Quantum Boolean Circuit Construction and Layout under Locality Constraint

I-Ming Tsai and Sy-Yen Kuo

Department of Electrical Engineering
National Taiwan University, Taipei, Taiwan
tsai@lion.ee.ntu.edu.tw ; sykuo@cc.ee.ntu.edu.tw

Abstract

The discovery of Shor's prime factorization and Grover's fast database search algorithm have made quantum computing the most rapidly expanding research field recently. Nanotechnology, in particular silicon-based nanoscale device, has been proposed as one of the candidates that can be used to implement a quantum computer. In this paper, we have derived a systematic procedure to realize any general m-to-n bit combinational boolean logic using elementary quantum gates. The quantum circuit layout under the locality constraint is then formulated, together with the gate count evaluation function, to reduce the total number of quantum gates required to implement the circuit.

1 Introduction

Since Feynman [1] and Deutsch [2] introduced the idea and theoretical model of quantum computer in the early 1980's, a great deal of research effort has been focused on the topic of *quantum information science*. The discovery of Shor's prime factorization [3] and Grover's fast database search algorithm [4] have made quantum computing the most rapidly expanding research field recently. For a quantum algorithm to be useful, it is essential that the algorithm can be implemented using elementary quantum gates. Not long after Deutsch proposed his theoretical model of quantum computer, he showed that a three-bit quantum gate is universal and capable of realizing any unitary operation [5]. A few years later, it was shown [6, 7] that two-bit gates are sufficient to implement any unitary operation. Furthermore, Yao [8] showed that any function computable in polynomial time by a quantum Turing machine has a polynomial-size quantum circuit. All these results make experimental implementation of quantum circuits more practical.

A straightforward realization of quantum boolean logic is to use auxiliary qubits as intermediate storage. This inefficient implementation causes a large number of auxiliary qubits to be used. In this paper, we have derived a systematic procedure to realize any general m-to-n bit combinational boolean logic using elementary quantum gates. Our approach transforms the m-to-n bit classical mapping into a t-bit unitary quantum operation with minimum number of auxiliary qubits, then the unitary operation can be decomposed into one-bit rotation and two-bit control-U gates. Finally, the circuit layout under the locality constraint is formulated, together with the gate count evaluation function, to reduce the total number of quantum gates required to implement the circuit. Our approach can be applied to most silicon-based quantum computer.

2 Quantum Boolean Circuit Construction

2.1 Problem Description

Since the time evolution of any quantum transformation is a unitary and logically reversible process,

0-7803-7215-8/01/$10.00©2001 IEEE.

thus any quantum boolean logic can be represented using a permutation. The problem of transforming any m-to-n bit combinational boolean logic into a permutation can be formulated as follows:

Problem 1: Given a classical m-to-n bit combinational boolean logic

$$C : A(\{0,1\}^m) \rightarrow B(\{0,1\}^n), \quad (1)$$

and an integer p $(0 \leq p \leq m)$, construct a t-bit permutation

$$Q : \Psi(\{0,1\}^t) \rightarrow \Psi(\{0,1\}^t) \quad (2)$$

such that for each classical mapping $\alpha = \alpha_0\alpha_1\cdots\alpha_{m-1} \in A$ $(\alpha_i \in \{0,1\})$ and $\beta = C(\alpha) = \beta_0\beta_1\cdots\beta_{n-1}$ $(\beta_i \in \{0,1\})$, there exist two states $\psi = \psi_0\psi_1\cdots\psi_{m-1}\cdots\psi_{t-1} \in \Psi$ and $\phi = \phi_0\phi_1\cdots\phi_{t-1} \in \Psi$ satisfying:

(1) $\psi_i = \alpha_i$, for $i = 0, 1, \ldots, m-1$
(2) $\psi_i = 0$, for $i = m, m+1, \ldots, t-1$
(3) $Q(\psi) = \phi$
(4) $\phi_i = \alpha_i$, for $i = 0, 1, \ldots, p-1$
(5) $\phi_i = \beta_{i-p}$, for $i = p, p+1, \ldots, p+n-1$

The construction process is described in the following sections.

2.2 Building the Quantum Transformation Table

For any combinational boolean logic, a *Classical Transformation Table* can be used to describe the behavior of the circuits. In this section, for the purpose of description, we will use the notation $A[i][*]$ to denote the i-th row and the notation $A[i][m : n]$ to denote the i-th row with column index starting from m to n. Similar notations are used to denote a column block.

Taking an m-to-n bit circuit as an example, a classical transformation table consists of two parts, a 2^m-by-m α table for input, and a 2^m-by-n β table for output. Each row of the α table, $\alpha[i][*]$, contains an m-bit input pattern, while the same row of the β table, $\beta[i][*]$, holds the corresponding n-bit output.

As in the classical case, a *Quantum Transformation Table* is used to describe a t-bit quantum boolean logic. A quantum transformation table consists of two parts, a 2^t-by-t ψ table for input, and a ϕ table of the same size for output. Each row of the ψ table, $\psi[i][*]$, contains a t-bit input pattern, while the same row of the ϕ table, $\phi[i][*]$, holds the corresponding t-bit output. Because the quantum operation is a reversible unitary transformation, the 2^t rows in the ϕ table are simply a permutation of the input patterns.

The steps to build the quantum transformation table based on the classical circuits is shown below:

Step I. Preserve the input qubits.
We define the *preserved* qubits to be the input qubits that have to stay unchanged after the operation, while the *volatile* qubits are the input qubits that can be over-written by the output qubits. Preserved qubits can be used as inputs for other circuits again. Without loss of generality, assume qubits 0 to $p-1$ $(0 \leq p \leq m)$ are the qubits to be preserved and qubits p to $m-1$ are volatile qubits. Note that p can be zero, in which case no input qubit is preserved. Now prepare two empty tables, ψ and ϕ, which are both of size 2^m-by-m. For each row i $(0 \leq i \leq 2^m - 1)$, copy $\alpha[i][0 : m-1]$ to $\psi[i][0 : m-1]$. If $p \neq 0$, also copy the preserved qubits from $\alpha[i][0 : p-1]$ to $\phi[i][0 : p-1]$.

Step II. Assign the output qubits.
Since qubits 0 to $p-1$ are used to preserve the input qubits, assign qubits p to $p+n-1$ to hold the output qubits. Expand the width of the ϕ table whenever needed. For each row i $(0 \leq i \leq 2^m - 1)$, copy $\beta[i][0 : n-1]$ to $\phi[i][p : p+n-1]$.

Step III. Distinguish each output state.
For a unitary quantum evolution, the quantum transformation table needs to be one-to-one and onto. If for every $a, b \in \{0,1\}^{p+n}$ in ϕ, $a \neq b$, then set $d = 0$, go to next step. Otherwise, set

$$d = \lceil \log_2 M \rceil \quad (3)$$

where M is the maximum number of occurrences for a repeat pattern. Add extra d columns (numbering from $\phi[*][p+n]$ to $\phi[*][p+n+d-1]$) to the ϕ table. Expand the width of the ϕ table whenever needed.

Step IV. Add auxiliary qubits

If $m = p+n+d$, no auxiliary qubit is needed. The total number of qubits, t, equals m, go to next step. Otherwise, if $m < p+n+d$, let $x = (p+n+d) - m$ and add x auxiliary qubits to the ψ table (numbering from $\psi[*][m]$ to $\psi[*][m+x-1]$). Assign these qubits to be all 0's. The total number of qubits, t, equals $p+n+d$.

Step V. Expand the quantum transformation table

If auxiliary qubits are used, expand both ψ and ϕ tables to be 2^t rows in length. For the ψ table, repeat the original block 2^x times and, for each block, fill in the auxiliary qubits with a unique x-bit pattern. For the ϕ table, leave the new entries blank.

Based on the constraints derived from the classical boolean circuit, the quantum transformation table is now partially constructed. The permutation can be completed simply by filling in the blanks to make it a one-to-one and onto mapping. To do this, we construct a *quantum state transformation digraph* $G = (V, E)$, where

$$\begin{aligned} V &= \{v_i \mid v_i = \psi[i][*]\} \\ E &= \{(v_s, v_d) \mid v_s = \psi[i][*], v_d \cong \phi[i][*]\} \quad (4) \end{aligned}$$

and $0 \le i \le 2^t - 1$. The digraph has 2^t vertices, corresponding to each of the 2^t rows in the ψ table. An edge is defined from v_s to v_d if it is possible for Q to map v_s to v_d. The notation $v_d \cong \phi[i][*]$ is used to denote, when only u ($u < t$) qubits are specified in $\phi[i][*]$, all states that are *compatible* to the entry. This results in 2^{t-u} edges to be generated for each of the possible (v_s, v_d) pairs. Filling in the $t-u$ blank qubits in the ϕ table selects one of the possible edges and delete others.

Using the digraph G, the problem is equivalent to finding a set of disjoint cycles that cover all the vertices in V, the problem is formulated as follows:

Problem 2: Given a digraph $G = (V, E)$, find a family of sets $S = \{S_i \mid S_i = \{v_0^i, v_1^i, \ldots, v_{n-1}^i\}, v_j^i \in V\}$ and corresponding cycles $C = \{C_i \mid C_i = (v_0^i, v_1^i, \ldots, v_{n-1}^i), v_j^i \in V\}$ such that $\bigcup_{S_i \in S} S_i = V$ and $\bigcap_{S_i \in S} S_i = \emptyset$.

This is clearly a *set partitioning problem*, with each partition being a cycle. The problem has been extensively studied in graph theory and can be used to construct the permutation Q. An example of Boolean logic $X = A + B$ and $Y = A + \bar{A}\bar{B}$ without preserved qubit is shown in Fig.(1).

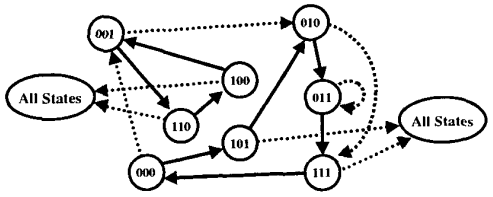

Figure 1: Set covering a digraph with disjoint cycles

2.3 Implement the Permutation using Elementary Quantum Gates

A permutation can be implemented using elementary quantum gates by the following propositions.

Proposition I. A permutation consists of one or multiple disjoint cycles. Since disjoint cycles commute, each cycle in the permutation can be implemented individually.

Proposition II. Given a general cycle $C = (p_0, p_1, p_2, \ldots, p_{n-1})$, C can be constructed using $n-1$ transpositions, *i.e.* $C = (p_0, p_1)(p_1, p_2) \cdots (p_{n-3}, p_{n-2})(p_{n-2}, p_{n-1})$.

Proposition III. Given any two general states p and q, with $\Delta(p, q) = d$, the transposition $U = (p, q)$ can be decomposed into $2d-1$ adjacent state transpositions.

To further implement an adjacent state transposition, we introduce the generalized n-bit Toffoli gate. Assuming $S, R, I \in \{0,1\}^n$, $S \wedge R = R \wedge I = S \wedge I = 0$, and the Hamming distance between I and $\{0\}^n$ is 1, an n-bit Toffoli gate can be represented by $T(S,R,I)$. In this notation, S and R, if expressed in binary digits, mark the positions of control bits. The bits that are set in S specify the control bits that have to be 1's to activate the logic. Similarly, the bits that are set in R specify the bits that have to be 0's to activate the logic. I simply represents the target bit to be inverted when the conditions of S and R are satisfied. Those bits that are not specified in either S, R, or I are *don't care* bits. The operation of a $T(S,R,I)$ gate is summarized as follows: assuming an input $X = x_{n-1}x_{n-2}\cdots x_1x_0$, the target bit x_r is simply the bit that is set in I, and

$$\begin{aligned} \hat{x}_r &= (\wedge_{i=0}^{n-1}((s_i \wedge x_i) \vee (r_i \wedge \bar{x}_i) \vee (\bar{s}_i \wedge \bar{r}_i))) \oplus x_r \\ \hat{x}_i &= x_i, \quad i = 0,\ldots,r-1,r+1,\ldots,n-1 \end{aligned} \tag{5}$$

Using this notation, we have the following proposition:

Proposition IV. Given any two states p and q with $\Delta(p,q) = 1$, the transposition $U = (p,q)$ can be implemented using a $T(S,R,I)$ gate with

$$S = p \wedge q, \qquad R = \bar{p} \wedge \bar{q}, \qquad I = p \oplus q. \tag{6}$$

Note that the $T(S,R,I)$ gate can be further decomposed into one-bit rotation and two-bit control-U gates [9]. This completes our quantum boolean circuit construction.

3 Quantum Boolean Circuit Layout

In Kane's Si:P quantum system [10], spins associated with donors in silicon function as qubits. Quantum operations are performed using a combination of voltage applied to gates adjacent to the spins and magnetic fields applied resonant with spin transitions. In this proposal, the SWAP operation in combination with single qubit rotation can be used as universal quantum operations. Similarly, Loss and DiVincenzo [11] have proposed an implementation of a universal set of one- and two-qubit gates using spin states of coupled single-electron quantum dots. The one-bit rotation, SWAP and XOR gate can be implemented by gating of the tunneling barrier between neighboring quantum dots.

Although previous work [9] has shown that for any unitary 2×2 matrix U, a $\wedge_1(U)$ gate can be simulated by six basic gates, there is no *locality restriction* imposed on the qubits. Note that in the implementation described above and most silicon-based system, spins can only interact with their neighbors. If the locality constraint is taken into consideration, all elementary gates with non-adjacent inputs should be further decomposed using quantum gates between two adjacent qubits, as shown in Fig.(2).

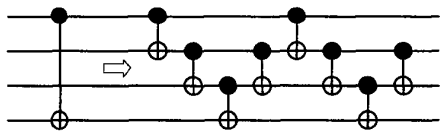

Figure 2: Adjacent control-not gate implementation

From the decomposition of non-adjacent XOR gate, we see the total number of adjacent XOR's depends linearly on the distance between the control bit and the target bit. Assume the distance between the control bit and the target bit is d, then $4(d-1)$ adjacent gates are required. This suggests that we can arrange the *layout* (*i.e.* position) of the qubits to get a more efficient circuit. This problem is formulated as follows:

Problem 3: Given a weighted graph $G(V,E)$, assume the weight $w(v_1,v_2)$ be the number of XOR operations to be performed between v_1 and v_2, find a distinct integer assignment I_v for each $v \in V$, such that $\sum_{(v_1,v_2)\in E}(w(v_1,v_2) * |I_{v_1} - I_{v_2}|)$ is minimized.

The problem suggested above can be taken as the gate count evaluation function when the total num-

ber of gate count is to be reduced. This transforms the set partitioning problem into the following form:

Problem 4: Given a digraph $G = (V, E)$ and the gate count cost Ω_{C_i} associated with each cycle C_i, find a family of sets $S = \{S_i \mid S_i = \{v_0^i, v_1^i, \ldots, v_{n-1}^i\}, v_j^i \in V\}$ and corresponding cycles $C = \{C_i \mid C_i = (v_0^i, v_1^i, \ldots, v_{n-1}^i), v_j^i \in V\}$ to reduce

$$\Omega_P = \sum_{C_i \in C} \Omega_{C_i} \tag{7}$$

subject to:

$$(1) \bigcup_{S_i \in S} S_i = V$$
$$(2) \bigcap_{S_i \in S} S_i = \emptyset$$

The problem is essentially a *constrained set partitioning problem*, with each partition being a cycle. The solution can be found using a combinatorial programming approach [12], as we demonstrated in the following section.

4 Gate Count Evaluation Process

A simple but effective algorithm is described in this section to demonstrate how the elementary gate count is reduced.

Step I. Enumerate all cycles.
Given the digraph $G(V, E)$ described in the quantum transformation table, list all cycles C_i ($i = 0, 1, 2, \ldots$) in the digraph. This can be done in the following way:

(i) Select a target edge (v_ψ, v_ϕ), and list all cycles containing the edge. To find all cycles containing (v_ψ, v_ϕ), just list all paths from v_ϕ to v_ψ, then cycles can be found by concatenating any path from v_ϕ to v_ψ with the edge (v_ψ, v_ϕ).

(ii) Delete the target edge in (i). If there is any edge left in G, go to (i), otherwise all cycles are found.

Step II. Initialization.

(i) Let $X = \{x_i\}$ be a matrix of size $n \times 1$ with $x_i = 1$ if C_i is included in the solution, and $x_i = -1$ if it has been excluded. Initially set $x_i = 0$ for each i.

(ii) Let $A = \{a_{ij}\}$ be an $m \times n$ matrix with $a_{ij} = 1$ if $v_i \in C_j$, and $a_{ij} = 0$ if $v_i \notin C_j$.

(iii) For each cycle C_i, calculate the elementary gate count Ω_{C_i}.

Step III. Reduction (optional).
The optional reduction process makes the optimization task easier. Although there are many effective rules, only three reductions are described here. Let S be an $n \times 1$ matrix with all elements set to 1's and $R = \{r_i\} = AS$ be the $m \times 1$ matrix that describes the coverage of the vertices.

(i) If $r_i = 0$ for any i, no solution exists.

(ii) For any i, if $r_i = 1$ and $a_{ij} = 1$, then mark C_j as included.

(iii) If C_j denotes any cycle that has been included, then all C_k with $C_k \cap C_j \neq \emptyset$ must be marked as excluded.

The reduction rules can be applied over again until no further reduction is possible.

Step IV. Search the optimal solution.
A depth-first search algorithm is used here to search the optimal solution.

(i) Let $M = \infty$ be the initial elementary gate count.

(ii) If all vertices are covered, update M and record X in case $\Omega_P < M$, return. Otherwise, for each C_j that has not been marked, update X to include C_j, apply the reduction rules as described in Step III, then recursively call step (ii) with the parameter X.

After these steps are done, the selected cycles are recorded in X and the gate count is M.

5 Conclusions

There are two reasons that we propose such an analysis on quantum circuit construction. First of all, since the conventional device architecture will eventually reach its physical limit, we have to take advantage of the quantum physics at nanometer scale

if we want to continue the computer hardware evolution. Secondly, since the time evolution of any quantum transformation is a thermodynamically reversible process, a general-purpose computer implemented using nanoscale devices and quantum operations can theoretically operate with arbitrary little energy [13, 14], which is an attractive feature beyond today's technology. In this paper, we have derived a systematic procedure to transform any general boolean logic into its quantum version. At the same time, the circuit layout under the locality constraint is analyzed to reduce the total number of gate count. This method can be applied to most silicon-based quantum computer system.

References

[1] R.P. Feynman. Int. J. Theor. Phys. **21**, 467 (1982).

[2] D. Deutsch. Proc. Roy. Soc. Lond. A **400**, 97 (1985).

[3] P.W. Shor. Proc. of the 35th Annual Symposium on the Foundations of Computer Science, IEEE, Computer Society Press, New York, 1994, p.124; SIAM J. Comput., vol 26, no. 5, 1484-1509 (Oct. 1997).

[4] L.K. Grover. Proc. of the 28th Annual ACM Symposium on the Theory of Computing, 1996, p.212.

[5] D. Deutsch. Proc. Roy. Soc. Lond. A **425**, 73 (1989).

[6] A. Barenco. University of Oxford preprint: *A Universal Two-Bit Gate for Quantum Computation.*

[7] D. P. DiVincenzo. Phys. Rev. A **50**, 1015 (1995).

[8] A. Yao. Proc. of the 34th Annual Symposium on the Foundation of Computer Science, IEEE, Computer Society Press, Los Alamitos, 1993, p.352.

[9] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. H. Smolin, and H. Weinfurter. Phys. Rev. A **52**, 3457 (1995).

[10] B.E. Kane. Nature **393**, 133 (1998).

[11] D. Loss, D.P. DiVincenzo. Phys. Rev. A **57**, 120 (1998).

[12] C. Berge. *Alternating chain methods: A survey,* in *Graph Theory and Computing*, Ed R. Read, Academic Press, New York, (1972).

[13] R. Landauer. IBM. J. Res. Develop. **3**, 183 (1961).

[14] C. Bennett. IBM. J. Res. Develop. **17**, 525 (1973).