Oracular Approximation of Quantum Multiplexors and Diagonal Unitary Matrices

Robert R. Tucci
P.O. Box 226
Bedford, MA 01730
tucci@ar-tiste.com

February 17, 2009

Abstract

We give a new quantum circuit approximation of quantum multiplexors based on the idea of complexity theory oracles. As an added bonus, our multiplexor approximation immediately gives a quantum circuit approximation of diagonal unitary matrices.

For an explanation of the notation used in this paper, see Ref.[1] Section 2.

Quantum multiplexors have proved themselves to be very useful as building blocks for quantum computing circuits. For a review of quantum multiplexors, see Ref.[1] Section 3.

As shown in Ref.[2], an $R_y(2)$-multiplexor with N_B controls can be compiled exactly using 2^{N_B} CNOTs.[1] It is believed that this number of CNOTs is a lower bound. It is therefore of interest to find multiplexor approximations with a lower CNOT count. Various multiplexor approximations have been considered before[3]. The goal of this paper is to give a new multiplexor approximation based on the idea of complexity theory oracles. As we shall see, any multiplexor approximation immediately gives an approximation of diagonal unitary matrices. Diagonal unitary matrices of dimension 2^{N_B} can also be compiled exactly using about 2^{N_B} CNOTs[2], and this number is believed to be a lower bound.

Consider an arbitrary $R_y(2)$-multiplexor whose target qubit is labelled τ and whose N_β control qubits are labelled $\vec{\beta} = (\beta_{N_\beta-1}, \ldots, \beta_1, \beta_0)$.

$$M = \exp\left(i \sum_{\vec{b}\in Bool^{N_\beta}} \theta_{\vec{b}}\sigma_Y(\tau)P_{\vec{b}}(\vec{\beta})\right) \quad (1a)$$

$$= \sum_{\vec{b}\in Bool^{N_\beta}} e^{i\theta_{\vec{b}}\sigma_Y(\tau)}P_{\vec{b}}(\vec{\beta}) \quad (1b)$$

$$= \prod_{\vec{b}\in Bool^{N_\beta}} \exp\left(i\theta_{\vec{b}}\sigma_Y(\tau)P_{\vec{b}}(\vec{\beta})\right) \ , \quad (1c)$$

for some $\theta_{\vec{b}} \in \mathbb{R}$. In the above, we define $\vec{b} = (b_{N_\beta-1}, \ldots, b_1, b_0)$, $\vec{\beta} = (\beta_{N_\beta-1}, \ldots, \beta_1, \beta_0)$, and $P_{\vec{b}}(\vec{\beta}) = \prod_{j=0}^{N_\beta-1} P_{b_j}(\beta_j)$. Also, $P_0 = |0\rangle\langle 0| = \overline{n} = 1-n$ and $P_1 = |1\rangle\langle 1| = n$, where n is the so called "number operator". In Eqs.(1), we've expressed M in 3 equivalent forms, the exponential, sum and product forms. The equivalence of these forms is readily established by applying $|\vec{b}\rangle_{\vec{\beta}}$ to the right hand side of each form. Note that we can "pull" the $\vec{b}$ sum out of the exponential, but only if we also pull out the projector $P_{\vec{b}}$.

Now we add a set of N_α ancilla qubits labelled $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_{N_\alpha})$. For each $\vec{b}$, the angle $\theta_{\vec{b}}$ can be expressed approximately, to a precision of N_α fractional bits, and this information can be stored in the qubits $\vec{\alpha}$. Let $|0\rangle_{\vec{\alpha}} = \prod_{k=1}^{N_\alpha} |0\rangle_{\alpha_k}$. If

[1] If we express a quantum circuit as a sequence of single-qubit rotations and CNOTs, then the number of CNOTs can be used as a measure of the time complexity of the circuit. Being two-body interactions, CNOTs take much longer to perform physically than single-qubit rotations, so we only count the former.

$$\theta_{\vec{b}} = 2\pi \sum_{k=1}^{N_\alpha} \frac{a_{\vec{b},k}}{2^k} \;, \tag{2}$$

where $a_{\vec{b},k} \in Bool$, then

$$\theta_{\vec{b}}|0\rangle_{\vec{\alpha}} = \left[\prod_{k=1}^{N_\alpha} \sigma_X(\alpha_k)^{a_{\vec{b},k}}\right] 2\pi \sum_{k=1}^{N_\alpha} \frac{n(\alpha_k)}{2^k} \left[\prod_{k=1}^{N_\alpha} \sigma_X(\alpha_k)^{a_{\vec{b},k}}\right] |0\rangle_{\vec{\alpha}} \;. \tag{3}$$

We will use the following evocative notation for the finite series:

$$\sum_{k=1}^{N_\alpha} \frac{n(\alpha_k)}{2^k} = 0.n(\alpha_1)n(\alpha_2)\ldots n(\alpha_{N_\alpha}) \;. \tag{4}$$

Substituting Eqs.(4) into Eq.(3), and then using the resulting expression for $\theta_{\vec{b}}|0\rangle_{\vec{\alpha}}$ in the definition Eq.(1) for M, gives:

$$M(\vec{\beta},\tau)|0\rangle_{\vec{\alpha}} = \sum_{\vec{b}} P_{\vec{b}}(\vec{\beta}) \left[\prod_{k=1}^{N_\alpha} \sigma_X(\alpha_k)^{a_{\vec{b},k}}\right] e^{i2\pi\, 0.n(\alpha_1)n(\alpha_2)\ldots n(\alpha_{N_\alpha})\sigma_Y(\tau)} \left[\prod_{k=1}^{N_\alpha} \sigma_X(\alpha_k)^{a_{\vec{b},k}}\right] |0\rangle_{\vec{\alpha}} \tag{5a}$$

$$= \Omega(\vec{\alpha}) e^{i2\pi\, 0.n(\alpha_1)n(\alpha_2)\ldots n(\alpha_{N_\alpha})\sigma_Y(\tau)} \Omega(\vec{\alpha})|0\rangle_{\vec{\alpha}} \;, \tag{5b}$$

where

$$\Omega(\vec{\alpha}) = \prod_{k=1}^{N_\alpha} \Omega(\alpha_k) \;, \tag{6}$$

where

$$\Omega(\alpha_k) = \sum_{\vec{b}} P_{\vec{b}}(\vec{\beta}) \sigma_X(\alpha_k)^{a_{\vec{b},k}} = \sigma_X(\alpha_k)^{\sum_{\vec{b}} a_{\vec{b},k} P_{\vec{b}}(\vec{\beta})} \;. \tag{7}$$

$\Omega(\vec{\alpha})$ is a product of N_α "standard quantum oracles" $\Omega(\alpha_k)$.[2] For example, if $N_\alpha = 2$ and $N_\beta = 2$ with

$$\begin{array}{l} \theta_{00} = 2\pi 0.01 \\ \theta_{01} = 2\pi 0.11 \\ \theta_{10} = 2\pi 0.10 \\ \theta_{11} = 2\pi 0.00 \end{array} \;, \quad [a_{\vec{b},k}] = \tag{8}$$

		$k \rightarrow$ 1	2
	00	0	1
$\vec{b}$	01	1	1
$\downarrow$	10	1	0
	11	0	0

then

[2]For an introduction to quantum oracles from a quantum computer programmer's perspective, see Ref.[4].

$$\Omega(\vec{\alpha}) = \quad (9)$$

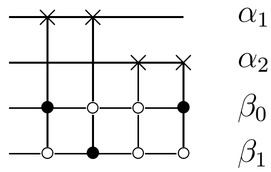

Fig.1 is an example, expressed in circuit form, of the multiplexor approximation that we hath wrought.

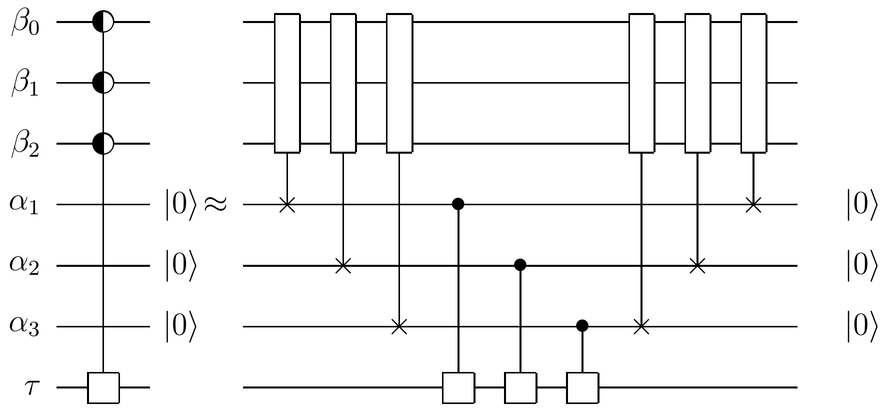

Figure 1: Oracular approximation of an $R_y(2)$-multiplexor with 3 controls, with the angles stored to a precision of 3 fractional bits.

Next, consider a diagonal unitary matrix D acting on qubits $\vec{\beta}$:

$$D = \exp\left(i \sum_{\vec{b}} \theta_{\vec{b}} P_{\vec{b}}(\vec{\beta}) \right) \ , \tag{10}$$

for some $\theta_{\vec{b}} \in \mathbb{R}$. By adding an ancilla target qubit τ, we can express D in terms of an $R_z(2)$-multiplexor:

$$D(\vec{\beta})|0\rangle_\tau = \exp\left(i \sum_{\vec{b}} \theta_{\vec{b}} \sigma_Z(\tau) P_{\vec{b}}(\vec{\beta}) \right) |0\rangle_\tau \tag{11a}$$

$$= e^{-i\frac{\pi}{4}\sigma_X(\tau)} \exp\left(i \sum_{\vec{b}} \theta_{\vec{b}} \sigma_Y(\tau) P_{\vec{b}}(\vec{\beta}) \right) e^{i\frac{\pi}{4}\sigma_X(\tau)} |0\rangle_\tau \tag{11b}$$

An oracular approximation of D follows immediately from this and the oracular multiplexor approximation.

In general, if a classical algorithm of polynomial complexity is known for calculating $\theta_{\vec{b}}$ given $\vec{b}$, then one can construct from this classical algorithm standard quantum oracles $\{\Omega(\alpha_k)\}_{\forall k}$, of polynomial complexity. *However, if no such classical algorithm of polynomial complexity is known, none may exist. If none exists, the standard quantum oracles* $\Omega(\alpha_k)$ *have exponential complexity.*

An upper bound on the error of our oracular multiplexor approximation is easily obtained. Let

$$\theta = 2\pi \sum_{k=1}^{\infty} \frac{a_k}{2^k} \ , \ \hat{\theta} = 2\pi \sum_{k=1}^{N_\alpha} \frac{a_k}{2^k} \ . \tag{12}$$

Then

$$|e^{i\theta} - e^{i\hat{\theta}}| = |e^{i\theta}| \left| 1 - \exp(-i2\pi \sum_{k=N_\alpha+1}^{\infty} \frac{a_k}{2^k}) \right| \tag{13a}$$

$$\leq 2\pi \left| \sum_{k=N_\alpha+1}^{\infty} \frac{a_k}{2^k} \right| \tag{13b}$$

$$\leq 2\pi \left| \sum_{k=N_\alpha+1}^{\infty} \frac{1}{2^k} \right| = \frac{2\pi}{2^{N_\alpha}} \left| \sum_{k=1}^{\infty} \frac{1}{2^k} \right| = \frac{2\pi}{2^{N_\alpha}} \ . \tag{13c}$$

To go from Eq.(13a) to Eq.(13b), we used the inequality $|1 - e^{ix}| = 2|\sin\frac{x}{2}| \leq |x|$ for $x \in \mathbb{R}$. Almost the same string of inequalities can be used to upper bound the error in our oracular multiplexor approximation. Let M be the multiplexor of Eq.(1) with the exact $\theta_{\vec{b}}$, and $\hat{M}$ the multiplexor with the approximate $\theta_{\vec{b}}$ (to a precision of N_α fractional bits). Using the matrix 2-norm[3], we get

$$\|M - \hat{M}\| = \|M\| \left\| 1 - \exp[-i2\pi \sum_{\vec{b}} \sum_{k=N_\alpha+1}^{\infty} \frac{a_{\vec{b},k}}{2^k} \sigma_Y(\tau) P_{\vec{b}}(\vec{\beta})] \right\| \tag{14a}$$

$$\leq 2\pi \|\sigma_Y(\tau)\| \left\| \sum_{\vec{b}} \sum_{k=N_\alpha+1}^{\infty} \frac{a_{\vec{b},k}}{2^k} P_{\vec{b}}(\vec{\beta}) \right\| \leq 2\pi \max_{\vec{b}} \left\| \sum_{k=N_\alpha+1}^{\infty} \frac{a_{\vec{b},k}}{2^k} \right\| \tag{14b}$$

$$\leq 2\pi \left| \sum_{k=N_\alpha+1}^{\infty} \frac{1}{2^k} \right| = \frac{2\pi}{2^{N_\alpha}} \left| \sum_{k=1}^{\infty} \frac{1}{2^k} \right| = \frac{2\pi}{2^{N_\alpha}} \ . \tag{14c}$$

Above, we used the fact that $\|M\| = \|\sigma_Y(\tau)\| = 1$ as is the case for any unitary matrix.

[3]The 2-norm of a matrix is defined as its largest singular value (singular values defined ≥ 0). For a review of matrix norms, see Ref.[5]

Acknowledgements

I got the idea for this paper from an exchange of emails between me, Rolando Somma, Emmanuel Knill, Howard Barnum, Pawel Wocjan and Richard Cleve.(Everyone received the same emails). I offered to one of these individuals to include him as a co-author of this paper, but he refused the offer (didn't even want to be acknowledged). None of the above mentioned people that corresponded with me has vetted or proof-read this paper. Any errors are my own. There is some indirect precedent to this work. According to Prof. Wocjan, "It is known how to efficiently simulate sparse Hamiltonians (for example, see the papers by Dorit Aharonov, et.al., Sanders, Cleve, et.al., which Rolando mentioned in his reply to you). These methods do not immediately allow to efficiently implement a sparse unitary matrix. Stephen Jordan and I found a simple method to simulate any sparse matrix by using the techniques for simulating sparse Hamiltonians. While this method is not very difficult, we still want to write a short note because the ability to simulate sparse unitaries can be used to derive quantum algorithms for evaluating link invariants ..." I have not seen the so far unpublished work of Jordan/Wocjan. I don't believe that I use "the techniques for simulating sparse Hamiltonians" in this paper. Furthermore, it appears that Jordan/Wocjan consider sparse unitary matrices in general, whereas I consider the very special sparse case of quantum multiplexors. Sometimes, by considering a special case, one can say much more and be more specific. For these reasons, I believe that this paper might be significantly different to the one by Jordan/Wocjan, and that it might be useful to someone besides me.

References

[1] R.R.Tucci,"Use of a Quantum Computer to do Importance and Metropolis-Hastings Sampling of a Classical Bayesian Network", arXiv:0811.1792

[2] R.R.Tucci,"A Rudimentary Quantum Compiler(2cnd Ed.)", arXiv:quant-ph/9902062

[3] R.R.Tucci, "Quantum Compiling with Approximation of Multiplexors", arXiv:quant-ph/0412072

[4] "Almost All Quantum Oracles Are Impossible to Realize in Practice(V.2)", by R.R. Tucci, available at www.qbnets.wordpress.com

[5] G.H. Golub and C.F. Van Loan, *Matrix Computations, Third Edition* (John Hopkins Univ. Press, 1996).