Quantum Value Gate Simulator

by Xin Chen

Advisor: Prof. Chris Pollett

Introduction

- Simulates a quantum circuit that performs the function of a classical value gate
- Spalek's Algorithm
  - Robert Spalek
  - "Quantum Circuits with Unbounded Fan-out" STACS 2003 v2, 2003
- Supports ways of experiments with error models applied to the base gates
- Uses polynomial space and time

Value Gate

- Value gate:

  $$T_m(a_1, a_2, \ldots, a_n) =_{def} [\sum_{i=1}^{n} a_i = m]$$

- Threshold gate:

  $$T_m(a_1, a_2, \ldots, a_n) =_{def} [\sum_{i=1}^{n} a_i \geq m]$$

- Application of Threshold circuits

How does Quantum Mechanics work - Qubit

- Qubit
  - a qubit is a unit vector in a two-dimensional complex vector space
  - an arbitrary state vector in the state space

$$|\psi\rangle = a|0\rangle + b|1\rangle$$

where a and b are complex numbers, and $|a|^2 + |b|^2 = 1$, $|0\rangle$ and $|1\rangle$ are known as computational basis, and form an orthonormal basis for this vector space

How does Quantum Mechanics work - Evolution

- Unitary transform

$$|\psi'\rangle = U|\psi\rangle$$

where U is a unitary matrix, that is, $U^\dagger U = I$

How does Quantum Mechanics work - Measurement

- $\{M_m\}$
- Immediately before the measurement the probability that result *m* occurs is:

$$p(m) = \langle\psi|M_m^{\dagger} M_m|\psi\rangle$$

- $$\frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^{\dagger} M_m|\psi\rangle}}$$
- $\sum_m M_m^{\dagger} M_m = I$

How does Quantum Mechanics work – Composite Systems(1)

- Tensor Product (Kronecker Product)

The tensor product of two vector spaces *V* and *W*, denoted $V \otimes W$, is a way of creating a new vector space analogous to multiplication of integers.

$$\begin{bmatrix} a \\ b \end{bmatrix} \otimes \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} a \times \begin{bmatrix} c \\ d \end{bmatrix} \\ b \times \begin{bmatrix} c \\ d \end{bmatrix} \end{bmatrix} = \begin{bmatrix} ac \\ ad \\ bc \\ bd \end{bmatrix}$$

How does Quantum Mechanics work – Composite Systems(2)

- State space of a composite physical system is the tensor product of the state spaces of the component physical systems.

$$|\psi_1\rangle \otimes |\psi_2\rangle \otimes \ldots \otimes |\psi_n\rangle$$

Quantum Circuit

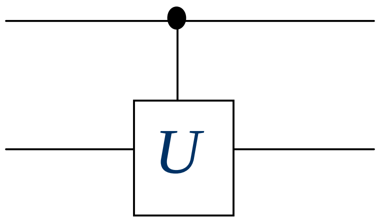
- Single qubit operation

$$H = 1/\sqrt{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

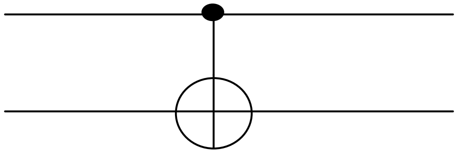
$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Quantum Circuit (Cont.)

- Controlled *U* operation



- Controlled *NOT* operation



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Spalek's Algorithm – Quantum Fan-out Operation

- Classical fan-out operation:

  $$|s\rangle|0\rangle^{\otimes n} \rightarrow |s\rangle^{\otimes n+1}$$

- Define a quantum fan-out operation on source qubit $|s\rangle$ and *n* target qubits $|t_k\rangle$ performs:

  $$|s\rangle \bigotimes_{K=1}^{n} |t_k\rangle \rightarrow |s\rangle \bigotimes_{K=1}^{n} |t_k \oplus s\rangle$$

Spalek's Algorithm – Parallelization Method

1. Apply the fan-out operation on a qubit to copy the state.
2. Apply each phase shift on a distinct "copy".
3. Apply the fan-out operation again, and clear the ancilla qubits.

Spalek's Algorithm – Quantum Hadamard Transform

- Quantum Fourier Transform (QFT)

$$y_k \equiv 1/\sqrt{N} \sum_{j=0}^{N-1} e^{2\pi ijk/N} x_i$$

transform a set of N complex numbers $x_0, \ldots, x_{N-1}$
into a set of complex numbers $y_0, \ldots, y_{N-1}$

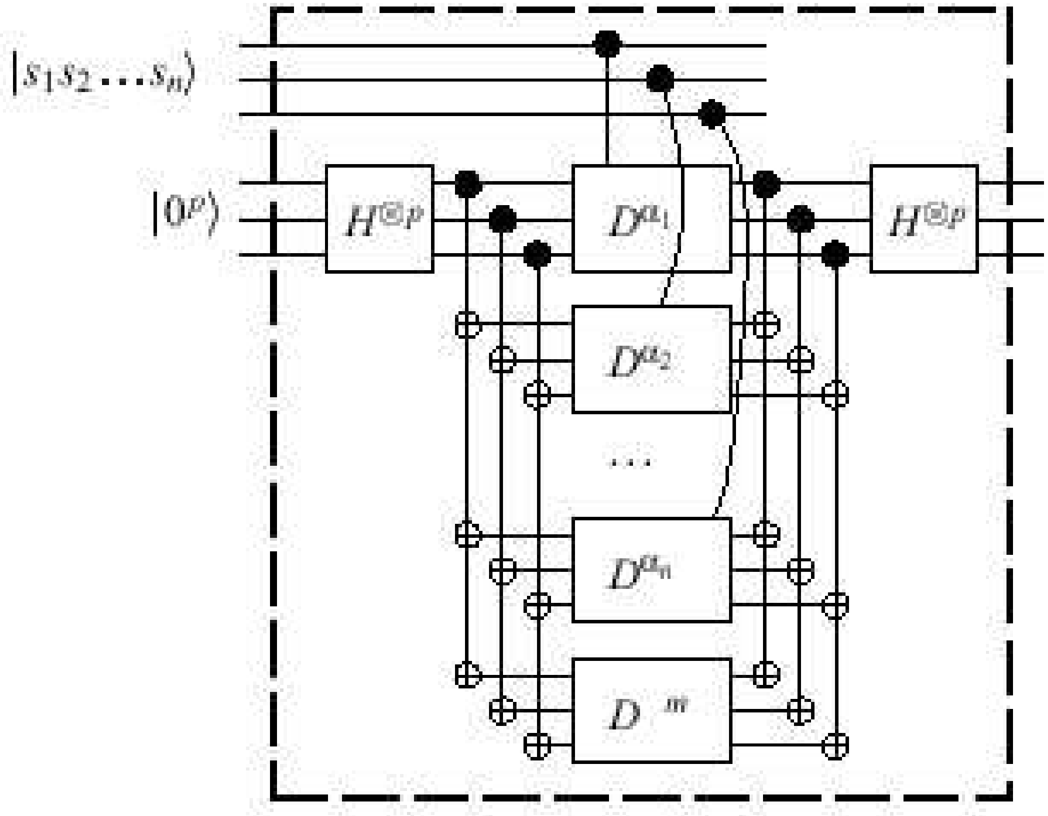
- Quantum Hadamard Transform
- A property of the Hadamard Transform:

$$H_n = H^{\otimes n}$$

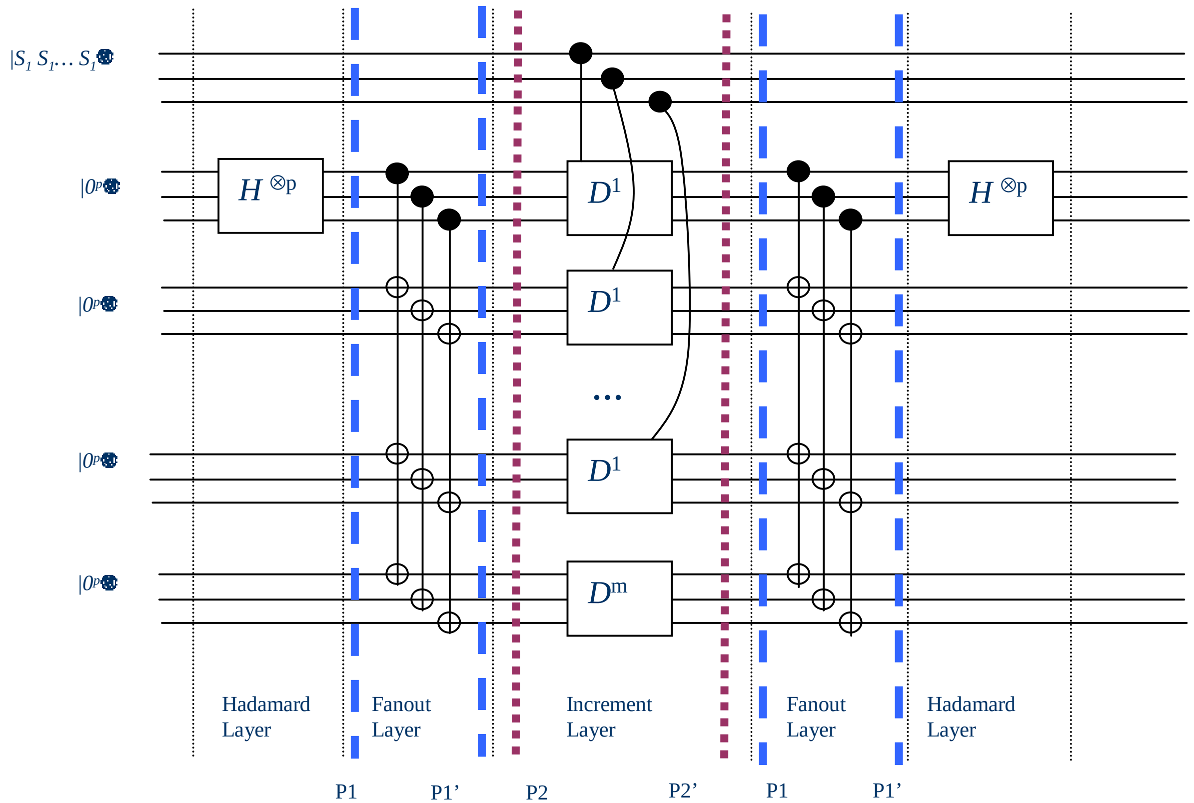
Spalek's Algorithm – Increment Operation

- An increment operation P on n qubits is an operation mapping each computational basis state $|x\rangle$ to $|x+1 \bmod 2^n\rangle$
- $D = FPF^\dagger$
- $R_z(\theta) = |0\rangle\langle 0| + e^{\theta i}|1\rangle\langle 1$
- $D_k = R_z(\pi / 2^{n-k}) \otimes D_{k-1}$

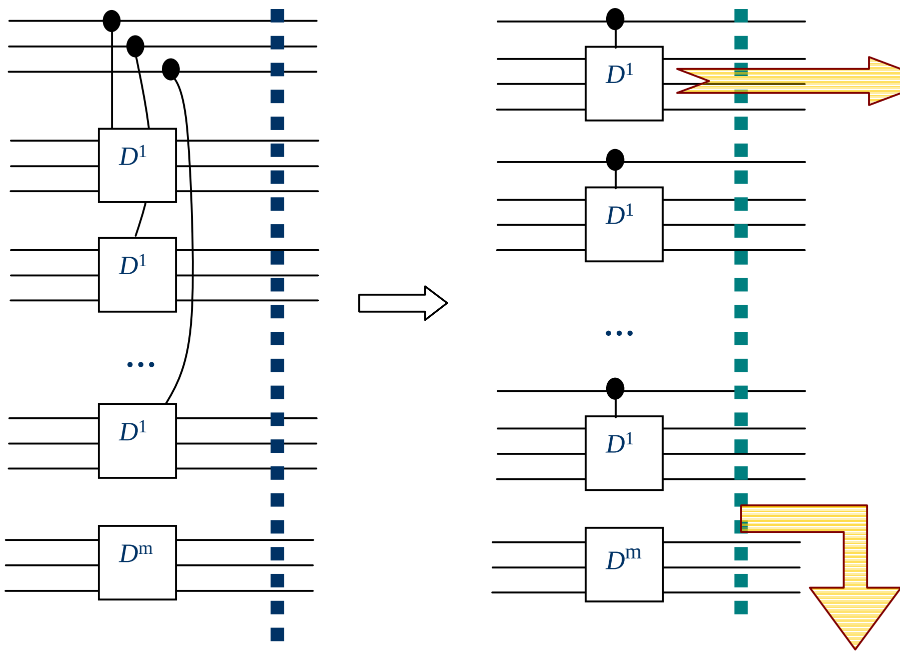
Spalek's Algorithm – Quantum Circuit for Value Gate



Design and Implementation – Matrix Representation



Design and Implementation – Matrix for the Increment Layer



$$\left[\begin{array}{cccccccc} 1 & 0 & \dots & \dots & 0 & \dots & \dots & 0 \\ 0 & 1 & \dots & \dots & 0 & \dots & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 & \dots & \dots & 0 \\ 0 & \dots & \dots & \dots & 0 & \dots & \dots & 0 \\ 0 & \dots & \dots & 1 & 0 & \dots & \dots & 0 \\ 0 & \dots & \dots & 0 & & & & \\ 0 & \dots & \dots & 0 & & d & & \\ 0 & \dots & \dots & 0 & & & & \\ 0 & \dots & \dots & 0 & & & & \\ 0 & \dots & \dots & 0 & & & & \end{array}\right] \begin{array}{l} \left.\begin{array}{c} \\ \\ \\ \\ \\ \end{array}\right\} 2^n \\ \left.\begin{array}{c} \\ \\ \\ \\ \\ \end{array}\right\} 2^n \end{array}$$

$$I\_layer = \underbrace{c\_d \otimes \dots \otimes c\_d \otimes d}_{n}$$

$$\begin{cases} P[x_i][y_i] = 1 & \text{when } x_i \in x,\ y_i \in y,\ i \in Z, \text{ and } 1 \le i \le n \\ P[x_i][j] = 0 & \text{when } x_i \in x,\ y_i \in y,\ j \ne y_i,\ i, j \in Z, \\ & \text{and } 1 \le i, j \le n \end{cases}$$

Design and Implementation – Matrix for the Entire Circuit

- $M = H\_layer \cdot P1 \cdot {}'F\_layer \cdot P1 \cdot P2'$
  $\cdot\ I\_layer \cdot P2 \cdot P1' \cdot F\_layer \cdot P1$
  $\cdot\ H\_layer$

Design and Implementation – A Naïve Implementation

- Space complexity:

  $O(2^{nlogn})$

- Time complexity:

  $O(2^{nlogn})$

Design and Implementation – An Efficient Implementation(1)

- For the result matrix *M*, we are only interested in one element
- In each matrix that represents a layer, there are at most $O(n)$ non-zero elements in each row/column
- Every row/column in any layer matrix can be computed in polynomial time

Design and Implementation – An Efficient Implementation(2)

- Complexity analysis
  - Creating each row: $O(n)$
  - Finding the $(S \cdot 2^{(n+1)\cdot p})$'th element: $O(n)$
  - Finding all h_x: $O(n^2)$
  - Total: $O(n^2) + O(n) = O(n^2)$

Error Scheme

- Adding error ε
  - $R_z(\theta + \varepsilon) = |0\rangle\langle 0| + (\cos(\theta + \varepsilon) + i\sin(\theta + \varepsilon))\,|1\rangle\langle 1|$
  - $R_z(\theta + \varepsilon') = |0\rangle\langle 0| + (\cos(\theta + \varepsilon') + i\sin(\theta + \varepsilon'))\,|1\rangle\langle 1|$

    where ε' is between - ε and ε

Test Case 1 (1)

- Parameters
  - String: Arbitrary 0/1 string with arbitrary length
  - Threshold:
    - Equals to the number of bits that are ‘on’
    - Not equals to the number of bits that are ‘on’

Test Case 1 (2)

- Sample results for test case 1

Input string	Threshold value	Error	Output
01110111010000000	7	0	1
01110111010000000	20	0	0
10011000001111	7	0	1
10011000001111	2	0	0
10000111000	4	0	1
10000111000	1	0	0
101	2	0	1
101	1	0	0
1	1	0	1
1	0	0	0

Tests Case 2 (1)

- Parameters
  - String: 0/1 string
  - Threshold
  - Error
  - Same Rotation Error: true/false

Test Case 2 (2)

- Sample results for test case 2

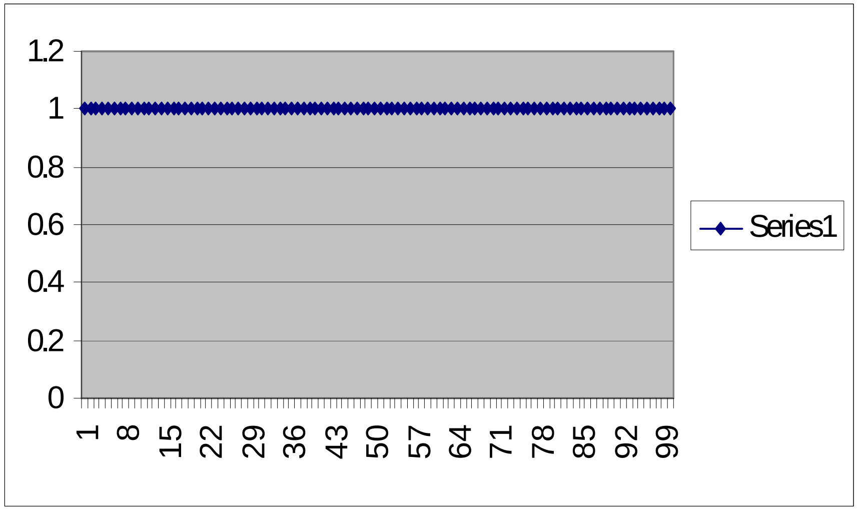


Figure 6- 1 Test result for test 2.1

- *String:* 10000111
- *Threshold:* 4
- *Error:* 0 to 1
- *Same Rotation Error:* true

Test Case 2 (3)

- Sample results for test case 2

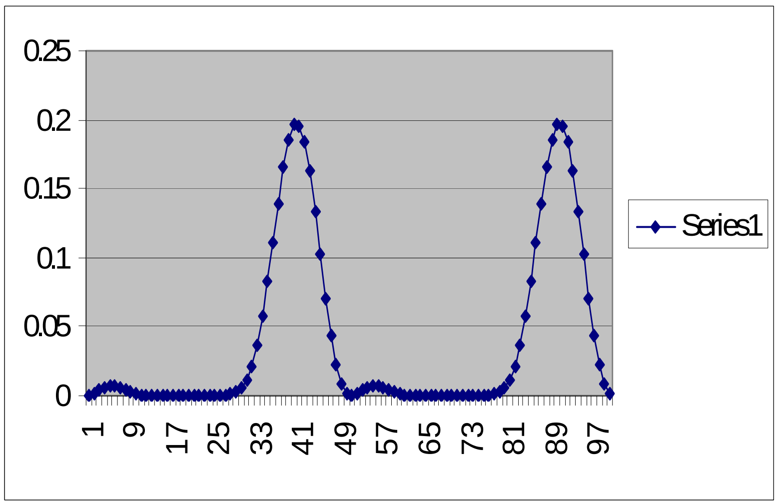


Figure 6- 2 Test result for test 2.2

- *String:* 10000111
- *Threshold:* 5
- *Error:* 0 to 12.56663706143 (around 4*PI)
- *Same Rotation Error:* true

Test Case 2 (4)

- Sample results for test case 2

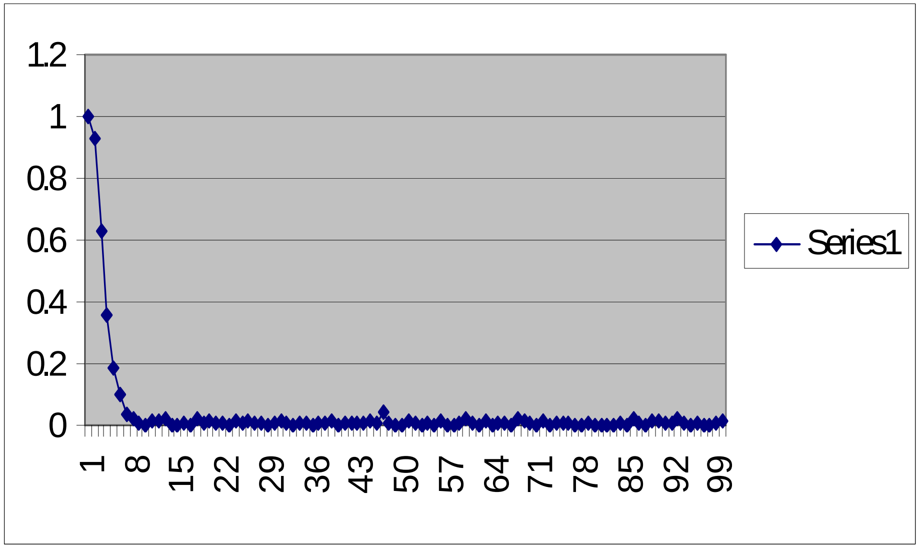


Figure 6-15 Test results on HP workstation

- *String:* 1111111111000000000011111111110000000000 1111111111
- *Threshold:* 30
- *Error:* 0 to 1
- *Same Rotation Error:* false

Conclusion

- Uses polynomial space and time
- Supports ways of experiments with error models applied to the rotation operator
- Tests on input up to 50 qubits

Thank You