langle \psi | M_m^\dagger M_m | \psi \rangle, \quad (2.41)$$

e o estado do sistema após a medição é

$$|\psi_o\rangle = \frac{M_m |\psi\rangle}{\sqrt{\langle \psi | M_m^\dagger M_m | \psi \rangle}}. \quad (2.42)$$

Os operadores de medição satisfazem a *equação de completude*,

$$\sum_m M_m^\dagger M_m = I \quad (2.43)$$

2.9.1 Medição Projetiva

Um caso especial importante de medição é a classe de medições conhecida como *medições projetivas*. Uma medição projetiva é descrita por uma *observável* (M), um operador hermitiano⁶ no espaço de estados do sistema sendo observado. A observável

⁶Um operador hermitiano A é aquele que obedece à equação $A = A^\dagger$

tem a seguinte decomposição espectral:

$$M = \sum_M m P_m \quad (2.44)$$

onde P_m é o projetor sobre o autoespaço de M com autovalor m . Os possíveis resultados da medição correspondem aos autovalores (m) da observável. Medindo-se o estado $|\psi\rangle$, a probabilidade de se obter o resultado m é dada por:

$$p(m) = \langle \psi | P_m | \psi \rangle \quad (2.45)$$

e o estado do sistema, correspondente ao resultado m , imediatamente após a medição, será:

$$|\psi_o\rangle = \frac{P_m |\psi\rangle}{\sqrt{p(m)}} \quad (2.46)$$

2.9.2 POVM

Às vezes não se está interessado nos estados após a medição, mas sim nas probabilidades dos possíveis resultados. Para casos assim, a medição POVM é especialmente útil.

Dado um conjunto de operadores positivos⁷ $\{Q_m\}$ cujo somatório é a identidade:

$$\sum_m Q_m = I \quad (2.47)$$

chamamos de POVM (*positive operator-valued measure*) ao tipo medição descrita por esses operadores de medição.

Os operadores Q_m podem ser obtidos dos operadores de medição:

$$Q_m = M_m^\dagger M_m \quad (2.48)$$

2.10 Estados Mistos

O estado $|\psi\rangle$ de um sistema quântico é dito ser um *estado puro* se ele é conhecido com exatidão. Caso contrário, ou seja, se apenas uma distribuição de probabilidades $\{(p_i, |\psi_i\rangle)\}$ é conhecida, o estado do sistema é dito ser uma *mistura estatística*, e é chamado de *estado misto* (ou *misturado*). Uma maneira eficiente de se representar estados mistos é através do conceito de *operador-densidade* (ou *matriz-densidade*). O operador-densidade também pode representar estados puros, pois é uma formulação matematicamente equivalente ao vetor de estados.

O operador-densidade para um estado puro $|\psi\rangle$, é definido como:

$$\rho = |\psi\rangle\langle\psi| \quad (2.49)$$

⁷Operadores positivos são aqueles cujos autovalores são não-negativos.

Para um estado misto, a representação é da seguinte forma:

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i| \quad (2.50)$$

onde os $|\psi_i\rangle$ são estados de um conjunto de possíveis estados em que cada um deles tem uma probabilidade p_i de ser o estado do sistema.

Um operador-densidade ρ possui as seguintes propriedades:

- Traço de ρ é igual a 1 ($\text{tr}(\rho) = 1$);
- ρ é positivo.

Uma forma de se saber se o operador-densidade representa um estado puro ou misto é calculando-se seus autovalores:

- se um dos autovalores for 1 (e, obviamente, os outros forem 0, pois os autovalores somam a 1), então o estado é puro;
- em todos os outros casos, o estado é misto.^[28]

Para se aplicar um operador U a um operador-densidade (ou, aplicar uma porta U a um estado misto) segue-se a equação:

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i| \xrightarrow{U} \sum_i p_i U|\psi_i\rangle\langle\psi_i|U^\dagger = U\rho U^\dagger \quad (2.51)$$

2.10.1 Traço Parcial

Uma das aplicações mais importantes do operador-densidade é como ferramenta para descrever subsistemas de um sistema quântico composto. Esta descrição é proporcionada pelo *operador-densidade reduzido*, que obtemos através da operação do *traço parcial*, explicada a seguir.

Suponha que tenhamos os sistemas físicos A e B , cujo estado é descrito por um operador-densidade ρ^{AB} . O operador-densidade reduzido para o sistema A é definido por

$$\rho^A \equiv \text{tr}_B(\rho^{AB}) \quad (2.52)$$

onde tr_B é um mapa de operadores conhecido como *traço parcial* sobre o sistema B . O traço parcial é definido por:

$$\text{tr}_B(|a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2|) \equiv |a_1\rangle\langle a_2| \text{tr}(|b_1\rangle\langle b_2|) \quad (2.53)$$

onde $|a_1\rangle$ e $|a_2\rangle$ são dois vetores no espaço de estados de A , $|b_1\rangle$ e $|b_2\rangle$ são dois vetores no espaço de estados de B , e tr é a operação traço, ou seja, a soma dos elementos da diagonal da matriz.

2.10.2 Canais Quânticos

Um problema básico tanto Computação Quântica quanto na Informação Quântica é transporte seguro e sem erros de informação entre dois pontos. Até mesmo o armazenamento da informação quântica pode ser problemático. Esses problemas se dão porque o sistema não é totalmente isolado do ambiente, sofrendo interferências externas — por exemplo, transporte de um qubit de informação em um fóton através de uma fibra ótica, onde pode haver ruído por irregularidades na fibra, ou mesmo pela absorção do fóton em algum ponto do percurso^[28].

Imagine um sistema quântico (uma partícula de spin 1/2 ou um fóton) que entre em um “tubo” (designação de um meio físico mediador, como, por exemplo, uma fibra ótica) com estado $|\psi\rangle$. Dizemos que o canal é *perfeito* se o mesmo estado $|\psi\rangle$ emerge na outra extremidade do tubo, para qualquer $|\psi\rangle$ na entrada. Porém, se tivermos uma perturbação tal que o resultado na saída seja $U|\psi\rangle$, onde U é um operador unitário que depende do campo magnético mas é independente da entrada $|\psi\rangle$, então este canal é chamado *canal unitário*, e é ilustrado na Figura 11 abaixo.



Figura 11: Canal quântico unitário

Se, por outro lado, a perturbação varia aleatoriamente com o tempo, causando uma mudança em seu estado, não se pode descrever a saída do canal em termos de um estado puro para uma dada entrada $|\psi\rangle$, mas sim em termos de um operador-densidade ρ , que depende do estado inicial ρ_0 . É comum se modelar este canal como um *canal linear* da seguinte forma ilustrada na Figura 12.

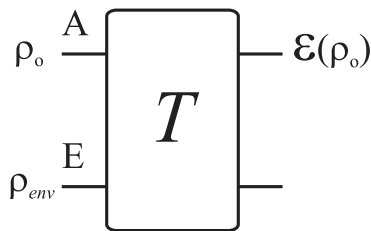


Figura 12: Canal quântico linear

Na entrada, tem-se o sistema (partícula) A , representado por ρ_0 , e o ambiente E , representado por ρ_{env} . A interação entre o sistema A , transportando a informação, e o ambiente E é então representada pelo operador unitário T (o canal). Aplicando-se o traço parcial sobre o ambiente na saída do canal obtém-se um operador-densidade reduzido que descreve o estado do sistema na saída do canal, desprezando-se o ambiente, de acordo com a equação abaixo:

$$\varepsilon(\rho_0) = \text{tr}_{env} [T(\rho_0 \otimes \rho_{env})T^\dagger] \quad (2.54)$$

A operação ε é chamada de *superoperador* porque é um mapeamento linear de operadores em operadores, em vez de mapeamento de kets em kets.

3 *Requisitos Gerais para um Novo Simulador*

A definição de requisitos é o primeiro passo para se desenvolver um sistema computacional. Os requisitos expressam as necessidades funcionais e não-funcionais tanto do cliente quanto dos usuários potenciais do sistema. Tendo-se definido os requisitos, é possível planejar o sistema com maior eficiência, utilizando-se uma metodologia adequada para tanto.

Neste capítulo, serão apresentados os requisitos gerais (recursos a serem oferecidos) desejáveis para um novo simulador de circuitos quânticos, tanto do ponto de vista funcional quanto do não-funcional.

3.1 Recursos do Sistema

Para que se possa atingir os objetivos definidos no Capítulo 1 deste trabalho, o novo simulador deve oferecer os seguintes recursos com relação à funcionalidade, à usabilidade e à documentação:

3.1.1 Recursos de Funcionalidade

Os requisitos gerais de funcionalidade devem contribuir para se atingir o objetivo de permitir ao usuário modelar e simular algoritmos e canais quânticos. Como foi visto no Capítulo 2, as funcionalidades necessárias para tal devem ser:

Estados puros e mistos: o usuário deve poder representar os estados do seu sistema através de *kets* (estados puros) ou através de matrizes-densidade (estados mistos) de forma que seja possível modelar e simular tanto algoritmos quanto canais quânticos.

Medição: o usuário deve poder simular a realização de uma operação de medição em qualquer ponto do circuito e utilizando qualquer tipo (classe) de medição, como definido na Seção 2.9.

Traço parcial: para simular a interação de um sistema quântico com o ambiente e permitir posteriormente a análise apenas do sistema, desprezando o ambiente, como é o caso de um canal quântico, a ferramenta deve disponibilizar a operação de traço parcial.

Erros: para que seja possível estudar os efeitos de erros ocorridos durante um experimento, a ferramenta deve permitir a geração de erros operacionais (erros na operação das portas) e de erros de descoerência (erros devidos à interferências do ambiente), além de permitir a simulação livre de erros.

Portas: para permitir que o usuário possa implementar qualquer operação unitária, a ferramenta deve disponibilizar um conjunto de portas básicas universais, além de permitir que o usuário crie portas personalizadas de maneira que possa implementar operações unitárias arbitrárias exatas.

Bibliotecas: para que o usuário possa armazenar e reutilizar as portas personalizadas, a ferramenta deve permitir a criação e edição de bibliotecas de portas.

3.1.2 Recursos de Usabilidade

Os requisitos gerais de usabilidade devem contribuir para se atingir o objetivo de permitir ao usuário utilizar a ferramenta de forma fácil e intuitiva. Para tal, a ferramenta deve ter uma interface do usuário que ofereça os seguintes recursos:

Interação: a ferramenta deve oferecer, e o usuário deve poder manipular diretamente, representações gráficas de portas sobre uma grade representando a placa do circuito.

Visualização de estados: a ferramenta deve permitir uma visualização apropriada do estado do sistema nas formas de *braket*, matriz-densidade e histograma, quer no final da simulação ou em qualquer ponto desta.

Edição: o usuário deve poder editar o circuito utilizando as operações de: copiar, recortar e colar portas e grupos de portas.

Desfazer: o usuário deve poder retroceder ou desfazer qualquer ação que ele possa ter realizado no processo de edição do circuito (*undo*).

3.1.3 Recursos de Documentação

Os requisitos gerais de documentação devem contribuir para a compreensão da arquitetura do sistema e do seu processo de desenvolvimento. Para tal, a ferramenta deve vir acompanhada da seguinte documentação:

Processo de desenvolvimento: descrição do processo de desenvolvimento da ferramenta, explicando desde a fase de obtenção de requisitos até a implementação da ferramenta.

Arquitetura do sistema: diagrama descrevendo a arquitetura do sistema, com seus blocos principais, e as relações entre eles.

Modelo conceitual: diagramas detalhados dos blocos constituintes do diagrama de arquitetura e as relações internas entre seus constituintes.

Diagrama de classes: diagramas descrevendo em detalhes (métodos e atributos) as classes (objetos) do sistema e as relações entre elas.

Manual do usuário: documento descrevendo como utilizar a ferramenta em forma de ajuda *on-line* na própria ferramenta.

3.1.4 Outros Requisitos

Além dos requisitos gerais listados acima, outros requisitos se fazem importantes:

Extensibilidade: a ferramenta deverá ser facilmente extensível, de maneira a que versões posteriores à primeira versão possam incorporar novas funcionalidades reutilizando o código existente.

Linguagem de programação: a ferramenta deverá ser implementada em uma linguagem que permita acelerar o desenvolvimento, evitando, por exemplo, problemas com a gerência de objetos na memória, e que atenda também ao requisito de extensibilidade.

Plataforma: a ferramenta deve ser multiplataforma de maneira a poder ser utilizada por um universo maior de usuários.

Estratégia de cálculo: a ferramenta deverá fazer uso de um pacote externo para realizar os cálculos, economizando-se tempo no desenvolvimento e garantindo-se confiabilidade por já terem sido extensivamente testados.

4 *Processo de Desenvolvimento*

Neste capítulo é descrito o processo de desenvolvimento utilizado para a construção do simulador de circuitos quânticos doravante denominado *Zeno*. Considerando o princípio da independência do diálogo, adotou-se uma arquitetura em dois módulos (funcional e de interação), possibilitando a adoção de processos de desenvolvimento diferentes para cada um deles, partindo da análise da tarefa. Para o módulo funcional adotou-se um processo simplificado baseado no Processo Unificado^[33], e para o módulo de interação adotou-se um processo de desenvolvimento baseado na metodologia MEDITE^[30]. A escolha do nome “Zeno” para o simulador de circuitos quânticos desenvolvido neste trabalho foi motivada pelo caráter paradoxal que a Computação quântica apresenta em relação à Computação Clássica em analogia ao paradoxo de Zenão^[4].

4.1 Princípio da Independência do Diálogo

As primeiras abordagens de concepção de sistemas interativos tinham uma tendência a “misturar” fortemente o código da interface com o código que implementa a funcionalidade do sistema. A dificuldade de modificação proibia uma abordagem iterativa, praticamente impedindo de se levar em conta as opiniões de especialistas em comunicação e os resultados da observação do usuário durante a concepção e o desenvolvimento da interface.

Então, uma solução para uma abordagem iterativa do desenvolvimento do sistema é a separação modular entre o código que implementa o componente funcional (aplicação) e o código que implementa o componente responsável pela interação com o usuário (interface). A Figura 13 mostra esta abordagem graficamente. O componente funcional define a funcionalidade do sistema, ou seja, as funções que o sistema disponibiliza para o usuário realizar suas tarefas. O componente de interação é o software responsável pelo diálogo entre o usuário e o sistema. Essa decomposição deve permitir se conceber, desenvolver e modificar os módulos/componentes independentemente um do outro^[19, 20, 22]. O *princípio da independência do diálogo*, termo consagrado para caracterizar essa decomposição, tem se tornado um princípio essencial a respeitar no processo de concepção e desenvolvimento de sistemas interativos, pelas seguintes razões: a possibilidade de um desenvolvimento independente e iterativo dos módulos e de utilização de profissionais e/ou conhecimentos em comunicação (ergonomistas, artistas gráficos, psicólogos da cognição, lingüistas, etc.).

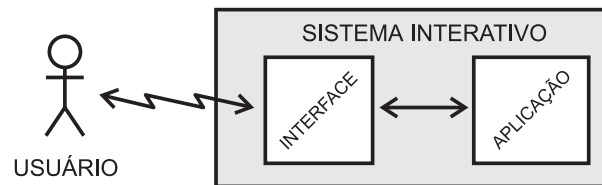


Figura 13: Software interativo

Apesar de independentes, não faria sentido iniciar o processo de desenvolvimento dos dois módulos a partir de análises de requisitos (funcionais e não-funcionais) separadas, ou seja, com requisitos levantados a partir de atividades de análise independentes, pois isso poderia levar a uma incoerência entre os requisitos levantados para cada um dos módulos. Assim, baseado nas experiências de Medeiros^[16] e Cordeiro^[13] com êxito no desenvolvimento de uma ferramenta para análise e modelagem da tarefa, foi decidida a adoção de um processo similar para o desenvolvimento do simulador Zeno, ou seja, seguindo o princípio da independência do diálogo e partindo da análise e modelagem da tarefa “Utilizar uma ferramenta de software para projetar e simular circuitos quânticos”. Para a comunicação e a integração dos módulos foi decidido que, no processo do módulo funcional, a primeira coisa a ser feita após a análise da tarefa seria a identificação dos objetos do domínio, o projeto das classes, e a definição dos serviços que tais classes disponibilizariam para o módulo de interação através de uma API (*Application Program Interface*), que será o ponto de comunicação e integração entre os dois módulos. A API define uma interface geral de interação com o módulo funcional.

4.2 Processo Unificado Simplificado

Para o desenvolvimento do módulo funcional, foi utilizado um processo simplificado baseado no Processo Unificado. O Processo Unificado, conforme descrito por Meloche^[33], engloba atividades de alto e de baixo nível. No mais alto nível, temos os ciclos de desenvolvimento. Os ciclos são compostos por quatro fases: planejamento, elaboração, construção e transição. As fases são divididas em iterações, que identificam, cada uma, alguns aspectos do sistema a serem considerados. As iterações são compostas por fluxos de atividades, que descrevem como cada atividade fornece informações para outra. Por fim, os fluxos de atividades são compostos por atividades de baixo nível, tais como definir classes e seus atributos.

O Processo Unificado tem três características principais, que são as seguintes:

- Direcionado a *use-cases*: os *use-cases* ajudam a identificar quem usa e o que deve ser implementado pelo sistema (funcionalidades). Os *use-cases* são utilizados para assegurar que a evolução do projeto do sistema esteja sempre de acordo com o que o usuário requisitou, fornecendo, portanto a possibilidade de rastreamento entre o que foi implementado e os requisitos do sistema.
- Centrado na arquitetura: procura entender os aspectos dinâmicos e estáticos mais significantes em termos de arquitetura de software, identificando o que deve fazer parte de uma arquitetura e como deve ser feito o projeto e a implementação da mesma.

- Iterativo e incremental: considera prático dividir projetos grandes em projetos menores, ou *mini-projetos*. Cada mini-projeto compreende uma iteração que resulta em um incremento. Uma iteração pode englobar todos os fluxos de atividades de um processo.

4.2.1 As Quatro Fases de um Processo

O Processo Unificado consiste de ciclos, os quais podem se repetir durante a vida do sistema. Um ciclo compreende quatro fases: planejamento, elaboração, construção e transição.

Fase de Planejamento: a fase de planejamento estabelece a viabilidade do projeto e delimita o seu escopo.

Fase de Elaboração: a maioria dos *use-cases* são especificados em detalhes e a arquitetura do sistema é desenvolvida.

Fase de Construção: o produto passa da arquitetura para um sistema completo o suficiente para ser entregue à comunidade de usuários.

Fase de Transição: o objetivo é assegurar que os requisitos foram alcançados ao ponto de satisfazer os clientes.

4.2.2 Fluxos de Atividades

O Processo Unificado identifica fluxos de atividades principais: Requisitos, Análise, Projeto, Implementação e Teste. O levantamento de requisitos tem foco nas atividades de identificação dos requisitos funcionais e não-funcionais do sistema. O produto final dessa atividade é o modelo de *use-cases*. Os requisitos identificados servem de entrada para a atividade de análise, onde são reestruturados em termos mais técnicos do software a ser construído. Um modelo de análise é obtido com tal reestruturação. Do modelo de análise é gerado o modelo de projeto. E deste é gerada a implementação, que representa a codificação do projeto em uma linguagem apropriada, sua compilação, distribuição e geração da documentação. Em seguida, o software gerado durante o a implementação é testado para assegurar que todos os requisitos funcionais foram implementados.

Cada fluxo de atividades abrange um conjunto de atividades principais que devem ser executadas durante cada um deles. As atividades principais são vistas na Tabela 1 abaixo:

Fluxo	Atividade
Requisitos	<ul style="list-style-type: none"> – Identificar e descrever os atores do sistema – Identificar os <i>use-cases</i> – Descrever os use cases apropriadamente – Descrever o modelo de use case – Preparar um glossário com os termos utilizados no sistema

Tabela 1: (continuação)

Análise	<ul style="list-style-type: none"> – Fazer uma análise arquitetural – Identificar as classes do modelo de análise – Organizar as classes identificadas em pacotes – Fazer uma análise de use cases para gerar <i>use-case realizations</i>
Projeto	<ul style="list-style-type: none"> – Fazer um projeto arquitetural – Gerar classes de projeto – Identificar as interfaces de projeto
	<ul style="list-style-type: none"> – Gerar use case realizations de projeto a partir dos <i>use-case realization</i> gerados no <i>workflow</i> de Análise – Identificar subsistemas
Implementação	<ul style="list-style-type: none"> – Implementar o esqueleto da arquitetura – Definir o modelo de implementação – Implementar subsistemas, classes e interfaces – Integrar sistemas
Testes	<ul style="list-style-type: none"> – Planejar os testes a serem feitos para sistema – Projetar e implementar os casos de teste, criando casos de teste executável – Executar os testes e analisar os resultados obtidos

Tabela 1: Atividades de cada fluxo de atividades^[13]

A primeira modificação com relação ao Processo Unificado é que não foram utilizados *use-cases*, mas os objetos gerados a partir da análise de tarefas. Devido a essa modificação, foram modificados também as atividades em cada fluxo de atividade, como se vê na Tabela 2 a seguir:

Fluxo	Atividade
Fluxo	Atividade
Requisitos	– Realizar Análise da Tarefa
Análise	<ul style="list-style-type: none"> – Fazer uma análise arquitetural – Identificar as classes do modelo de análise e as relações entre elas
Projeto	<ul style="list-style-type: none"> – Fazer um projeto arquitetural – Gerar classes de projeto – Identificar as interfaces de projeto
Implementação	<ul style="list-style-type: none"> – Definir o modelo de implementação – Implementar subsistemas, classes e interfaces – Integrar sistemas
Testes	<ul style="list-style-type: none"> – Projetar e implementar testes de unidade, criando casos de teste executável – Executar os testes e analisar os resultados obtidos

Tabela 2: Atividades modificadas

4.3 MEDITE

MEDITE^[30] é uma metodologia baseada na tarefa e orientada a modelos que utiliza o conhecimento ergonômico logo no início do processo de concepção de interfaces de forma a produzir interfaces com alto grau de usabilidade.

MEDITE guia o projetista, especialmente aquele que não tem conhecimento sobre Ergonomia, segundo modelos bem definidos durante todo o processo de concepção de interfaces^[16].

MEDITE define o processo de construção de interfaces ergonômicas em 5 etapas, cada uma das quais gerando seus artefatos. As etapas são:

1. análise e modelagem da tarefa,
2. especificação conceitual da interação,
3. refinamento da especificação conceitual,
4. geração de protótipo, e
5. avaliação distribuída em todas as outras etapas do processo.

A etapa de análise e modelagem da tarefa tem o objetivo de descrever os objetivos dos usuários, assim como os meios utilizados no processo de realização da tarefa. A tarefa deve ser descrita em termos de objetivos, procedimentos, sub-tarefas, restrições, etc., usando o modelo TAOS^[16, 30]. Esta etapa tem como produto (artefato) de entrada informações sobre o usuário e sobre o domínio da tarefa, e como produto de saída a descrição TAOS (árvore TAOS e respectivos descritores).

Na etapa de especificação conceitual de interação produz-se uma especificação conceitual parcial da interação a partir da descrição TAOs da tarefa, determinando a estrutura e o estilo dos aspectos de interação de acordo com o modelo EDITOR. A estrutura obtida reflete diretamente a estrutura (hierarquia) e a apresentação de janelas da interface. Essa especificação permite uma visão inicial (esboço) da apresentação e do encadeamento do diálogo da interface em construção.

Na etapa de refinamento e especificação conceitual completa da interação, define-se os atributos da árvore EDITOR, ou seja, a especificação gerada na etapa anterior é completada e refinada com os atributos de *apresentação*, *abstração* e *controle*, definindo o encadeamento do diálogo entre as janelas da interface.

Na etapa de geração de protótipo é realizada a codificação da interface a partir da especificação conceitual completa definida na etapa anterior.

A avaliação é distribuída nas diversas etapas do processo.

4.4 Análise da Tarefa

Nesta seção é descrita a fase de análise e modelagem da tarefa “Utilizar uma ferramenta de software para projetar e simular circuitos quânticos”. Trata-se do ponto de partida dos processos de concepção do módulo da interface e do módulo funcional do simulador Zeno. A etapa de análise e modelagem da tarefa possui como entrada os dados sobre o usuário e sobre a tarefa e apresenta como saída a descrição da tarefa segundo o modelo TAOS.

4.4.1 Aquisição de Dados sobre a Tarefa e sobre o Usuário

O processo de coleta de dados sobre a tarefa “Utilizar uma ferramenta de software para projetar e simular circuitos quânticos” e sobre o usuário foi feito tomando como base principal um levantamento sobre as principais ferramentas para simulação de circuitos quânticos existentes atualmente. A partir deste levantamento, foi realizada uma análise dos recursos oferecidos por cada um deles, levando em conta aspectos de funcionalidade, de usabilidade, de documentação e de implementação, conforme descrito no Capítulo 1.

Foram levadas em conta também as necessidades do grupo local de estudos em Computação e Informação Quântica para se definir os requisitos de interesse.

4.4.2 Descrição da Tarefa Utilizando o TAOS

Abaixo, é apresentada uma parte da descrição da tarefa “Utilizar uma ferramenta de software para projetar e simular circuitos quânticos”, realizada a partir do levantamento dos dados sobre a tarefa e o usuário descrito na subseção anterior. A descrição completa da tarefa pode ser encontrada na Internet^[9].

A Figura 14, a seguir, apresenta um trecho da árvore de modelagem da tarefa que representa os níveis mais altos da hierarquia, ou seja, os objetivos gerais da tarefa “Utilizar uma ferramenta de software para projetar e simular circuitos quânticos”. Essa descrição TAOS da tarefa expressa fielmente os dados obtidos através do levantamento citado, o que significa afirmar que o usuário que deseja projetar e/ou simular circuitos quânticos utilizando o simulador Zeno terá que executar pelo menos uma das seguintes sub-tarefas: ativar ambiente, gerenciar circuitos, projetar circuito, simular circuito, gerenciar portas e bibliotecas, solicitar ajuda e desativar ambiente, segundo sua própria vontade, como podemos observar na Figura 14.

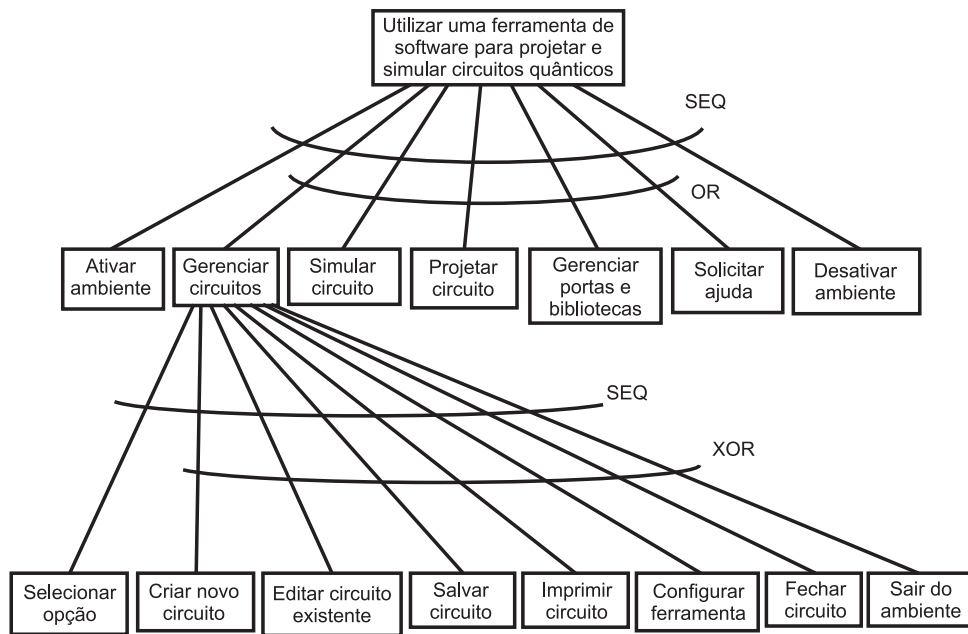


Figura 14: Tarefa “Utilizar uma ferramenta de software para projetar e simular circuitos quânticos”

Na Figura 15 a seguir, é apresentado um trecho da árvore que representa a subtarefa “Configurar ferramenta”. O usuário que desejar definir a configuração default da ferramenta deverá executar uma ou mais das seguintes ações: Definir número default de qubits, definir número default de colunas, definir tipo de estado.

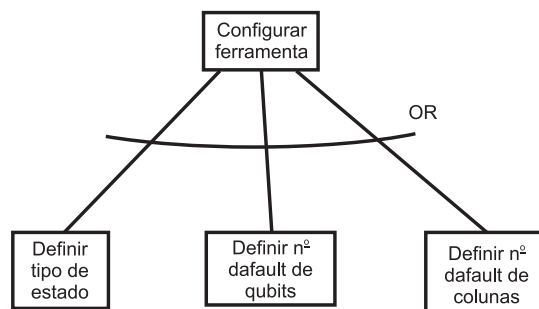


Figura 15: Tarefa “Configurar ferramenta”

Após a apresentação de dois trechos da árvore TAOS da descrição “Utilizar uma ferramenta de software para projetar e simular circuitos quânticos”, serão mostrados os descritores relativos à tarefa “Gerenciar circuitos” e à ação “Definir número de colunas” respectivamente, escolhidos de forma aleatória, e representados nas Tabelas 3–7. Os descritores definem os elementos e atributos das tarefas e ações da descrição gerada.

Classe	Tarefa
Nome	T1.2 – Gerenciar circuitos
Descrição	Gerencia circuitos: cria, abre e salva circuitos em arquivos, imprime circuitos, define propriedades da ferramenta.
Pré-situação	PS de A1.1
Pós-situação	PS de T1.2
Ações	A1.2.1, A1.2.7
Subtarefas	T1.2.2, T1.2.3, T1.2.4, T1.2.5, T1.2.6

Tabela 3: Descritor — tarefa ‘gerenciar circuitos’

Classe	Situação
Nome	PS de T1.2
Descrição	Situação após a execução de T1.2
Objetos	ambiente
Restrição	disponível(ambiente)
Como atingir	A1.2.1, T1.2.2, T1.2.3, T1.2.4, T1.2.5, T1.2.6, A1.2.7

Tabela 4: Descritor — pós-situação de ‘gerenciar circuitos’

Classe	Ação
Nome	A1.2.2.1.2 – Definir número de colunas
Descrição	Define número de colunas do novo circuito
Pré-situação	SI de A1.2.2.1.2
Pós-situação	PS de A1.2.2.1.2
Agente	usuário
Instrumentos	mouse, teclado
Objetos	caixa de texto ‘número de colunas’

Tabela 5: Descritor — ação ‘definir número de colunas’

Classe	Situação
Nome	SI de A1.2.2.1.2
Descrição	Requisitos para definir número de colunas
Objetos	caixa de texto ‘número de colunas’, usuário, mouse, teclado
Restrição	disponível(caixa de texto ‘número de colunas’), disponível(usuário), disponível(mouse), disponível(teclado), preenchido-com-default(caixa de texto ‘número de colunas’)
Como atingir	T1.2.2.1

Tabela 6: Descritor — requisito de ‘definir número de colunas’

Classe	Situação
Nome	PS de A1.2.2.1.2
Descrição	Situação após A1.2.2.1.2
Objetos	caixa de texto ‘número de colunas’
Restrição	preenchido(caixa de texto ‘número de colunas’)
Como atingir	A1.2.2.1.2

Tabela 7: Descritor — pós-situação de ‘definir número de colunas’

Através da análise de tarefas realizada, foram identificados tanto objetos do módulo funcional quanto do módulo de interface. As seções seguintes explicam como, a partir dos resultados da análise de tarefas, e utilizando-se processos diferentes, cada um dos módulos foi construído.

4.5 Desenvolvimento do Módulo Funcional

Nesta seção são descritas as fases do desenvolvimento do módulo funcional, explanando o que foi realizado em cada fase.

4.5.1 Fase de Planejamento

Na fase de planejamento foram realizadas as atividades de obtenção de requisitos e análise do sistema. Os requisitos foram obtidos através da análise e modelagem da tarefa, explicada na seção anterior, realizada de forma iterativa e incremental, através de reuniões semanais, com o intuito de identificar novas funcionalidades a cada iteração até que se atingiu o consenso de que todas as funcionalidades tinham sido exploradas.

Na atividade de análise, o modelo de tarefas foi utilizado para realizar a identificação da arquitetura e das entidades principais do sistema. A arquitetura representa os principais blocos do sistema, e é representada na Figura 16. O modelo de tarefas fornece as principais entidades do sistema como objetos presentes nas pré e pós-situações das tarefas. Essas entidades foram então representadas como classes. Entre as entidades principais estão a classe que representa um circuito quântico (QuantumCircuit), as classes que representam portas quânticas (QuantumGate, MatrixGate, MeasureGate) e as classes que representam estados (PureState, MixedState). Essas entidades estão representadas no modelo conceitual, como se pode ver na Figura 17.

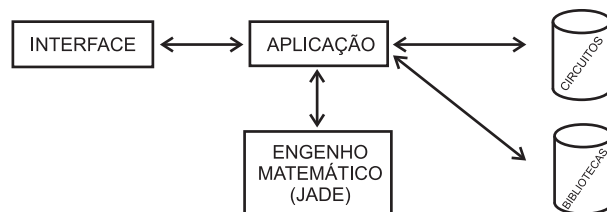


Figura 16: Arquitetura

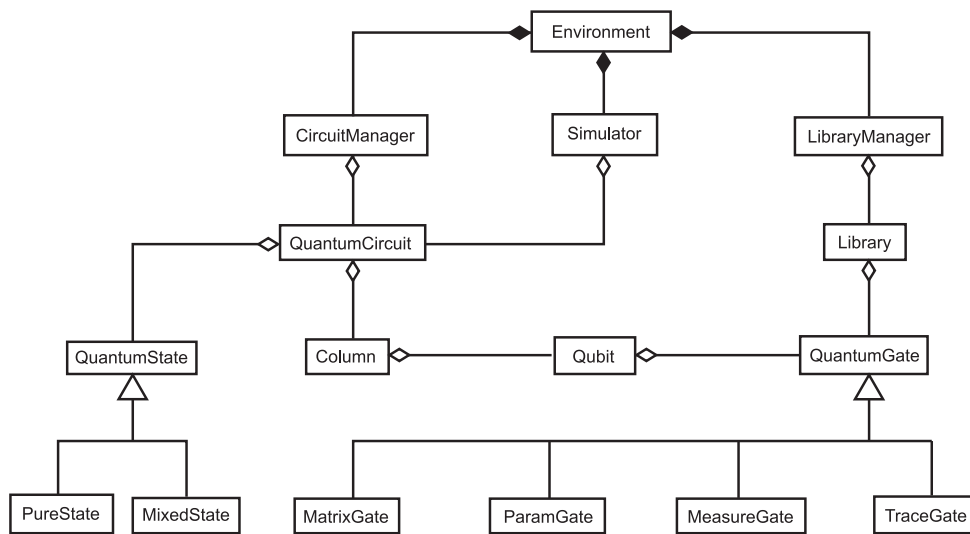


Figura 17: Modelo conceitual do módulo funcional

4.5.2 Fase de Elaboração

Nesta fase é realizado o projeto do sistema propriamente dito. O projeto é dirigido à implementação, e durante o mesmo são especificadas as entidades do modelo relacional, detalhando-se cada classe em suas propriedades e seus métodos, assim como as relações entre as classes.

O projeto arquitetural é o produto da análise da arquitetura, e representa a união da evolução da arquitetura base com algumas informações a respeito das decisões tomadas, tais como: linguagem de programação escolhida (Java), plataforma sobre a qual o sistema será executado, a escolha de padrões de projeto para aumentar o reuso das classes do projeto^[45], entre outros. No caso do presente projeto, o modelo de tarefas também entra na arquitetura, pois contém informações a respeito do comportamento dinâmico do sistema.

A fim de se produzir um código mais reusável, facilitando futuras alterações, a arquitetura base do módulo funcional foi definida em termos de interfaces, classes abstratas e padrões de projeto. As classes identificadas durante a fase de planejamento continuaram em vigor, porém o relacionamento entre elas mudou ligeiramente, devido ao surgimento das interfaces e das classes abstratas.

4.5.2.1 Decisões de Projeto

Durante desenvolvimento do simulador Zeno, tendo em vista principalmente o aceleração do desenvolvimento do simulador, algumas decisões foram tomadas a fim de se simplificar os algoritmos a serem implementados. Entre elas, as principais são as seguintes:

- Permitir apenas uma porta por coluna, simplificando assim os algoritmos de simulação e a implementação da interface gráfica do usuário.
- Permitir que portas *multi-qubit*¹ (à exceção das portas controladas) ocupem qubits

¹Portas que ocupam mais de um qubit.

apenas em posições contíguas. Esta estratégia, utilizada em todos os outros simuladores avaliados^[10], visa também a simplificação dos algoritmos de simulação e a implementação da interface gráfica do usuário.

- As portas controladas, na simulação, geram uma matriz correspondente à operação controlada considerando, além dos bits de controle, também os bits intermediários entre a porta e os controles e entre dois controles não-contíguos. Isso simplifica o algoritmo de simulação, porém pode acarretar um considerável aumento no tempo de simulação, visto que gera matrizes maiores.

4.5.3 Fase de Construção

A fase de construção foi composta basicamente pelas atividades de implementação e testes. Porém, foi necessário realizar a atividade de projeto algumas vezes a fim de se efetuar correções no projeto do sistema.

Nesta fase, foram implementadas e testadas as classes geradas na fase anterior. Esta fase gerou como artefatos: o código-fonte (em Java) referente às classes planejadas, documentação do código (JavaDoc) e arquivos de testes das classes implementadas^[9].

4.5.4 Fase de Transição

Na fase de transição, o módulo funcional foi disponibilizado para integração ao módulo de interação. O resultado da integração foi o simulador Zeno.

Os artefatos que acompanham a distribuição do módulo funcional são:

Código fonte — arquivos Java no pacote **src** e seus subpacotes: **src.quantumcircuit.circuit**, **src.quantumcircuit.gate**, **src.quantumcircuit.library** e **src.quantumcircuit.tests**;

Documentação das classes (API) — trata-se de um conjunto de documentos HTML interligados, permitindo a navegação entre a documentação das diversas classes que compõem o módulo com explicações de como utilizar cada uma delas adequadamente. Esse documento é direcionado para quem deseja utilizar o módulo funcional como plataforma para implementar uma ferramenta para projeto e simulação de circuitos quânticos;

Arquivos de *byte code* Java — arquivos pré-compilados do código fonte do módulo funcional;

Biblioteca JADE adaptada (jadeForZeno.jar) — a biblioteca JADE^[14] (Java Addition to the Default Environment) é um pacote Java desenvolvido para auxiliar na implementação de software científico em Java. Os arquivos pré-compilados e o código-fonte da biblioteca utilizada para realizar os cálculos matemáticos do simulador. Esta biblioteca é distribuída, assim como o próprio simulador, com código aberto.

4.6 Desenvolvimento do Módulo de Interação

Nesta seção, é descrito o processo de definição e implementação do módulo de interface a partir da descrição da tarefa “Utilizar uma ferramenta de software para projetar e simular circuitos quânticos”.

4.6.1 Identificação de Janelas e Objetos Visuais

A partir da descrição da tarefa “Utilizar uma ferramenta de software para projetar e simular circuitos quânticos”, foi feita uma listagem com todas as janelas identificadas do sistema. Dentro de cada janela, foram identificados os elementos visuais (botões, caixas de texto, etc.) presentes. Também foram identificados as formas de interação do usuário com o sistema.

4.6.2 Implementação

De posse da especificação dos objetos visuais, foi realizada a implementação do módulo de interação. Alguns ajustes foram realizados devido a dificuldades técnicas ou quando se percebeu uma maneira mais fácil de utilizar a ferramenta.

A implementação do módulo de interação foi feita usando a linguagem Java, com o pacote gráfico JHotDraw^[3], que oferece várias vantagens no que se refere ao desenho e à manipulação direta de objetos na tela.

4.7 Integração

A integração do módulo de interação com o módulo funcional foi realizada em paralelo à implementação do módulo de interação. Esta estratégia foi adotada porque algumas partes do módulo de integração dependiam de recursos da parte funcional para funcionar (e.g., a criação de um circuito com dimensões e tipo definidos, a montagem do circuito, entre outras).

4.7.1 Testes

Durante a integração, foram realizados testes para verificar se os dois módulos, o funcional e o de interação, estavam funcionando em conjunto. Os testes consistiram em usar a ferramenta realizando as mais diversas tarefas, como montar um circuito, simular um circuito, editar uma porta, etc.

5 *Análise de resultados*

Neste capítulo, será apresentada uma análise dos resultados obtidos com a implementação do simulador de circuitos quânticos Zeno. Esta análise terá dois focos principais: a análise dos recursos implementados em comparação com os recursos planejados do sistema, e testes de correção e desempenho comparativo (*benchmark*) do Zeno com outros simuladores de circuitos quânticos.

5.1 Recursos Planejados *vs.* Implementados

Nesta seção, serão comparados os recursos implementados no simulador Zeno com os recursos planejados para um novo simulador de circuitos quânticos. Seguindo a estrutura adotada no Capítulo 3, os recursos serão divididos em três categorias: recursos de funcionalidade, recursos de usabilidade e recursos de documentação.

5.1.1 Recursos de Funcionalidade

Estados puros e mistos: o simulador Zeno permite que se trabalhe tanto com estados puros quanto com estados mistos. Para tanto, conforme mostrado no diagrama de classes (v. documentação do projeto^[9]), o sistema dispõe de duas classes: `PureState` e `MixedState`, ambas estendendo uma classe abstrata genérica chamada de `QuantumState`, que apenas descreve o comportamento de suas subclasses. As classes `PureState` e `MixedState` contém as propriedades características dos respectivos tipos de estados, como a aridade (número de qubits ocupados pelo estado) e a matriz que descreve cada estado. A principal diferença entre elas é a matriz, que na classe `PureState` é uma matriz-coluna e na classe `MixedState` é uma matriz quadrada.

Além das classes `PureState` e `MixedState`, foi definida uma terceira classe, `CompositeState`, que permite representar estados independentes de um sistema sem a necessidade de realizar o produto tensorial, o que geraria uma explosão exponencial do espaço requerido para armazenar os estados do sistema.

Para que fosse possível atender ao requisito de oferecer a possibilidade de trabalho com estados puros e mistos, conforme a Seção 3.1.1, as classes `MatrixGate` e `ParamGate` foram definidas de forma a tratar estados puros e mistos de maneira diferente quando da aplicação de uma porta sobre um estado. Quando o estado é puro, a aplicação se dá segundo a seguinte equação:

$$|\psi_o\rangle = U|\psi_i\rangle, \quad (5.1)$$

onde $|\psi_o\rangle$ é o estado resultante da aplicação da porta representada pela matriz U ao estado de entrada $|\psi_i\rangle$. E quando o estado é misto, a aplicação se dá da seguinte forma:

$$\rho_o = U\rho_iU^\dagger, \quad (5.2)$$

onde ρ_o é o estado resultante da aplicação da porta representada pela matriz U ao estado de entrada ρ_i .

Outras classes também tratam estados puros e mistos de forma diferente, como é o caso da classe `MeasureGate`, que é explicada a seguir.

Medição: para atender os requisitos definidos na Seção 3.1.1, foram definidas as classes `MeasureGate`, e subclasses desta: `GeneralMeasureGate`, `ProjectiveMeasureGate` e `POV-MeasureGate`. Destas últimas, só foi implementada efetivamente a classe `ProjectiveMeasureGate`. Assim, essa classe, além de permitir que o usuário simule a realização de uma medida projetiva, permite também que ele possa realizar a medição em qualquer ponto do circuito e em qualquer base além da base computacional.

A medição de estados puros e de estados mistos é realizada de formas diferentes. A medição de um estado puro é feita segundo a seguinte equação:

$$|\psi_o\rangle = \frac{M_j|\psi_i\rangle}{\sqrt{p(M_j)}}, \quad (5.3)$$

onde M_j é o operador de projeção do j -ésimo estado da base de medição (j é gerado aleatoriamente) e $p(M_j)$ é a probabilidade de o j -ésimo estado da base ser o resultado da medição. Para estados mistos, a equação é a seguinte:

$$\rho_o = \frac{M_j\rho_iM_j^\dagger}{\sqrt{p(M_j)}}. \quad (5.4)$$

Traço parcial: para implementar a operação de traço parcial, conforme requerido (v. Seção 3.1.1), foi definida a classe `PartialTraceGate`, definida como uma porta. Essa classe implementa a porta de traço parcial que pode ser aplicada a um qubit de um sistema composto, permitindo a visualização do estado composto pelos demais qubits do sistema quântico. Uma outra porta, que funciona em conjunto com a porta de traço parcial é a porta de bit de trabalho (*ancilla*), implementada pela classe `AncillaBitGate`. Esta porta insere um qubit com estado definido ($|0\rangle\langle 0|$ ou $|1\rangle\langle 1|$) na posição do qubit eventualmente descartado pela porta de traço parcial.

Erros: apesar de ser um recurso planejado (v. Seção 3.1.1), a implementação atual do simulador Zeno não contempla simulação da ocorrência de erros operacionais nem de erros de descoerência. A implementação dos erros se daria na classe `Simulator`, que controla a simulação de circuitos. Esta classe define métodos que inserem erros no estado sendo simulado. Para implementar a ocorrência de erros em futuras versões do simulador, será preciso apenas definir uma nova classe que estenda a classe `Simulador`, de forma a implementar estes métodos.

Portas: conforme planejado (v. Seção 3.1.1), um conjunto de portas básicas universais é disponibilizado através das classes `MatrixGate`, que implementa as portas X, Y, Z, H, S, T, CNOT, Toffoli e SWAP, e `ParamGate`, que implementa portas parametrizadas de rotação, cujos parâmetros definem a matriz da porta segundo a equação abaixo:

$$U = \cos(\theta/2)I + i \sin(\theta/2)(\vec{n} \cdot \vec{\sigma}) = e^{i\theta\vec{n}\cdot\vec{\sigma}} \quad (5.5)$$

Além disso, o simulador Zeno permite que o usuário crie portas personalizadas definindo matrizes unitárias diferentes para as portas do tipo MatrixGate.

Bibliotecas: conforme planejado (v. Seção 3.1.1), o simulador Zeno permite a criação e edição de bibliotecas de portas através da classe Library, que define métodos para o armazenamento e recuperação de portas em arquivos para reutilização. O simulador também oferece um conjunto de bibliotecas predefinidas, implementadas como classes que estendem a classe Library, como as bibliotecas *1-qubit*, com portas básicas de um qubit, *multi-qubit*, com portas de dois ou mais qubits, e *miscellaneous*, com portas diversas, como a de medição e de traço.

5.1.2 Recursos de Usabilidade

Interação: de acordo com o requisitos de usabilidade definidos na Seção 3.1.2, a interface do usuário implementada permite que o usuário manipule diretamente representações gráficas de portas sobre uma grade representando a placa do circuito. Utilizando o mouse, o usuário pode “arrastar” uma porta de qualquer ponto do circuito e “soltá-la” em qualquer outro ponto desde que a coluna esteja vazia (não contenha nenhuma outra porta) e que a porta fique dentro dos limites do circuito. A Figura 18 abaixo ilustra a representação gráfica de um circuito como visto pelo usuário do simulador.

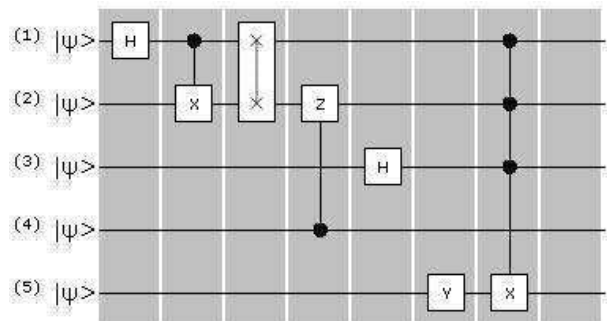


Figura 18: Circuito representado no simulador Zeno

Visualização de estados: o simulador Zeno permite uma visualização do estado do sistema de acordo com o planejado (v. Seção 3.1.2), ou seja, o estado pode ser visto nas formas de *braket*, matriz-densidade e histograma, quer no final da simulação ou em qualquer ponto desta. Na janela de visualização, o usuário escolhe, através de botões, qual o tipo de visualização que deseja, facilitando, desta forma, sua leitura e interpretação do estado do sistema. As Figuras 19–21 abaixo ilustram as três formas de visualização do estado do sistema simulado.



Figura 19: Visualização do estado no simulador Zeno como ket

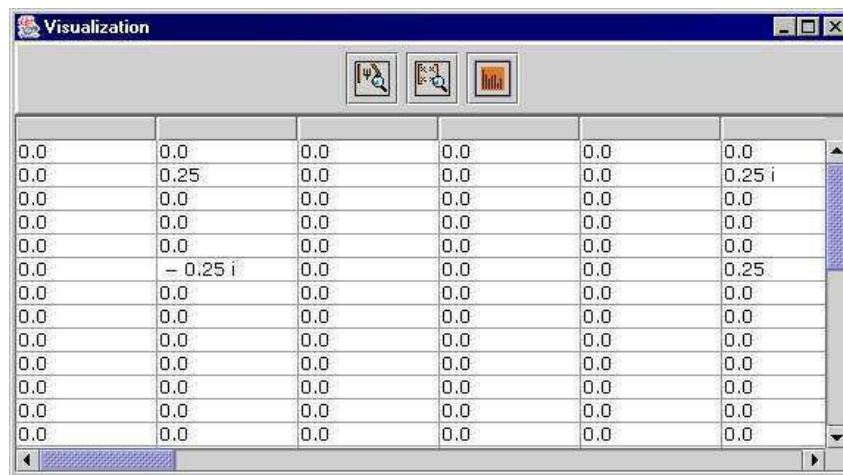


Figura 20: Visualização do estado no simulador Zeno como matriz-densidade

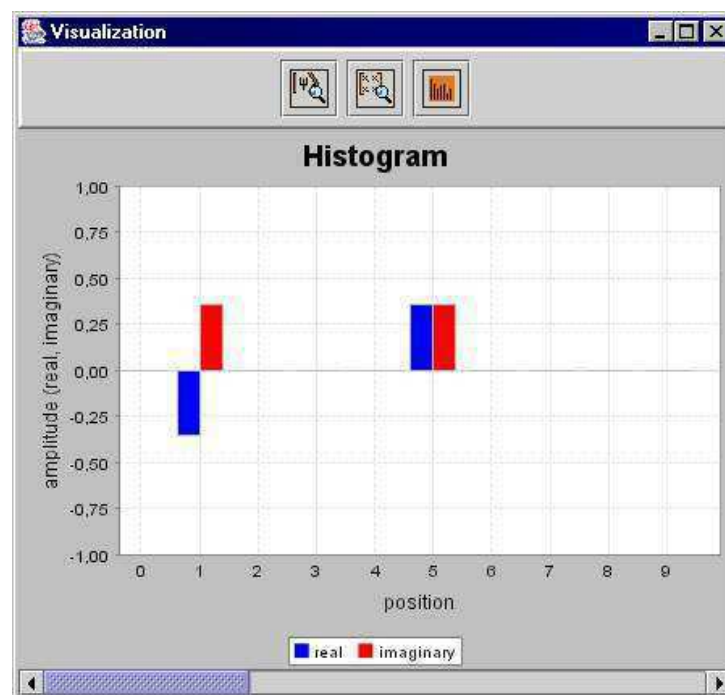


Figura 21: Visualização do estado no simulador Zeno como histograma

Edição: o simulador Zeno, segundo os requisitos planejados (v. Seção 3.1.2), disponibiliza ao usuário as operações de: copiar, recortar e colar portas e grupos de portas. Como, só é permitida uma porta por coluna, essas operações são equivalentes a copiar, recortar e colar colunas ou grupos de colunas, e assim foram implementadas. Para copiar e recortar, o usuário seleciona uma ou mais colunas e aperta o botão de copiar/recortar. Para colar, o usuário seleciona uma coluna do circuito e aperta no botão de colar.

Desfazer: conforme planejado, o simulador Zeno permite que o usuário retroceda ou desfça qualquer ação que ele possa ter realizado no processo de edição do circuito (*undo*). Para fazer isto, basta clicar no botão de desfazer ou selecionar a opção desfazer no menu apropriado (Edit), e o circuito voltará ao estado anterior.

5.1.3 Recursos de Documentação

Processo de desenvolvimento: segundo os requisitos definidos da Seção 3.1.3, toda a documentação sobre o processo de desenvolvimento da ferramenta, explicando desde a fase de obtenção de requisitos até a sua implementação, foi elaborada e disponibilizada na internet^[9]. Fazem parte desta documentação o diagrama arquitetural, o modelo conceitual e o diagrama de classes do simulador Zeno, conforme planejado.

Manual do usuário: um manual do usuário não será disponibilizado nesta primeira versão do simulador, porém, um breve guia de uso do simulador é disponibilizado no Apêndice A.

5.2 Testes de Correção e Desempenho

Nesta seção, serão apresentados os testes de correção e desempenho realizados com o simulador Zeno e uma comparação dos mesmos testes realizados com o simulador de circuitos quânticos jaQuzzi^[38]. Esses testes foram feitos apenas utilizando estados puros, porque o simulador jaQuzzi não trabalha com estados mistos. Com relação a estados mistos, foram feitos apenas testes de correção dos resultados apresentados pelo simulador Zeno. O simulador jaQuzzi foi escolhido em função de (i) sua disponibilidade para montagem e simulação de um circuito qualquer, (ii) ser simulador universal, (iii) permitir uma montagem fácil e flexível de portas e circuitos (manipulação direta), e (iv) ter sido implementado na mesma linguagem do simulador Zeno (Java). Uma descrição mais detalhadas deste simulador pode ser encontrada no trabalho de Schürmann^[38] e no relatório técnico^[10].

O hardware utilizado para a realização dos testes foi um PC com processador Intel Pentium II, com 348MHz e 192MB de RAM.

A estratégia de teste de corretude e desempenho adotada foi a seguinte:

1. Definir um conjunto de circuitos a serem testados
2. Montar e executar os circuitos em cada uma das ferramentas
3. Comparar tempo decorrido nos testes

4. Comparar resultados obtidos nos testes

Foram escolhidos quatro circuitos para o conjunto de testes: (i) o somador de números de dois bits, apresentado em Menscher^[34], que foi escolhido por ser um circuito simples e que mostra um circuito clássico sendo implementado por portas quânticas; (ii) o circuito da transformada de Fourier quântica (QFT) de três qubits, apresentado em Nielsen e Chuang^[36], escolhido por ser um circuito simples que apresenta um algoritmo inerentemente quântico de grande uso em outros algoritmos; (iii) o circuito QFT de seis qubits baseado no circuito, apresentado em Ekert et al.^[21], que é uma expansão do circuito anterior para um número maior de qubits, escolhido para se testar a simulação, em termos de tempo, principalmente, com circuitos maiores; (iv) e o circuito para calcular o autovalor, apresentado em Schürmann^[38] por se tratar de um circuito mais complexo e que implementa um algoritmo quântico eficiente para encontrar os autovalores de uma hamiltoniana local^[5].

5.2.1 Somador de Dois Bits

Este circuito soma dois números de dois bits cada, resultando em um número de 3 bits (9 portas). O circuito utiliza 7 qubits no total. O circuito é apresentado em Menscher^[34]. As Figuras 22 e 23 mostram a montagem deste somador nos simuladores Zeno e jaQuzzi.

simulador	duração (puro)	duração (misto)
Zeno	800ms	25s 56ms
jaQuzzi	~ 2s	—

Tabela 8: Tempo de execução do circuito somador de dois bits

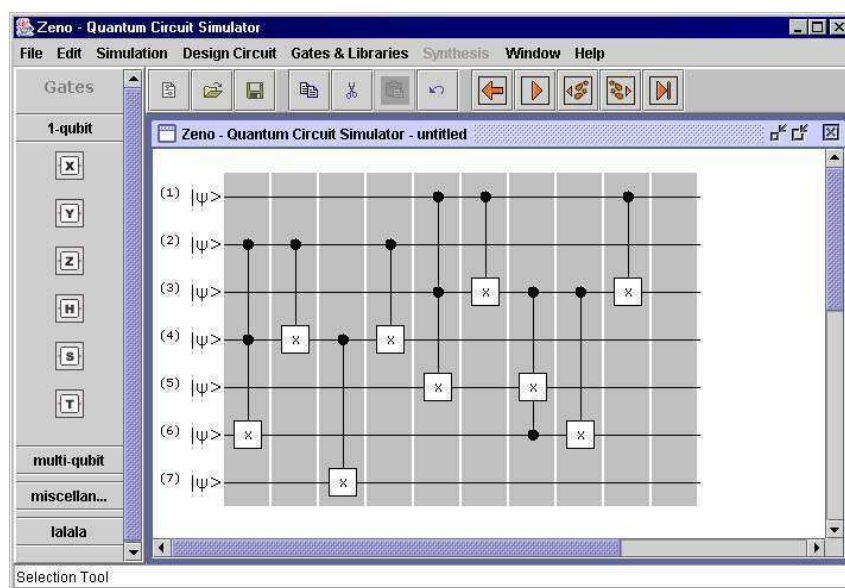


Figura 22: Montagem do somador de números de 2 bits no simulador Zeno

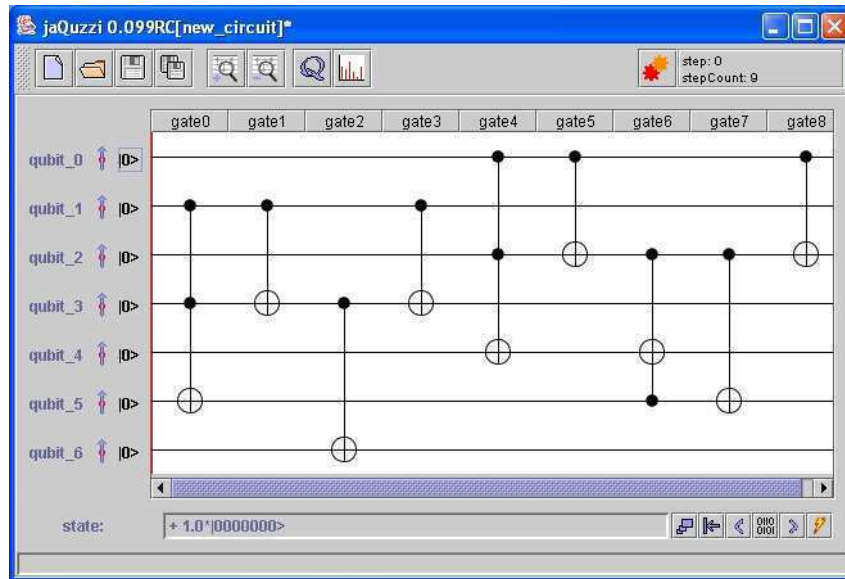


Figura 23: Montagem do somador de números de 2 bits no simulador jaQuzzi.

Percebe-se neste teste que os tempos de execução se equiparam nos dois simuladores. O circuito foi executado para todos os valores de entrada possíveis e válidos. Os resultados obtidos no simulador Zeno foram os seguintes:

Estado de saída	Problema de adição equivalente
$ 00\ 00\ 000\rangle$	$0 + 0 = 0$
$ 00\ 01\ 001\rangle$	$0 + 1 = 1$
$ 00\ 10\ 010\rangle$	$0 + 2 = 2$
$ 00\ 11\ 011\rangle$	$0 + 3 = 3$
$ 01\ 00\ 001\rangle$	$1 + 0 = 1$
$ 01\ 01\ 010\rangle$	$1 + 1 = 2$
$ 01\ 10\ 011\rangle$	$1 + 2 = 3$
$ 01\ 11\ 100\rangle$	$1 + 3 = 4$
$ 10\ 00\ 010\rangle$	$2 + 0 = 2$
$ 10\ 01\ 011\rangle$	$2 + 1 = 3$
$ 10\ 10\ 100\rangle$	$2 + 2 = 4$
$ 10\ 11\ 101\rangle$	$2 + 3 = 5$
$ 11\ 00\ 011\rangle$	$3 + 0 = 3$
$ 11\ 01\ 100\rangle$	$3 + 1 = 4$
$ 11\ 10\ 101\rangle$	$3 + 2 = 5$
$ 11\ 11\ 110\rangle$	$3 + 3 = 6$

Tabela 9: Resultados de simulação do circuito somador de dois bits

5.2.2 Transformada de Fourier Quântica (QFT)

Dois circuitos foram testados para este algoritmo, um de 3 qubits^[36] (6 portas) e outro de 6 qubits^[21] (21 portas).

As Figuras 24 e 25 mostram a montagem do circuito QFT para 3 qubits nos simuladores Zeno e jaQuzzi respectivamente.

simulador	duração (puro)	duração (misto)
Zeno	$< 1ms$	$10ms$
jaQuzzi	$< 1ms$	—

Tabela 10: Tempo de execução do circuito QFT para 3 qubits

simulador	duração (puro)	duração (misto)
Zeno	$600ms$	$5s$
jaQuzzi	$< 1s$	—

Tabela 11: Tempo de execução do circuito QFT para 6 qubits

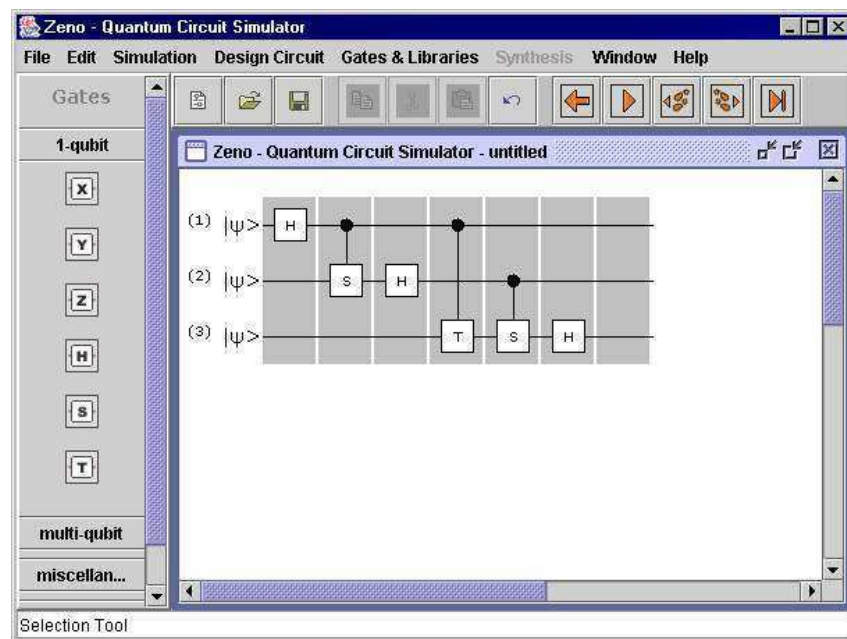


Figura 24: Circuito da QFT para 3 qubits no simulador Zeno.

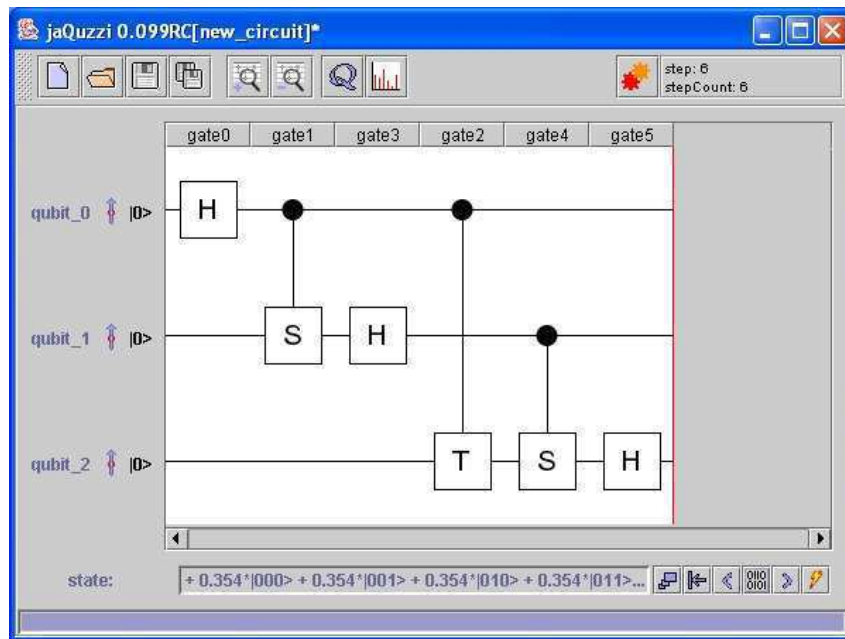


Figura 25: Circuito da QFT para 3 qubits no simulador jaQuzzi.

Nestes dois testes, o jaQuzzi também apresenta uma velocidade comparável à do Zeno.

As Figuras 26 e 27 mostram o resultado da simulação nos simuladores Zeno e jaQuzzi para a entrada $|000\rangle$ no circuito QFT para 3 qubits.

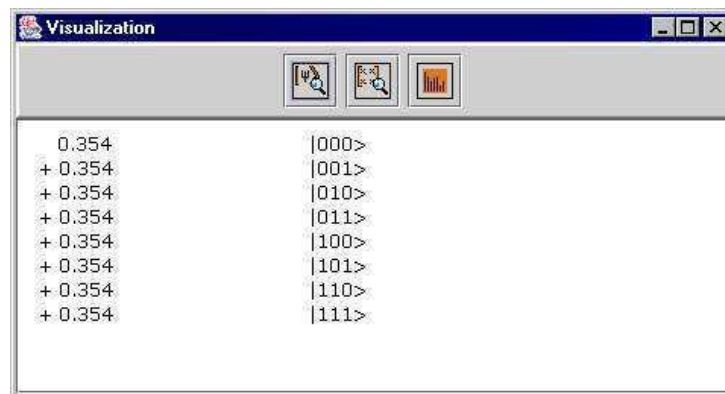


Figura 26: Resultado da simulação para a entrada $|000\rangle$ no circuito QFT para 3 qubits no simulador Zeno.

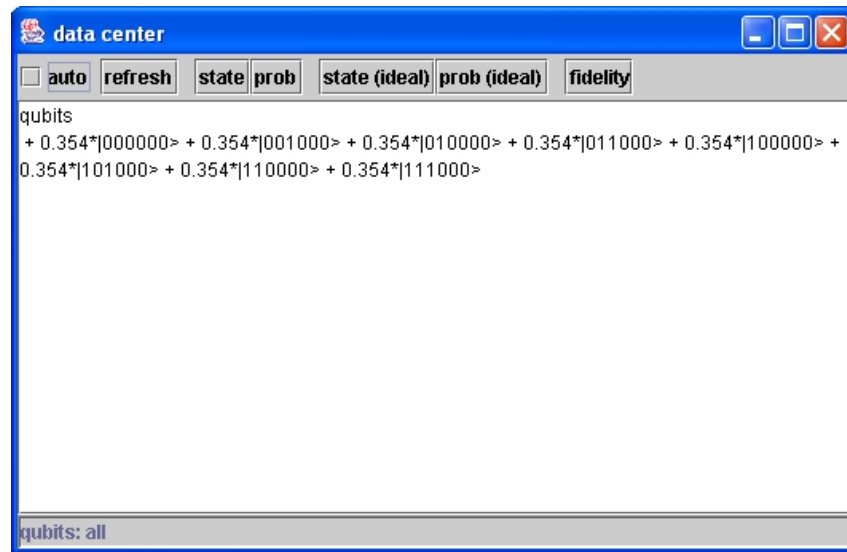


Figura 27: Resultado da simulação para a entrada $|000\rangle$ no circuito QFT para 3 qubits no simulador jaQuzzi.

A Figura 28 mostra o resultado da simulação para estados mistos, ou seja, em forma de matriz-densidade.

0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125

Figura 28: Resultado do mesmo circuito no simulador Zeno com estados mistos.

5.2.3 Circuito para Calcular o Autovalor

Este circuito calcula o autovalor de um operador quântico. O circuito (45 portas) pode ser encontrado em Schürmann^[38]. As Figuras 29 e 30 mostram a montagem do circuito nos simuladores Zeno e jaQuzzi respectivamente.

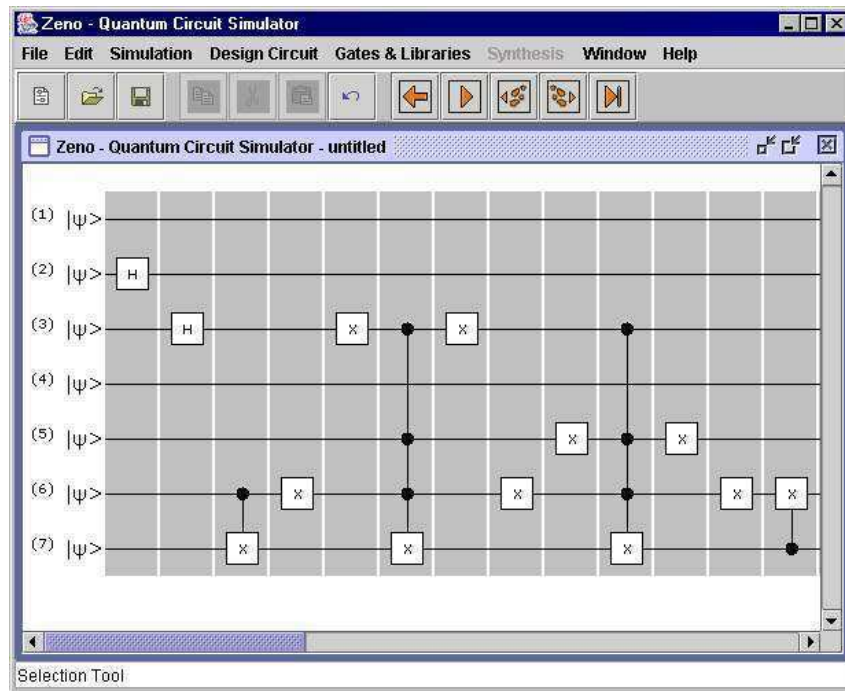


Figura 29: Circuito para calcular o autovalor de um operador unitário no simulador Zeno.

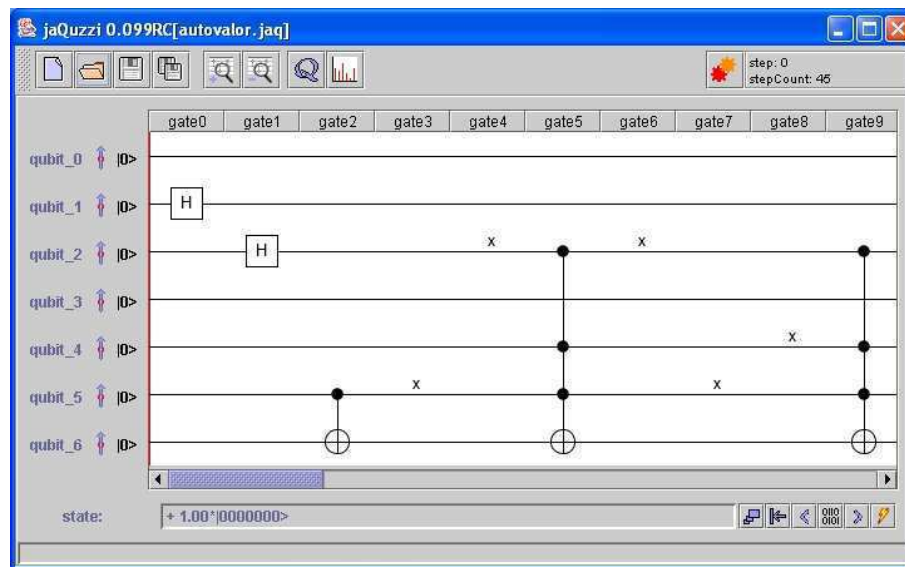


Figura 30: Circuito para calcular o autovalor de um operador unitário no simulador jaQuzzi.

simulador	duração (puro)	duração (misto)
Zeno	$\sim 2min$	$3min\ 50s$
jaQuzzi	$\sim 1s$	—

Tabela 12: Tempo de execução do circuito para calcular o autovalor de um operador quântico

Com este circuito, o simulador Zeno apresenta o pior desempenho, pois realiza a simulação do circuito na ordem dos minutos, enquanto que o jaQuzzi o faz em cerca de um segundo.

O resultado da simulação no simulador Zeno foi comparado com o resultado no simulador jaQuzzi e os valores foram iguais. As Figuras 31 e 32 mostram o resultado obtido nos simuladores Zeno e jaQuzzi para a simulação deste circuito.

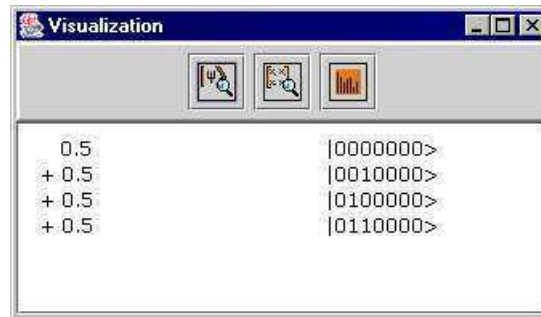


Figura 31: Resultado da simulação do circuito para cálculo de autovalor no simulador Zeno.

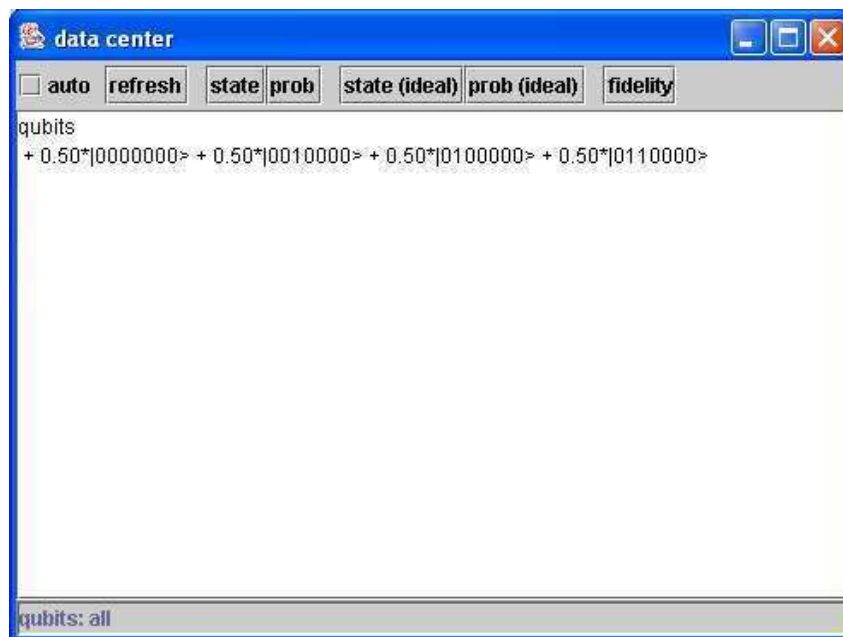


Figura 32: Resultado da simulação do circuito para cálculo de autovalor no simulador jaQuzzi.

5.2.4 Testes com a Porta de Medição

Para testar as portas de medição, o circuito QFT de 3 qubits foi adicionado de três portas de medição no final, uma porta em cada qubit. As Figuras 33 e 34 mostram a montagem deste circuito no simulador Zeno.

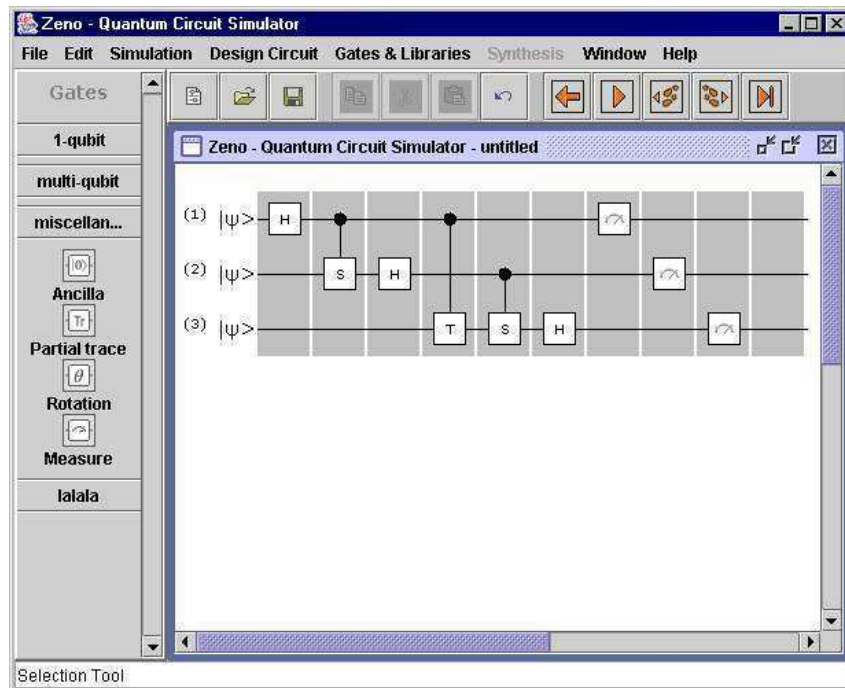


Figura 33: Circuito da QFT para 3 qubits com portas de medição nos simuladores Zeno (esquerda) e jaQuzzi.

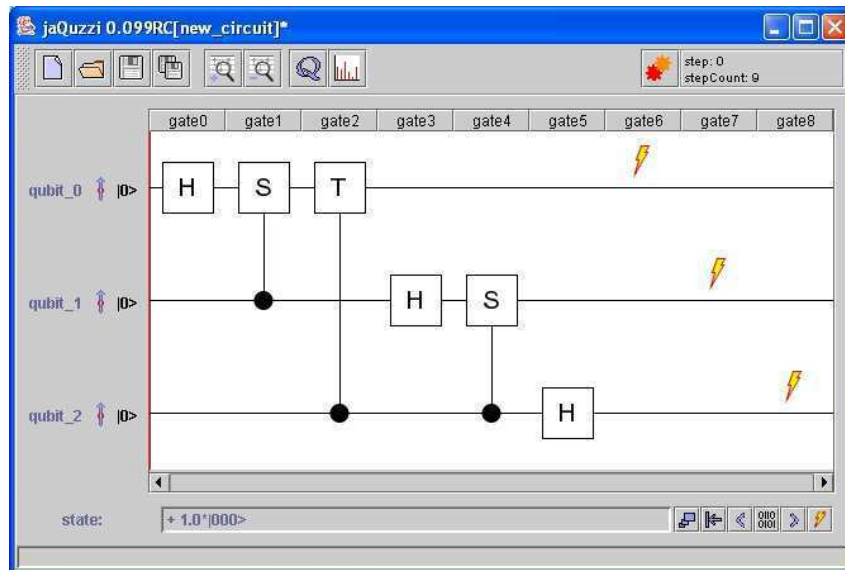


Figura 34: Circuito da QFT para 3 qubits com portas de medição nos simuladores Zeno (esquerda) e jaQuzzi.

Os resultados obtidos, em várias simulações do circuito com entradas $|000\rangle$ e $|111\rangle$, foram:

Entrada	Saídas	Entrada	Saídas
$ 000\rangle$	$ 000\rangle$	$ 111\rangle$	$ 000\rangle$
	$ 001\rangle$		$- 001\rangle$
	$ 010\rangle$		$-i 010\rangle$
	$ 011\rangle$		$i 011\rangle$
	$ 100\rangle$		$0.7071067811865476 - i0.7071067811865476 100\rangle$
	$ 101\rangle$		$-0.7071067811865476 + i0.7071067811865476 101\rangle$
	$ 110\rangle$		$-0.7071067811865476 - i0.7071067811865476 110\rangle$
	$ 111\rangle$		$0.7071067811865476 + i0.7071067811865476 111\rangle$

Tabela 13: Resultados de simulação do circuito QFT de 3 qubits com leitura de todos os qubits no final

5.3 Resultados dos Testes

Duas conclusões podem ser tiradas dos testes acima. A primeira é que o simulador Zeno está simulando os circuitos corretamente, ou seja, os resultados dos cálculos são os esperados. Segundo, o simulador Zeno, na maioria dos testes, está realizando as simulações em tempos equiparáveis aos do simulador jaQuzzi, no entanto, para o circuito que calcula o autovalor, o simulador Zeno apresenta um desempenho muito abaixo do desempenho do simulador jaQuzzi.

Quanto à simulação com estados mistos, os testes correram como esperado, ou seja, a simulação demorou mais e os estados de saída resultaram corretos.

6 Conclusão

Neste trabalho, foi planejado e implementado o simulador de circuitos quânticos Zeno, assim como foi apresentado o processo de concepção e desenvolvimento utilizado. Inicialmente, foi justificada a necessidade e a importância do desenvolvimento de uma nova ferramenta para projeto e simulação de circuitos quânticos. Em seguida, foram introduzidos os conceitos teóricos sobre Computação Quântica e Informação Quântica utilizados no simulador.

Na seqüência, foram apresentados os requisitos gerais desejáveis para um novo simulador quântico que expressam as necessidades dos usuários de um simulador de circuitos quânticos. Esses requisitos foram identificados (i) a partir dos aspectos teóricos explanados no Capítulo 2, (ii) através de um levantamento sobre ferramentas de simulação de circuitos quânticos^[10] e (iii) de necessidades do grupo local de estudo da Computação e Informação Quântica. Esses requisitos serviram como base para a realização da análise da tarefa, realizada logo após e abordada na descrição do processo de desenvolvimento.

Em seguida, foi descrito o processo de desenvolvimento do simulador Zeno, apresentando a análise de tarefas realizada e como ela serviu para o desenvolvimento tanto do módulo funcional quanto do módulo de interação. Também foram descritos os processos utilizados em cada um dos módulos para atender aos requisitos gerais desejáveis: no módulo funcional, um processo baseado no Processo Unificado, e no módulo de interação, um processo baseado na metodologia MEDITE.

Por fim, foram analisados os resultados obtidos com a implementação do simulador com relação a dois itens: recursos planejados *versus* recursos implementados, e correção e desempenho do simulador Zeno *versus* os resultados e o desempenho dos outros simuladores utilizados.

6.1 Objetivos Alcançados

O objetivo geral do trabalho de dissertação proposto foi atingido, ou seja, foi projetado e construído um simulador universal de circuitos quânticos. O simulador construído (Zeno) atende de forma total ou aproximada os três objetivos específicos apresentados na Seção 1.3, quais sejam:

1. oferece recursos funcionais que permitem modelar e simular algoritmos e canais quânticos utilizando o modelo de circuitos quânticos. Para isso, oferece a possibilidade de se trabalhar com estados puros e mistos, de oferecer conjuntos de portas universais, e portas de traço parcial e de *ancilla bit* (duas operações comumente

usadas em canais quânticos), de realizar medição projetiva em qualquer base além da base computacional, entre outras funcionalidades;

2. oferece recursos avançados de *manipulação direta* que permitem a edição e a montagem de circuitos de maneira fácil e intuitiva. Para tal, foi construída uma interface gráfica onde o usuário manipula as portas no circuito com a ajuda do mouse e do teclado;
3. disponibilize uma documentação apropriada tanto do processo de desenvolvimento quanto da ferramenta a ser construída. O desenvolvimento do simulador Zeno está documentado na presente dissertação e em página na internet^[9]. A documentação da ferramenta será provida com esta.

6.2 Contribuições

A principal contribuição deste trabalho é o simulador Zeno. O Zeno apresenta como diferencial o seguinte conjunto de características não encontrado em nenhum outro dos simuladores existentes:

1. Apresenta uma interface do usuário gráfica e de uso fácil e intuitivo.
2. Recursos de edição como recortar, copiar e colar.
3. Possibilidade de o usuário desfazer uma ação de edição do circuito.
4. Trabalha tanto com estados puros quanto mistos.
5. Permite realizar medições em qualquer base, além da base computacional.
6. Disponibiliza a operação de traço parcial e a porta de bit de ancilla.
7. Disponibiliza bibliotecas de portas para que o usuário possa reutilizar portas criadas por ele em vários circuitos.

Outra contribuição importante é o auxílio que o simulador Zeno deverá dar no ensino e na aprendizagem da Computação Quântica, fornecendo uma maneira de se testar rapidamente os conceitos aprendidos quanto a circuitos quânticos.

6.3 Perspectivas Futuras

Se os simuladores de circuitos quânticos continuarem seguindo a tendência evolutiva prevista no Capítulo 1, dentro em pouco tempo haverá simuladores de circuitos quânticos que ofereçam todos os recursos funcionais previstos neste trabalho, aliados a uma maior facilidade de uso, além de alguns recursos não previstos aqui.

Uma melhoria importante a ser implementada no futuro é a otimização dos algoritmos de simulação pois, em alguns casos, ele se apresenta muito lento em relação a outros simuladores, por exemplo, o circuito para calcular o autovalor (v. Seção 5.2.3).

Outra modificação a ser considerada é permitir que se simule a ocorrência de erros operacionais e de descoerência nos circuitos, além da implementação dos vários tipos de medição.

Um recurso interessante para ajudar no ensino e na aprendizagem da Computação Quântica, é que se implemente também algoritmos para gerar circuitos que computem (com aproximação definida pelo usuário) uma operação unitária arbitrária.

Com relação à avaliação da usabilidade da ferramenta, é desejável que sejam realizados testes de usabilidade e/ou alguma outra forma de avaliação da interface do usuário, como, por exemplo, avaliação heurística ou testes de conformidade segundo um padrão internacional (ISO 9241^[43]). A avaliação da usabilidade é um item importante no projeto de qualquer ferramenta de software, porém demanda um conhecimento que extrapola a formação do autor, e tempo e pessoal que excedem os recursos planejados.

Referências

- 1 Quantum computer simulator - user guide. *Senko Corporation*, 1999.
<http://www.senko-corp.co.jp/qcs/>.
- 2 Circuit Maker 2000 – The Virtual Electronics Lab, 2004.
<http://www.circuitmaker.com/>.
- 3 JHotDraw as Open-Source Project, 2004. <http://jhotdraw.sourceforge.net/>.
- 4 Zeno’s paradox, wikipedia, 2004. http://en.wikipedia.org/wiki/Zeno%27s_paradox.
- 5 D. S. Abrams e S. Lloyd. Quantum algorithm providing exponential speed increase for finding eigenvalues and eigenvectors. *Physical Review Letters*, 3(24), 1999.
- 6 A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, e H. Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457–3467, 1995.
- 7 A. Barenco, A. Ekert, A. Sanpera, e C. Machiavello. A short introduction to quantum computation. *La Recherche*.
- 8 C. Bennett. Demons, engines and the second law. *Scientific American*, November:108–116, 1987.
- 9 G. E. M. Cabral. Zeno – Simulador de Circuitos Quânticos – Documentação do Desenvolvimento, Rel. Tec. n^o dsc/001/04. *Universidade Federal de Campina Grande*, 2004.
- 10 G. E. M. Cabral e B. Lula Jr. O Estado da Arte em Ferramentas para Síntese e Simulação de Circuitos Quânticos, Rel. Tec. n^o dsc/003/03. *Universidade Federal de Campina Grande*, 2003.
- 11 M. Cass e P. E. Black. QCSim – quantum computer simulator (não publicado), 2002.
- 12 J. Cirac e P. Zoller. Quantum computations with cold trapped ions. *Physical Review Letters*, 74:4091–4094, 1995.
- 13 P. B. Cordeiro. Projeto e implementação do módulo TAME da ferramenta *iTAOS* para análise e modelagem da tarefa, 2003. Dissertação de Mestrado, COPIN, Universidade Federal de Campina Grande – UFCG.
- 14 Jean-Marie Dautelle. J.A.D.E. – Java Addition to Default Environment, 2003.
<http://jade.dautelle.com/>.

- 15 L. de Broglie. *Recherches sur la théorie des quanta (tr. Pesquisas sobre a teoria quântica)*. PhD thesis, Faculdade de Ciências, Universidade de Paris, 1924.
- 16 F. P. A. de Medeiros. Projeto e implementação do módulo TAOS-Graph da ferramenta *iTAOS* para análise e modelagem da tarefa, 2003. Dissertação de Mestrado, COPIN, Universidade Federal de Campina Grande – UFCG.
- 17 D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. Roy. Soc. Lond. A*, 400:97–117, 1985.
- 18 D. Deutsch. Quantum computational networks. *Proc. Roy. Soc. Lond. A*, 425:73–90, 1989.
- 19 S. W. Draper e D. A. Norman. Software engineering for user interfaces. *IEEE Trans. Softw. Eng. SE-11*, pages 252–258, 1985.
- 20 R. W. Ehrich e H. R. Hartson. Dms – and environment for dialogue management, setembro 1981.
- 21 A. Ekert, P. Hayden, e H. Inamori. Basic concepts in quantum computation. 2001. quant-ph/0011013.
- 22 Dodani et al. Separation of powers. *Byte*, março 1989.
- 23 R. P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6/7), 1982.
- 24 R. P. Feynman. Quantum mechanical computers. *Foundations of Physics*, 16(6), 1986.
- 25 E. Fredkin e T. Toffoli. Conservative logic. *International Journal of Theoretical Physics*, pages 219–253, 1982.
- 26 B. Lula Jr. G. E. M. Cabral, A. F. de Lima. Interpretando o Algoritmo de Deutsch no Interferômetro de Mach-Zehnder. *aceito para publicação na Revista Brasileira do Ensino de Física*, 2004.
- 27 N. A. Gershenfeld, I. L. Chuang, e M. Kubinec. Experimental implementation of fast quantum searching. *Physical Review Letters*, 80(15):3408–3411, 1998.
- 28 R. B. Griffiths. Notes on quantum information theory, 2004.
- 29 L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proc. 28th Annual ACM Symposium on Theory of Computing*. ACM, New York, 1996.
- 30 C. V. S. Guerrero. MEDITE, Uma metodologia orientada a modelos para concepção de interfaces ergonômicas, 2002. Dissertação de Mestrado, COPIN, Universidade Federal de Campina Grande – UFCG.
- 31 J. Jones e M. Mosca. Implementation of a quantum algorithm to solve deutsch’s problem on a nuclear magnetic resonance quantum computer. *Journal of Chemical Physics*, 109:1648–1653, 1998. quant-ph/9801027.

- 32 J. Jones, M. Mosca, e R. H. Hansen. Implementation of a quantum search algorithm on a quantum computer. *Nature*, 393:344–346, 1998. [quant-ph/9805069](https://arxiv.org/abs/quant-ph/9805069).
- 33 T. Meloche. The rational unified process. *The Menlo Institute LLC*, 2002. <http://www.menloinstitute.com/freestuff/whitepapers/rup.htm>.
- 34 D. P. Menscher. Modeling the Quantum Computer on the Classical Computer, 1997. Dissertação de Mestrado, Brigham Young University.
- 35 C. Monroe, D. Meekhof, B. King, W. Itano, e D. Wineland. Demonstration of a fundamental quantum logic gate. *Physical Review Letters*, 75(25):4714–4717, 1995.
- 36 M. A. Nielsen e I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
- 37 J. Preskill. *Lecture Notes for Physics 229: Quantum Information and Computation*. California Institute of Technology, 1998. <http://www.theory.caltech.edu/people/preskill/ph229/>.
- 38 F. Schürmann. Interactive Quantum Computation. 2000. Dissertação de Mestrado, State University of New York. <http://www.eng.buffalo.edu/~phygons/jaQuzzi/>.
- 39 B. Schwarzschild. Labs demonstrate logic gates for quantum computation. *Physics Today*, March:21–23, 1996.
- 40 P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe*, pages 124–134. IEEE Computer Society Press, 1994.
- 41 P. W. Shor. Fault-tolerant quantum computation. In *Proc. 37th Annual Symposium on Foundations of Computing*, pages 56–65. IEEE Computer Society Press, 1996.
- 42 A. Steane. Quantum computing, 1997. [quant-ph/9708022](https://arxiv.org/abs/quant-ph/9708022).
- 43 D. Travis. *Bluffer's Guide to ISO 9241*. User Focus, 2004. <http://www.userfocus.co.uk/articles/ISO9241.html>.
- 44 Q. Turchette, C. Hood, W. Lange, H. Mabuchi, e H. Kimble. Measurement of conditional phase shifts for quantum logic. *Physical Review Letters*, 75(25):4710–4713, 1995.
- 45 J. Vlissides, R. Helm, E. Gamma, e R. Johnson. *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison Wesley Pub., 1994.
- 46 J. Wallace. Quantum Computer Simulators – A Review. University of Exeter, 1999. <http://www.dcs.ex.ac.uk/~jwallace/simrevab.htm>.
- 47 C. P. Williams e S. H. Clearwater. *Explorations in Quantum Computing*. Telos, 1997.

APÊNDICE A – Guia de como Usar o Simulador Zeno

Neste apêndice será dada uma breve apresentação de como usar o simulador Zeno. São apresentadas as partes principais da janela do simulador, e então cada parte será explicada em mais detalhes, porém de forma sucinta.

A janela principal do simulador Zeno é constituída por quatro partes, como é mostrado na Figura 35. Estas quatro partes são detalhadas a seguir.

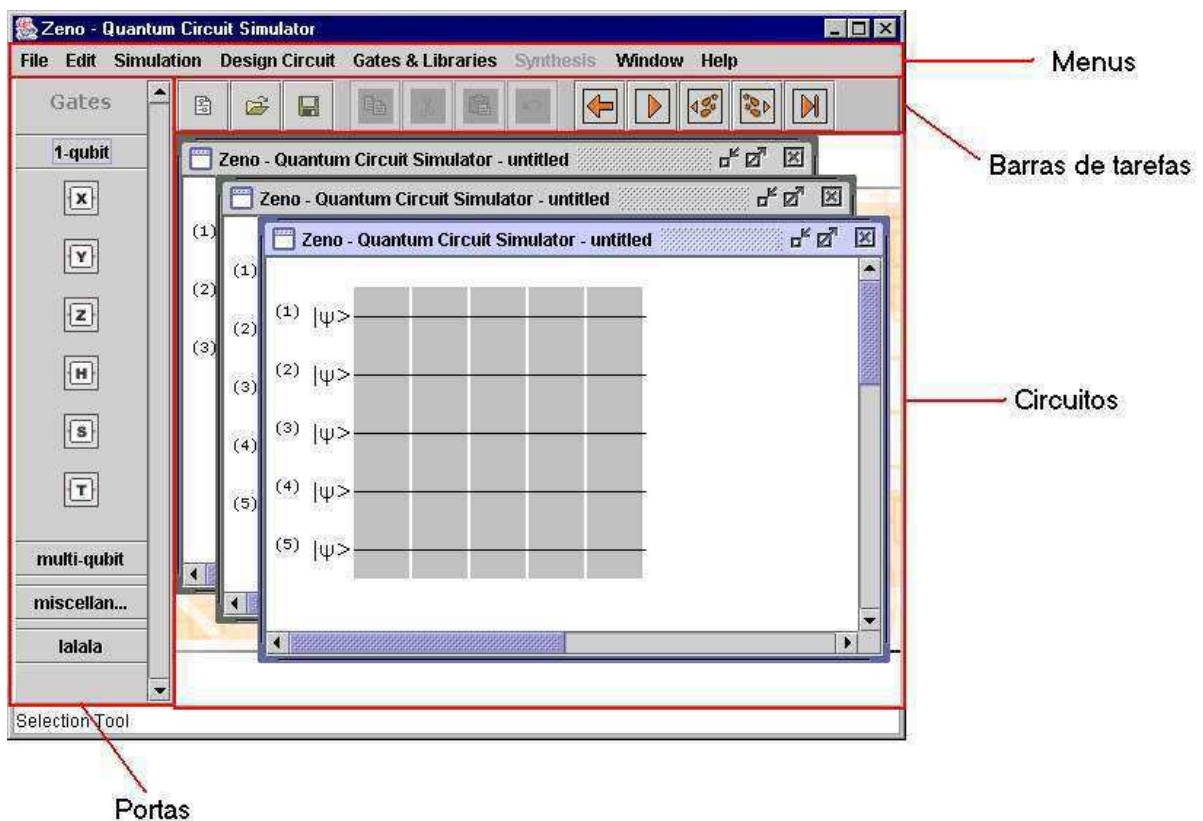


Figura 35: As partes da janela principal do simulador Zeno

A.1 Menus

Nesta seção, são descritos os menus do simulador Zeno.

O menu *File* (arquivo) é dedicado à configuração e gerenciamento da ferramenta e ao gerenciamento de circuitos. Este menu tem os seguintes itens:

- **New:** cria um novo circuito vazio. O usuário define as dimensões do circuito, assim como o tipo (puro ou misto)
- **Open:** abre um circuito salvo anteriormente
- **Save:** salva um circuito como um arquivo em disco
- **Print:** imprime o circuito ativo
- **Preferences:** configura a ferramenta
- **Close:** fecha o circuito ativo
- **Exit:** fecha a ferramenta

O menu *Edit* (edição) é dedicado aos recursos de edição do circuito como *copiar e colar*, e *desfazer*. Seus itens são os seguintes:

- **Undo:** desfaz a última edição realizada no circuito
- **Copy:** copia uma ou mais colunas do circuito
- **Cut:** “corta” (guarda cópia e remove) uma das colunas do circuito
- **Paste:** “cola” uma ou mais colunas anteriormente copiadas ou cortadas

O menu *Simulation* (simulação) contém funcionalidades relacionadas à simulação do circuito. Este menu contém os seguintes itens:

- **Normal simulation:** simula o circuito até a última coluna
- **Step-by-step simulation:** simula a próxima coluna ainda não simulada
- **To-cursor simulation:** simula o circuito até a coluna selecionada
- **Visualization:** mostra visualização do estado do sistema

O menu *Design Circuit* (projetar circuito) contém funcionalidades para a edição do circuito. Os itens deste menu são os seguintes:

- **Insert gate:** insere uma porta no circuito
- **Remove gate:** remove a porta selecionado do circuito
- **Edit gate:** edita as propriedades da porta

- **Insert qubit:** insere um novo qubit (linha) na posição selecionada
- **Remove qubit:** remove o qubit selecionado
- **Insert column:** insere uma nova coluna na posição selecionada
- **Remove column:** remove a coluna selecionada
- **Insert control:** insere um controle na porta selecionada
- **Remove control:** remove o controle selecionado
- **Group columns:** agrupa colunas em uma única coluna
- **ungroup columns:** desagrupa uma coluna agrupada em suas devidas colunas

O menu *Gates & Libraries* (portas e bibliotecas) é dedicado ao gerenciamento de bibliotecas de portas e das portas nessas bibliotecas. Os itens deste menu são:

- **Manage gates:** gerencia as portas nas bibliotecas (adiciona, remove, edita)
- **Manage libraries:** gerencia as bibliotecas (adiciona, remove, edita)

O menu *Window* (janela) é dedicado ao gerenciamento das janelas de circuitos e à barra de bibliotecas. Tem os seguintes itens:

- **Hide libraries:** esconde ou mostra a barra de bibliotecas
- **Window list:** mostra a lista dos circuitos abertos

O menu *Help* (ajuda) tem apenas um item que chama a caixa “sobre”, com informações básicas sobre o simulador Zeno.

A.2 Portas

A barra vertical à esquerda da janela do simulador Zeno mostra as bibliotecas existentes com as portas disponíveis em cada uma. Clicando sobre o botão de uma biblioteca, serão mostradas as portas pertencentes àquela biblioteca, enquanto que as portas das outras bibliotecas serão escondidas.

Para colocar uma porta no circuito, o usuário deve clicar sobre a porta desejada e depois clicar na linha e coluna do circuito em que quer a porta. Se for uma posição válida, então a porta será colocada no circuito.

A.3 Barra de Tarefas

A barra de tarefas contém algumas funções muito usadas pelo usuário do simulador. São as seguintes funcionalidades:

- **Novo circuito:** cria um novo circuito vazio
- **Abrir circuito:** abra um circuito anteriormente salvo em arquivo
- **Salvar circuito:** salva o circuito ativo
- **Salvar como...:** salva o circuito com um novo nome
- **Copiar:** copia a coluna ou o grupo de colunas selecionado
- **Recortar:** “corta” a coluna ou o grupo de colunas selecionado
- **Colar:** “cola” a coluna ou grupo de colunas anteriormente copiado ou cortado na posição selecionada
- **Desfazer:** desfaz a última edição no circuito
- **Simulação normal:** realiza a simulação até a última coluna
- **Simulação passo-a-passo:** simula uma coluna do circuito
- **Simulação passo-a-passo para trás:** simula uma coluna do circuito da direita para a esquerda
- **Simulação até o cursor:** simula até a coluna selecionada

A.4 Circuitos

Nesta parte da janela são mostrados os circuitos abertos pelo usuário. Em cada janela de circuito, é mostrado o estado inicial — com os qubits representados por $|\psi\rangle$ (para estados puros) ou $[::]$ (para estados mistos)—, e o circuito em si.

Clicando com o botão direito do mouse sobre um qubit do estado de entrada, surgirá um menu com as seguintes opções:

- **Edit qubit:** edita o valor do qubit
- **Insert qubit:** insere um qubit na posição clicada
- **Remove qubit:** remove o qubit selecionado

Clicando com o botão direito do mouse sobre um ponto vazio do circuito, surgirá um menu com as seguintes opções:

- **Insert gate:** insere uma port ana posição selecionada

- **Insert control:** insere um controle na posição selecionada
- **Insert column:** insere uma nova coluna no circuito na posição selecionada
- **Remove column:** remove a coluna ou o grupo de colunas selecionado
- **Group columns:** agrupa as colunas selecionadas
- **Ungroup column:** desagrupa uma coluna agrupada em suas componentes, se a coluna selecionada for uma coluna agrupada

Clicando com o botão direito do mouse sobre uma porta, surgirá um menu com as seguintes opções:

- **Edit gate:** edita as propriedades da porta
- **Remove gate:** remove a porta selecionada do circuito

Clicando com o botão direito do mouse sobre um controle, surgirá um menu com as seguintes opções:

- **Remove control:** remove o controle selecionado do circuito