Array Representation of Entanglement for Pseudo-Quantum Gate Computing on Conventional Digital Circuits

Hirano Takeshi [*]

2020/06/24

Abstract

Entanglement is important on quantum gate based computer. Simple implementation of simulated entanglement on conventional digital circuits requires spatial complexity $O(2^N)$ where N is number of qubits. In this article, array representation of entanglement which requires spatial complexity $O(N^2)$ is proposed for simulated quantum gate on conventional digital circuits. Simulation shows proposed method can solve the problem which Deutsch-Jozsa algorithm solves.

1 Background

Entanglement is important on quantum gate based computer. Entanglement is caused by CNOT gate. XOR (namely CNOT) gate shrinks two input qubits to one output qubit. This output qubit has dependency to input qubits. Representation of qubit which consists of two complex number is not sufficient to represent this dependency. But representation based on divided cases which have complexity $O(2^N)$ seems to be too large. If this output dependent to N independent qubits, representation of the output should have information $O(N)$ not to lose any information with XOR. So I propose the representation of qubit which consists of array with length of N.

2 Proposed Representation

Unitary gates and CNOT gates can make any other quantum gates. Investigating how unitary gates and CNOT gates affect to independent bits will help constructing the representation of its output.

Here we use independent qubits q_1, q_2, each of them consists of two complex number. U represents 2×2 unitary matrix. NOT and I are special case of U.

$$q_1 = \begin{pmatrix} q_{1(0)} \\ q_{1(1)} \end{pmatrix} \tag{1}$$

$$q_2 = \begin{pmatrix} q_{2(0)} \\ q_{2(1)} \end{pmatrix} \tag{2}$$

$$NOT = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{3}$$

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{4}$$

$|q_{k(0)}|^2$ is the probability of that kth qubit takes value 0, as conventional representation of qubit.

When CNOT gate takes q_1 and q_2 as inputs, it outputs q_1 and $XOR(q_1, q_2)$. Combination of unitary gate and XOR gate behave like below.

$$XOR(q_1, q_2) = (q_{1(0)}|q_1 = 0 >)(q_2) + (q_{1(1)}|q_1 = 1 >)(NOT.q_2) \tag{5}$$

$$XOR(q_1, U.q_2) = (q_{1(0)}|q_1 = 0 >)(U.q_2) + (q_{1(1)}|q_1 = 1 >)(NOT.Uq_2) \tag{6}$$

$$U.XOR(q_1, q_2) = (q_{1(0)}|q_1 = 0 >)(U.q_2) + (q_{1(1)}|q_1 = 1 >)(U.NOT.q_2) \tag{7}$$

From these behavior, two 2×2 matrices for one independent qubit seems to be required. Then output qubit q_o which has dependency on independent N qubits can be array like as Eq.8.

[*]hyranno4pub@gmail.com http://uncotechhack.net

$$q_o = \{M_{1[0]}, M_{1[1]}, M_{2[0]}, M_{2[1]}, ...M_{N[0]}, M_{N[1]}\} \tag{8}$$

2.1 Marginal probability

To perform a measurement, probability of result $0, 1$ have to be calculated on output qubit q_o. From equations above, marginal probability P_m of result on q_o should be as Eq.11.

$$f_p(M,q) = \begin{pmatrix} \left|M_{(11)}q_{(0)}\right|^2 + \left|M_{(12)}q_{(1)}\right|^2 \\ \left|M_{(21)}q_{(0)}\right|^2 + \left|M_{(22)}q_{(1)}\right|^2 \end{pmatrix} \tag{9}$$

$$f_{mp}(k) = \begin{cases} f_p(M_{k[0]}, q_k) & \text{if}(k=1) \\ f_{mp}(k-1)_0 . f_p(M_{k[0]}, q_k) + f_{mp}(k-1)_1 . f_p(M_{k[1]}, q_k) & \text{otherwise} \end{cases} \tag{10}$$

$$P_m(q_o) = f_{mp}(N) \tag{11}$$

With this representation, q_o may have $M_{k[0]}$ and $M_{k[1]}$ even if q_o have no dependency on q_k, considering C_0, C_1 below.

$$C_0 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \tag{12}$$

$$C_1 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix} \tag{13}$$

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} = C_0 . q_k \tag{14}$$

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} = C_1 . q_k \tag{15}$$

If q_o has no dependency on q_k, then $M_{k[0]} = C_0$ and $M_{k[1]} = C_1$. So if $q_o = q_j$, q_o can be represented as array like below.

$$q_o = \{M_{1[0]}, M_{1[1]}, M_{2[0]}, M_{2[1]}, ...M_{N[0]}, M_{N[1]}\} \tag{16}$$

$$M_{k[0]} = \begin{cases} I & \text{if}(k=j) \\ C_0 & \text{otherwise} \end{cases} \tag{17}$$

$$M_{k[1]} = \begin{cases} I & \text{if}(k=j) \\ C_1 & \text{otherwise} \end{cases} \tag{18}$$

While P_m is marginal probability of single output, you need to calculate conditional probability when you measure multiple output. To calculate conditional probability, you will need $AND(q_{o1}, q_{o2})$.

2.2 Unitary gate

Assuming from Eq.7, unitary gate behave like below.

$$(M_{k[0]}, M_{k[1]}) \rightarrow \begin{cases} (U.M_{k[0]}, U.M_{k[1]}) & \text{if}(k=m) \\ (M_{k[0]}, M_{k[1]}) & \text{otherwise} \end{cases}$$
$$\text{where } m = \max(l \mid \det(M_{l[0]}) \neq 0) \tag{19}$$

2.3 CNOT gate

I had little more investigation on XOR shown as Eq.21 and Eq.23. $E_{k(v)}$ and f_{XOR} has been defined for shortening text as Eq.20 and Eq.22.

$$E_{k(v)} = (q_{k(v)} | q_k = v >) \tag{20}$$

$$\begin{aligned}
&XOR(M_a.q_1, M_b.q_1)\\
&= \left((M_a.q_1)_{(0)}|M_a.q_1 = 0>\right).(M_b.q_1) + \left((M_a.q_1)_{(1)}|M_a.q_1 = 1>\right).(NOT.M_b.q_1)\\
&= \begin{pmatrix} M_{a(11)}.M_{b(11)}E_{1(0)} + M_{a(12)}.M_{b(12)}E_{1(1)} \\ M_{a(11)}.M_{b(21)}E_{1(0)} + M_{a(12)}.M_{b(22)}E_{1(1)} \end{pmatrix}\\
&\quad + \begin{pmatrix} M_{a(21)}.M_{b(21)}E_{1(0)} + M_{a(22)}.M_{b(22)}E_{1(1)} \\ M_{a(21)}.M_{b(11)}E_{1(0)} + M_{a(22)}.M_{b(12)}E_{1(1)} \end{pmatrix}\\
&= \begin{pmatrix} (M_{a(11)}.M_{b(11)}) + (M_{a(21)}.M_{b(21)}) & (M_{a(12)}.M_{b(12)}) + (M_{a(22)}.M_{b(22)}) \\ (M_{a(11)}.M_{b(21)}) + (M_{a(21)}.M_{b(11)}) & (M_{a(12)}.M_{b(22)}) + (M_{a(22)}.M_{b(12)}) \end{pmatrix}.(q_1)
\end{aligned} \tag{21}$$

$$f_{XOR}(M_a, M_b) = \begin{pmatrix} (M_{a(11)}.M_{b(11)}) + (M_{a(21)}.M_{b(21)}) & (M_{a(12)}.M_{b(12)}) + (M_{a(22)}.M_{b(22)}) \\ (M_{a(11)}.M_{b(21)}) + (M_{a(21)}.M_{b(11)}) & (M_{a(12)}.M_{b(22)}) + (M_{a(22)}.M_{b(12)}) \end{pmatrix} \tag{22}$$

$$\begin{aligned}
&XOR(M_a.XOR(q_1, q_2), M_b.q_2)\\
&= XOR\left(M_a.\begin{pmatrix} E_{1(0)}E_{2(0)} + E_{1(1)}E_{2(1)} \\ E_{1(0)}E_{2(1)} + E_{1(1)}E_{2(0)} \end{pmatrix}, M_b.\begin{pmatrix} E_{2(0)} \\ E_{2(1)} \end{pmatrix}\right)\\
&= \left(M_{1(11)}.(E_{1(0)}E_{2(1)} + E_{1(1)}E_{2(0)}) + M_{1(12)}.(E_{1(0)}E_{2(0)} + E_{1(1)}E_{2(1)})\right).\begin{pmatrix} M_{2(11)}.E_{2(0)} + M_{2(12)}.E_{2(1)} \\ M_{2(21)}.E_{2(0)} + M_{2(22)}.E_{2(1)} \end{pmatrix}\\
&\quad + \left(M_{1(21)}.(E_{1(0)}E_{2(1)} + E_{1(1)}E_{2(0)}) + M_{1(22)}.(E_{1(0)}E_{2(0)} + E_{1(1)}E_{2(1)})\right).\begin{pmatrix} M_{2(21)}.E_{2(0)} + M_{2(22)}.E_{2(1)} \\ M_{2(11)}.E_{2(0)} + M_{2(12)}.E_{2(1)} \end{pmatrix}\\
&= \begin{pmatrix} M_{1(11)}.(M_{2(11)}.E_{1(0)}E_{2(0)} + M_{2(12)}.E_{1(1)}E_{2(1)}) + M_{1(12)}.(M_{2(12)}.E_{1(0)}E_{2(1)} + M_{2(11)}.E_{1(1)}E_{2(0)}) \\ M_{1(11)}.(M_{2(21)}.E_{1(0)}E_{2(0)} + M_{2(22)}.E_{1(1)}E_{2(1)}) + M_{1(12)}.(M_{2(22)}.E_{1(0)}E_{2(1)} + M_{2(21)}.E_{1(1)}E_{2(0)}) \end{pmatrix}\\
&\quad + \begin{pmatrix} M_{1(21)}.(M_{2(21)}.E_{1(0)}E_{2(0)} + M_{2(22)}.E_{1(1)}E_{2(1)}) + M_{1(22)}.(M_{2(22)}.E_{1(0)}E_{2(1)} + M_{2(21)}.E_{1(1)}E_{2(0)}) \\ M_{1(21)}.(M_{2(11)}.E_{1(0)}E_{2(0)} + M_{2(12)}.E_{1(1)}E_{2(1)}) + M_{1(22)}.(M_{2(12)}.E_{1(0)}E_{2(1)} + M_{2(11)}.E_{1(1)}E_{2(0)}) \end{pmatrix}\\
&= E_{1(0)}.\begin{pmatrix} (M_{1(11)}.M_{2(11)} + M_{1(21)}.M_{2(21)})E_{2(0)} + (M_{1(12)}.M_{2(12)} + M_{1(22)}.M_{2(22)})E_{2(1)} \\ (M_{1(11)}.M_{2(21)} + M_{1(21)}.M_{2(11)})E_{2(0)} + (M_{1(12)}.M_{2(22)} + M_{1(22)}.M_{2(12)})E_{2(1)} \end{pmatrix}\\
&\quad + E_{1(1)}.\begin{pmatrix} (M_{1(12)}.M_{2(11)} + M_{1(22)}.M_{2(21)})E_{2(0)} + (M_{1(11)}.M_{2(12)} + M_{1(21)}.M_{2(22)})E_{2(1)} \\ (M_{1(12)}.M_{2(21)} + M_{1(22)}.M_{2(11)})E_{2(0)} + (M_{1(11)}.M_{2(22)} + M_{1(21)}.M_{2(12)})E_{2(1)} \end{pmatrix}\\
&= E_{1(0)}.\begin{pmatrix} (M_{1(11)}.M_{2(11)} + M_{1(21)}.M_{2(21)}) & (M_{1(12)}.M_{2(12)} + M_{1(22)}.M_{2(22)}) \\ (M_{1(11)}.M_{2(21)} + M_{1(21)}.M_{2(11)}) & (M_{1(12)}.M_{2(22)} + M_{1(22)}.M_{2(12)}) \end{pmatrix}.(q_2)\\
&\quad + E_{1(1)}.\begin{pmatrix} (M_{1(12)}.M_{2(11)} + M_{1(22)}.M_{2(21)}) & (M_{1(11)}.M_{2(12)} + M_{1(21)}.M_{2(22)}) \\ (M_{1(12)}.M_{2(21)} + M_{1(22)}.M_{2(11)}) & (M_{1(11)}.M_{2(22)} + M_{1(21)}.M_{2(12)}) \end{pmatrix}.(q_2)\\
&= E_{1(0)}.f_{XOR}(M_a, M_b).(q_2) + E_{1(1)}.f_{XOR}(M_a.NOT, M_b).(q_2)
\end{aligned} \tag{23}$$

Assuming from Eq.23, CNOT gate behave like below.

$$\left((M_{ak[0]}, M_{ak[1]}), (M_{bk[0]}, M_{bk[1]})\right) \rightarrow \left(f_{XOR}(M_{ak[0]}, M_{bk[0]}), f_{XOR}(M_{ak[1]}, M_{bk[1]})\right) \tag{24}$$

3 Simulation

With Eq.11, Eq.19 and Eq.24, quantum gate computing should be simulated. But I have not got correct mathematical proof for this representation yet. So I wrote a simulation to show it work fine. Simulation is written in SystemVerilog and run by ModelSim. SystemVerilog has been chosen to ensure this representation can be implemented as physical gates, not only software.

3.1 Problem

This simulation solves the problem which Deutsch-Jozsa algorithm solves. In this problem, we are given a black box quantum computer as an oracle, which implements a function $f : \{0,1\}^N \rightarrow \{0,1\}$. This function is either "constant" or "balanced". If it is constant, the function returns always 0, or always 1. If it is balanced, the function returns 0 for half of the input domain and 1 for the others. The task is to determine if the f is constant or balanced by using the oracle. In this simulation, also the oracle uses proposed array representation.

3.2 Implementation

You can take a look and run the attached SystemVerilog code for detail. Unlike real-quantum computer, we can get marginal probability of the output. So what we have to do is to initialize to 0 and apply Hadamard gate to each input qubit, then connect all to the oracle and calculate the marginal probability of the output. If the probability is $\{1, 0\}$, the function is constant returning 0. If the probability is $\{0, 1\}$, the function is constant returning 1. If the probability is $\{0.5, 0.5\}$, the function is balanced.

3.3 Result

Simulation works fine for number of qubits 4, solving the problem correctly.

4 Conclusion

Array representation of entanglement is proposed. With this proposed method, we can at least solve the Deutsch-Jozsa problem on conventional circuits in spatial complexity $O(N^2)$ instead of $O(2^N)$, where N is number of input qubits.