Quantum Counters

Smita Krishnaswamy
Igor L. Markov
John P. Hayes

Contents

- Motivation
- Background
- Previous work
- Basic Circuit
- Good Parameters
- Improved Circuits
- Current Work /Conclusions

BACKGROUND

Motivation

- Goal of QC thus far has been to solve problems faster than classical computing
- Deutsch's algorithm first example of algorithm with faster quantum algorithm
- Our goal is to obtain exponential memory improvement for a specific problem
  - We use sequential circuits to achieve this

Sequential Circuits

- Sequential circuits contain a combinational portion and a memory portion
- Combinational portion is re-used and therefore usually simpler
- Sequential circuits are modeled by finite automata

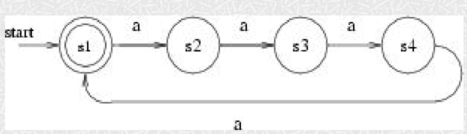
Finite Automata

- *5*-tuple $\{Q, \Sigma, \delta, q_0, Q_{acc}\}$
- Q=set of states
- Σ=input alphabet
- q_0=starting state
- Q_{acc}=set of accepting states
- δ = transition function

$$\delta : Q \times \Sigma \to Q$$

More Finite Automata

- The memory portion stores state info
- An FSA for a counter that counts to *4*:

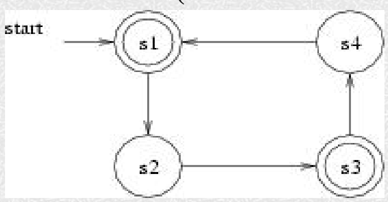


Quantum Finite Automata

- RFA: Reversible finite automata i.e. only one arrow going into each state
- A QFA is a reversible finite automata that transitions between quantum states
- Q=the vector space of state vectors
- Q_{acc}=the accepting subspace with an operator P_{acc} that projects onto it
- δ is a unitary matrix

Example

- RFA for *2*-counter (can do it in 2 states)



- RFA's cannot recognize $E=\{a^j|j=2k+3\}$

PREVIOUS WORK

Prime Counter

- For p prime let language $L_p = \{a^j : p \mid j\}$
- Any deterministic FA recognizing L_p takes at least p states
- Ambainis and Freivalds [1] show that a QFA needs only *O(log p)* states
- *O(log p)* states requires only *O(log log p)* qubits. This is an exponential decrease

QFA for L_p

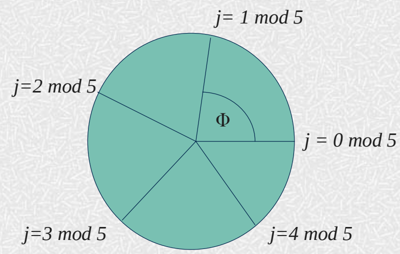
- Set of states : $Q = \{|0\rangle,|1\rangle\}$
- Starting state: $q_0=|0\rangle$
- Set of accepting states: $Q_{acc} = span(|0\rangle)$
- Next state function δ:

$$\begin{bmatrix} \cos(\phi) & i\sin(\phi) \\ i\sin(\phi) & \cos(\phi) \end{bmatrix}$$

Counter QFA

- Pick a rotation angle $\Phi=2\pi k/p,\ 0<k<p$
- For input a^j , rotate qubit j times by Φ
- If j is a multiple of p then the state is definitely $|0>$, else we have $|1>$ with probability $\cos^2(\Phi)$
- Want to pick a set of k's that increases the probability of obtaining state $|1>$ for every j

1-qubit Counter for $p=5$



Sequence of QFAs

- This QFA rejects any x not in L_p with varying probability of error ranging from 0 to 1
- Therefore any one of these QFAs is not enough
- We can pick a sequence of $8\ ln\ p$ QFA'S with different values for k where $\Phi = 2\ \pi\ k/\ p$
- The values for k can be picked such that the probability of error is always less than $7/8$

Proof Sketch

- At least half of all of the k's that we consider reject any given a^j not in L_p with probability at least ½
- There is a sequence of length $8\ ln\ p$ that ¼ of all elements reject every a^j not in L_p with probability ½ (This follows from Chernoff Bounds)

Problems To Address

- No explicit circuit construction given
- No explicit description of the sequence of angle parameters given
- Loose error estimate

CIRCUIT IMPLEMENTATION

Quantum Circuits

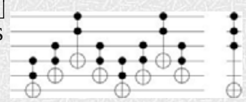
- Quantum operators are unitary matrices
- A larger matrix is broken into a matrix or tensor product of *2x2* matrices (*1*-qubit gates)
- Gate library: {R_x, R_y, R_z ,C-NOT, NOT}
- R_x (Φ)= $\begin{bmatrix} \cos(\phi) & i\sin(\phi) \\ i\sin(\phi) & \cos(\phi) \end{bmatrix}$

Gate Library

- R_y (Φ)= $\begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix}$
- R_z (Φ)= $\begin{bmatrix} e^{i\phi} & 0 \\ 0 & e^{-i\phi} \end{bmatrix}$
- NOT = $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

Gate Library

- C-NOT = $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
- K-controlled Rotations

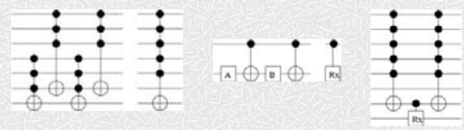


- H=$R_y(\pi/4)$

Controlled Rotations

Barenco et. al. [2] give construction for *k*-controlled rotations from basic gates

They are constructed from K-controlled NOT gates and *1*-controlled rotation gates



Implementation

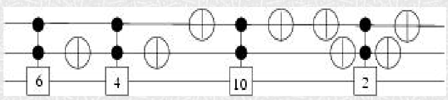
- The QFAs cannot be implemented separately without wasting space
  - Need *O(log p)* qubits for this
- Can implement as a block-diagonal matrix
- Each block on the diagonal is an R_x
- Can be implemented with controlled rotations

Block-Diagonal Matrix

$$\begin{bmatrix} R_x(\frac{2\pi k_1}{p}) & 0 & 0 & 0 \\ 0 & R_x(\frac{2\pi k_2}{p}) & 0 & 0 \\ 0 & 0 & R_x(\frac{2\pi k_3}{p}) & 0 \\ 0 & 0 & 0 & R_x(\frac{2\pi k_4}{p}) \end{bmatrix}$$

Basic Circuit

Each block diagonal corresponds to one
k-controlled rotation gate
Example:



GOOD PARAMETERS

Error Probability

- The probability of error for this circuit is:

$$P_{err} = \max_{1 \le j < p} \frac{1}{n} \sum_{i=1}^{n} \cos^2\left(\frac{2\pi k_i j}{p}\right)$$

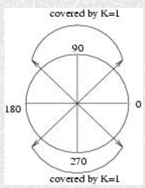
- The expression is the sum over the error contribution of each k_i
- Try to minimize the maximum error for any value of j<p

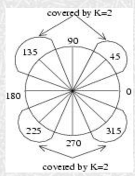
Picking parameters:Observations

- **Theorem**: No parameter set can have probability of error <½
- **Proof**:
  - For a given *k* the average probability of error over all *j*'s is ½
  - A sequence of k's has the same average probability of error
  - Therefore P_{err} which is the max of all of these has to be > ½

Rejection Patterns

- Each parameter is “good” for a ½ of the j’s and they occur in a specific pattern



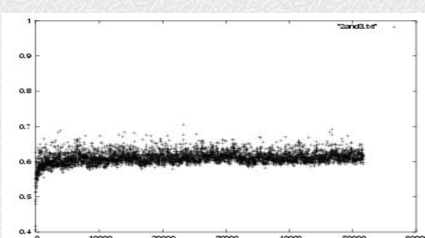


Greedy Selection of Parameters

- Try to cover as much new area as possible
- Continue this process until all parameters are rejected with a certain probability
- Obtain a set of parameters that follow the sequence $m^i l^j$ where m and l are constants
- Use mutually prime values of m and l to avoid repetition
  - Usually $m=2$ and $l=3$

Asymptotic Behavior

- These params give low P_{err} for all p’s



Estimating Error Bounds : Idea

- Discretize the cosine expression
- If $\sin^2(\varphi)> ½$ regard it as 1/2
  If $\sin^2(\varphi)< ½$ regard it as 0
- The area *covered* by a parameter k : the portion of the unit circle where $\sin^2(2\pi k/p) > ½$
- For the parameters $m^i l^j$ can get recursive expressions for which areas are covered by n of the k’s
- If n was half the total number of k’s.The probability lower bounded by $(1/2)(1/2)=(1/4)$

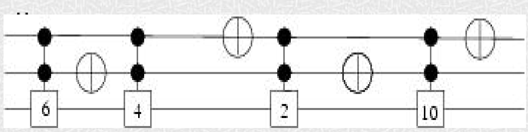
IMPROVED CIRCUITS

Circuit Complexity

- Basic block-diagonal circuit: too many gates
- There are only *O(log log p)* qubits where as there are *O(log p)* gates
- Different circuit decomposition may yield better results
  - Some reductions are possible

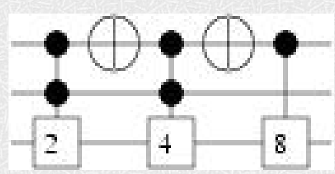
Reductions

- Order the parameters such that their controls are in Gray Code order
- Only one C-NOT is required between any set of C-NOTS



Reducing Control Bits

- Can use only *log p* different rotations
- We apply them with fewer control bits by using binary addition



Other Techniques

- Random circuit simulations
  - Pick a set of *k*-controlled circuits repeatedly
  - Save the best

Prime	19	53	101	241	307	521	1021	16381	65521	316691
$P[err]$	.508	.527	.549	.552	.558	.563	.575	.607	.624	.637

Table 3: Best $P[err]$ observed in 100 independent circuit simulations for each prime.

Greedy Simulation

- Pick gates one by one first go through controls then through rotations, p^2 possibilities
- Continue this while probability decreases
- Order does not matter

Diagonalization

$$\begin{bmatrix} H & & & \\ & H & & \\ & & H & \\ & & & H \end{bmatrix} \begin{bmatrix} R_z(\frac{2\pi k_1}{p}) & & & \\ & R_z(\frac{2\pi k_1}{p}) & & \\ & & R_z(\frac{2\pi k_1}{p}) & \\ & & & R_z(\frac{2\pi k_1}{p}) \end{bmatrix} \begin{bmatrix} H & & & \\ & H & & \\ & & H & \\ & & & H \end{bmatrix}$$

Using Diagonal Computations

- Diagonal computation uses identity

  $HR_x(\theta)H = R_z(\theta)$
- NDIAG algorithm by Bullock and Markov [4] reduces gate counts by tensor-product decomposition
- This technique may not be useful for this circuit for an inherent reason
  - the range of the parameters too big
  - Tensor product of two rotations adds the angles.
  - Need to explore this: could mean there are NO good circuits for this computation.

Tensor Products

- Diagonal unitary matrices have the form $\begin{bmatrix} e^{i\alpha} & 0 \\ 0 & e^{i\beta} \end{bmatrix}$
- Tensor products of two such matrices have the form

$$\begin{bmatrix} e^{i(\alpha+\gamma)} & & & \\ & e^{i(\alpha+\delta)} & & \\ & & e^{i(\beta+\gamma)} & \\ & & & e^{i(\beta+\delta)} \end{bmatrix}$$

On-going Work: Proof Sketch

- We want a circuit with *O(log log p)* gates and we are trying to combine them to form a computation that has *O(log p)* rotations.
- This is possible if we use the gates as rotations with angles that add like binary numbers.
- Problem: We want *O(log p)* rotations spread out in the range *[1,p]*. Using *O(log log p)* rotations we can either get *[1, log p]* consecutive rotation angles or *[1 p]* rotation angles with big holes in between.

Current Work

- Finding better circuits
- Finding circuits or proving that no good circuits exist for the greedy parameters
- Coming up with an analytical error bound for these parameters
- Empirically the error value is around *.60*

Conclusions

- Studied counters with exp memory savings
  - Can construct unitary computations
  - Can construct quantum circuits

Open Questions

- Are there any polynomial sized circuits or is there a size-accuracy tradeoff?
- Will Fourier transforms or other techniques give friendlier parameters?
- Do *other* quantum sequential circuits improve memory usage over classical circuits?

References

[1] A. Ambainis and R. Freivalds, " 1-way Quantum Finite Automata, Strengths, Weaknesses and Generalizations.", 1998. http://xxx.lanl.gov/quant-ph/9802062

[2] A. Barenco et al.,"Elementary Gates for Quantum Computation", *Physical Review A* (52), 1995, 3457-3467.

[3] M. Nielsen and I. Chuang,"Quantum Computation and Quantum Information", Cambridge Univ. Press, 2000.

[4] S.S. Bullock and I. L. Markov, "Smaller Circuits for Arbitrary n-qubit Diagonal Computations."http://xxx.lanl.gov/abs/quant-ph/0303039

[5] C. Moore and J. P. Crutchfield "Quantum Automata and Quantum Grammars." http://xxx.lanl.gov/quant-ph/9707031

[6] G. Brassard and P. Bratley, "Fundamental Algorithms", Prentice Hall, 1996.