Quantum Trick Cellular Automata

Sai Venkatesh Balasubramanian

Sree Sai Vidhya Mandhir, Mallasandra, Bengaluru-560109, Karnataka, India.
saivenkateshbalasubramanian@gmail.com

Abstract

The present work explores a radically new perspective exploiting the advantages of quantum computation using classical systems. Termed 'Quantum Tricks', the essence of the technique is replacing the inherent probabilistic nature of a qubit with a pseudorandom generator, resulting in the implementation of basic Quantum Gates such as the Pauli X, Y, Z, Hadamard and CNOT gates using extremely simple mathematical operations followed by an implementation of entanglement. Following this, a brief overview of the concept of cellular automata and associated rules is presented. Finally, the 'Quantum Trick Cellular Automata' (QTCA) concept is formulated, differing from the classical cellular automata in two aspects - presence of superposed states other than 1 and 0, and implementation of above mentioned quantum gates. It is seen that QCTA comprising of CNOT and Entanglement operations give rise to rich, complex and ornamental patterns, characteristic of the underlying chaos. The extreme simplicity of the proposed design coupled with its efficacy in implementing rich patterns using cellular automata concepts form the highlights of the present work.

Keywords: Cellular Automata, Quantum Tricks, Pseudorandom Numbers, Entanglement

1. Introduction

Quantum Computing has witnessed growth in leaps and bounds, thanks in part due to the exciting prospects it provides to understanding the nature of various quantum phenomena, and partially due to its exciting applications such as quantum teleportation and superdense coding [1, 2, 3, 4, 5, 6, 7]. At the heart of all these is the Quantum Bit, or 'qubit', an extension of the conventional bit with the added properties of quantum superposition and measurement-induced collapse [1]. The use of such superposed states enable qubits to perform simultaneously, the functionalities of many bits. Conventionally, quantum computation has been implemented using quantum systems, such as superconductiing qubits, spintronics, nuclear magnetic resonance, or laser powered photonic devices [5, 6, 7, 8, 9, 10, 11].

The present work proposes a radically different way to achieve quantum computation based applications. The central idea is to use quantum tricks, using classical systems. Specifically, the present work purports to achieve the inherent probabilistic nature of a qubit using pseudorandom number generators [12]. Using this analogue, the basic Pauli Rotation Operators are explored, followed by an important quantum computing phenomenon, entanglement [5]. Following this, a brief overview of the concept of cellular automata, in the light of conventional classical logic, along with the concepts of rule definition and evolution is presented. Based on this, the QTrick Cellular Automata concept is formulated, and various quantum logic based rules are explored. It is seen that certain rules involving QTrick entanglement are able to self-generate ornamental patterns, characteristic of the underlying chaos. The extreme simplicity of the pseudorandom number based quantum trick cellular automata combined with the ability to generate rich patterns forms the novelty of the present work.

2. Quantum Trick Gates and Entanglement

The primary step in a quantum computation formulation is the definition of the quantum bit, or qubit. For this purpose, the standard Bloch sphere representation of a qubit is given in Fig. (1) [1]. All points lying on the surface of the Bloch Sphere are considered to represent classical states of $|0>$ or $|1>$. Points not on the Bloch sphere surface represent mixed states of $|0>$ and $|1>$, which are nothing but superposition states of $|0>$ and $|1>$. In accordance with normalization criteria, the superposed state is typically set to $1/\sqrt{2}[|0>+|1>]$ [1].

In order to generate qubits with their inherent probabilistic nature intact, the present work proposes the use of a pseudorandom number generator, generating random elements from the set [0,0.5,1] [12]. Consequently, the occurrence of 0 and 1 are assumed as the states $|0>$ and $|1>$ respectively, while the occurrence of 0.5 is assumed as the superposition state $1/\sqrt{2}[|0>+|1>]$. These qubit analogues generated using the pseudorandom quantum trick are termed 'QTrick-Bits'.

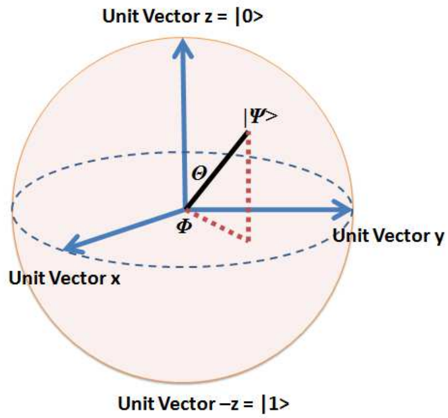

Figure 1: The Bloch Sphere Representation of a Qubit

It is well-known that any operator in quantum computing can be replaced as suitable combinations of the basic Pauli Gates, of which there are three - X, Y and Z, corresponding to the three rotation operators along three axes of the Bloch Sphere [1].

The Pauli X Gate, similar in function to a conventional NOT Gate is given by the following matrix, essentially interchanging $|0>$ and $|1>$ states, and for this reason it is called the Bit-Flip operator. Denoting the QTrick Bit by Q, the Pauli X Gate is mathematically implemented as $X = 1 - Q$. [1].

$$[X] = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{1}$$

The Pauli Z Gate is given by the following matrix [1]:

$$[Z] = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2}$$

This operator leaves $|0>$ unchanged while shifting $|1>$ by a phase of 180 degrees and for this reason is called the Phase-Flip operator. Mathematically, for a QTrick Bit, Pauli Z Gate is $Z = -Q$.

The Pauli Y Gate combines the functionalities of both the X and Z Gates, and is given by the following matrix [1]:

$$[Y] = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}. \tag{3}$$

The most notable feature of the Pauli Y Gate is its ability to transform between real and imaginary spaces. According to the commutivity principle, the Y Gate can be constructed as a combination of the X and Z gates in the following fashion: $ZX = iY$ [1].

Based on the above mentioned relations, the Pauli X, Y and Z QTrick Gates are implemented using Altera Cyclone II 2C20 for a QTrick Bit Q, generated using a pseudorandom number generator for 1000 samples, each of which represents a qubit sampled at that instant. The results for the three Pauli Gates are shown in Fig. (2).

The next step is to explore the phenomenon of Entanglement, where the phase relationship between two Qubits forces the observer to describe one particle in terms of another [13, 14, 15, 16]. Thus, any measurement made on one particle instantaneously affects the other, irrespective of the distance of separation between them, causing Einstein to famously describe it as spooky action at a distance [4]. The standard way of generating entanglement in quantum computation is by creating Bell states, given for a two qubit system as follows: $|B00 >= 0.707(|00 > +|11 >)$, $|B01 >= 0.707(|01 > +|10 >)$, $|B10 >= 0.707(|00 > -|11 >)$ and $|B11 >= 0.707(|01 > -|10 >)$ [1]. The generation of Bell states is achieved by using a combination of the Hadamard and the CNOT gates. The Hadamard Gate is essentially a quantum interference gate, capable of transforming pure states into superposed states and vice versa. The matrix formulation of a Hadamard Gate is as follows [1]:

$$[H] = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{4}$$

For QTrick bits, Hadamard Gate is represented as $H = -0.5 \times [Q - 1]$. The CNOT Gate is a two qubit gate, which performs a Pauli X operation on the second qubit (called the target) if and only if the first qubit (called the control)

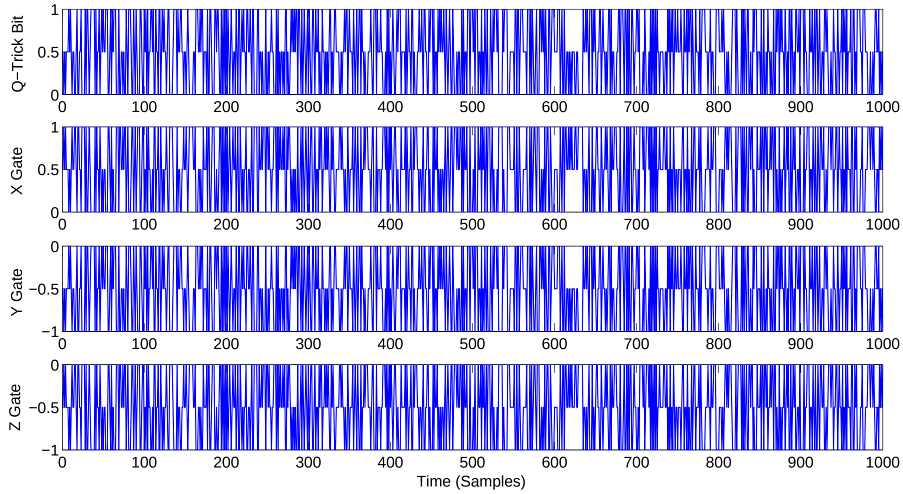

Figure 2: Pauli Gate Implementations for Q-Trick Bits

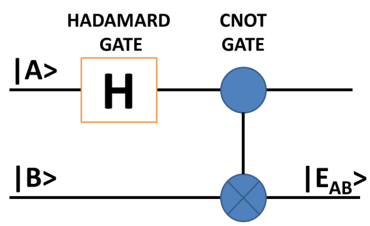

Figure 3: Schematic of Entanglement Operation

is at a $|1>$ state. The operation of the CNOT Gate is exactly similar to the classical Exclusive-OR (XOR) Gate. The matrix representation of the CNOT Gate is as follows [1]:

$$[CNOT] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \quad (5)$$

For two QTrick Bits QC (control) and QT (Taget), $CNOT = QC - QT$. Based on these two gates, the creation of Bell Entangled States is represented as a schematic in Fig. (3), with the Entanglement operation denoted by ENT.

3. QTrick Cellular Automata

In order to formulate a QTrick Cellular Automata, it is first necessary to understand the basic concept of Cellular Automata. A cellular automaton is comprises of cells on a 1D or 2D grid evolving through a time in discrete steps based on a set of 'rules' applied iteratively which determine the state of any cell based on the states of neighboring cells [17]. Cellular automata are typically studied as a possible model for biological systems as well as in other disciplines such as demographics, epidemiology and neural networks [17]. In 2D grids, the most popular cellular automaton is Conway's Game of Life, which is a binary totalistic cellular automaton with a Moore neighborhood of range 1 [17]. It has been shown that this cellular automaton is capable of constructing a Turing Machine, which are universal cellular automata capable of simulating the behavior of any other cellular automata [17].

One Dimensional Cellular Automata, also called Elementary Cellular Automata are the simplest kind of Cellular Automata, where each cell has one of two possible values, 0 or 1 at any instant t, and this value is determined by the values of the cell and its neighbouring cells at the previous instant $t-1$. In a 1D Cellular Automata, the grid is simply depicted as a row vector A, and with the vertical axis denoting time, a 2D spatiotemporal plot depicting the evolution is obtained. With a given cell denoted by $A(y)$, its neighbouring cells are denoted by $A(y-1)$ and $A(y+1)$.

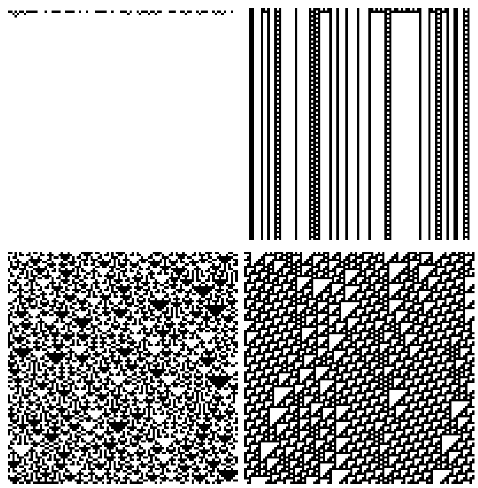

Figure 4: Examples of the four classes of Cellular Automata Rules: Rule 32 (top left), Rule 108 (top right), Rule 150 (bottom left) and Rule 110 (bottom right)

The 'rules', then are given by relations between the current value $A(y,t)$ and the previous values $A(y-1,t-1)$, $A(y,t-1)$ and $A(y+1,t-1)$. Since, in classical digital logic, each cell can take only one of two values, the set of $A(y-1,t-1)$, $A(y,t-1)$ and $A(y+1,t-1)$ is usually written as a 3-bit number, and for each of the eight possible cases (000,001,010,011,100,101,110,111), the value of the output $A(y,t)$ is noted. These eight values are then written together as an 8-bit binary number, with 111 depicting the most significant bit (MSB) and 000 depicting the least significant bit (LSB). This number, converted into decimal, gives the rule number [17]. The evolution of the rule is then typically studied using an initial grid of 0's with the middle cell alone set to 1.

Based on the evolution of patterns, the Cellular Automata rules are typically classified into four classes, as follows [17]:

1. Class 1: All initial patterns evolve into stable, homogeneous state, leading to the disappearance of any randomness. Examples include rules 0, 32, 160 and 232.
2. Class 2: All initial patterns evolve stable oscillating structures filtering out some of the initial randomness, and preserving local changes to the initial pattern. Examples include rules 4, 108, 218 and 250.
3. Class 3: All initial patterns evolve in pseudo-random or chaotic manner, with stable structures quickly destroyed and with local changes to the initial pattern spreading indefinitely. Examples include rules 22, 30, 126, 150, 182.
4. Class 4: All initial patterns evolve into structures that interact in complex ways, forming of local structures that survive for long periods of time. The most popular example of this class is Rule 110.

An example of each of the four classes is given in Fig. (4), with white representing 1 and black representing 0.

Similar to the concept of Cellular Automata briefly described above, the present work proposes the 'Quantum Trick Cellular Automata' (QTCA), which differs from classical cellular automata in two key aspects:

1. The states of cells are not restricted to 0's and 1's; superposed states are allowed. According to the QTrick bit definition explained earlier, a superposed state can be written in the form $a|0>+b|1>$, with $a+b=1$, and thus, superposed state values lie in between 0 and 1. For instance, a superposed state with a at 0.7 and b at 0.3 will have a value of 0.3. Even if the starting initial values of the grid are set to pure states 0 and 1, gates such as the QTrick Hadamard Gate can case the superposed state values to emerge in successive iterations.
2. Since the formation of a finite set rules using superposed states can quickly escalate to a tedious task, the 'rules' in QTCA's are defined using logic operations between $A(y-1,t-1)$, $A(y,t-1)$ and $A(y+1,t-1)$, rather than as 8-bit rule numbers. The logic operations include both classical and quantum gates.

In the present paper, the QTCA patterns generated combinations of the 2-QTrick bit gates CNOT and ENT are explored. In each case, the rule defining $A(y,t)$ is given in two stages, firstly an interaction between $A(y-1,t-1)$ and $A(y+1,t-1)$ using one of the four gates, and then an interaction between the resultant output and $A(y,t-1)$ using one of the four gates. In this manner, the 'pure' patterns obtained for CNOT/CNOT and ENT/ENT are given in Fig. (5).

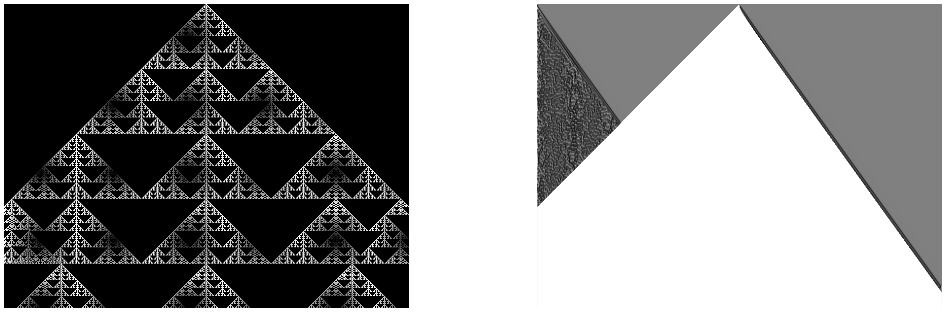

Figure 5: Evolution Patterns of the CNOT/CNOT (left) and ENT/ENT (right) QTCA's

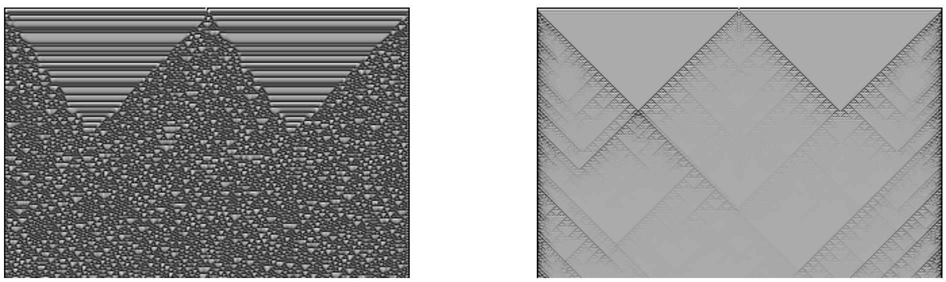

Figure 6: Evolution Patterns of the CNOT/ENT (left) and ENT/CNOT (right) QTCA's

It is seen that while CNOT/CNOT evolution pattern is an ornamental and symmetrical fractal design, while the ENT/ENT rule generates an asymmetrical design with reasonably rich patterns observed on the left extreme cells.

In a similar fashion, the evolution patterns for the mixed rules CNOT/ENT and ENT/CNOT are plotted in Fig. (6). It is seen that while ENT/CNOT patterns consist almost entirely of superposed states, CNOT/ENT rule patterns give rise to highly complex, ornate patterns, comparable to Class 4 and 4 cellular automata.

Finally, the effect of rule symmetry is studied by plotting CNOT/ENT and ENT/CNOT patterns, but with the first operation defined between $A(y,t-1)$ and $A(y+1,t-1)$ and the second operation defined between the resultant and $A(y-1,t-1)$. These are plotted in Fig. (7).

It is seen that the patterns generated using these asymmetric rules differ significantly from the symmetric counterparts, both in directionality and feature size.

Thus, in summary it is seen that Quantum Trick Cellular Automata, defined by the QTrick Operation based rules,

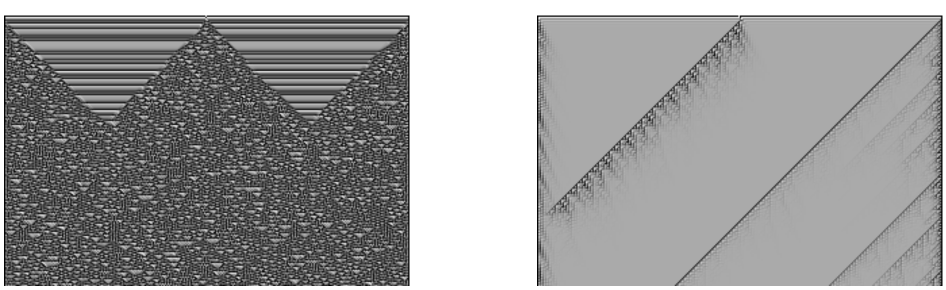

Figure 7: Evolution Patterns of the 'asymmetric' CNOT/ENT (left) and ENT/CNOT (right) QTCA's

generate ornate and complex patterns, as in the case of Cellular Automata, but with an enhanced richness, thanks to the inclusion of superposition states. In particular, it is seen that the entanglement operation is able to generate ornamental patterns of great complexity. Thus, by developing a full-fledged rule set, covering both quantum and classical logic operations, it is possible to enhance the functionalities, capabilities and applications of the cellular automata concept, enabling one to enjoy the intriguing results brought in by quantum operations.

4. Conclusion

A novel approach to exploit the advantages of quantum computation, using Quantum Tricks is proposed. Specifically, the inherent randomness of a qubit is achieved with a pseudorandom number generator, implemented using FPGA. Basic Pauli gates using this formulation are explored, following which Entanglement is studied. After exploring the concept of Cellular Automata, Quantum Trick Cellular Automata (QTCA), allowing superposed states and quantum logic operations is formulated. In this system, rules are written based on the logic operations, and in this context, various QTCA's involving CNOT and Entanglement operations are explored. It is seen that these QTCA's give rise to rich, ornamental patterns, characteristic of the underlying chaos. The extreme simplicity of the proposed design, combined with the efficiency of implementation of cellular automata, producing various rich patterns that can be studied from biological, demographic and networking contexts form the highlights of the present work. It is believed that this will lead to the full-fledged formulation and study of QTrick Cellular Automata Rules, and subsequent development of a Quantum Analogue Turing Machine, thus ushering in a new era of 'Affordable Quantum Computation'.

References

[1] D. McMahon, Quantum Computing Explained, Wiley, 2007.
[2] A. Pathak, Elements of Quantum Computation and Quantum Communication, Taylor and Francis, 2013.
[3] S. Lloyd, Programming the Universe: A Quantum Computer Scientist Takes on the Cosmos, Random House, 2011.
[4] A. Einstein, B. Podolsky, N. Rosen, Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?, Phys. Rev, 47, 777, 1935.
[5] X-S. Ma, Integrated Quantum Photonics: On-Chip Teleportation, Nature Photonics, 8, 749-754, 2014.
[6] W. Rong, Y. G. Hui, Entanglement Teleportation via One Dissipative Quantum Channel, Int. J. Theor. Phys, 53, 3948-3957, 2014.
[7] L. J. Cai, W. Y. Li, Quantum Dense Coding in Trapped Ions, Int. J. Theor. Phys, 1-4. 2014.
[8] J. Clarke, F. K. Wilhelm, Superconducting quantum bits, Nature, 453, 1031-1042, 2008.
[9] M. A. Neilsen, E. Knill, R. Laflamme, Complete quantum teleportation using nuclear magnetic resonance, Nature, 396, 52-55, 1998.
[10] K. G. Fodorov, A. V. Shcherbakova, M. J. Wolf, D. Beckmann, A. V. Ustinov, Fluxon Readout of a Superconducting Qubit, Phys. Rev. Lett, 112, 160502, 2014.
[11] D. J. Egger andF. K. Wilhelm, Optimized controlled-Z gates for two superconducting qubits coupled through a resonator, Supercond. Sci. Technol, 27, 014001, 2014.
[12] L. Shujun, M. Xuanqin and C. Yuanlong, Pseudo-random Bit Generator Based on Couple Chaotic Systems and Its Applications in Stream-Cipher Cryptography, Springer, 2247, 316-329 (2001).
[13] K. Wodkiewicz, J. H. Eberly, Coherent states, squeezed fluctuations, and the SU(2) am SU(1,1) groups in quantum-optics applications, JOSA B, 2, 458-466, 1985.
[14] V. Vedral, Quantum Entanglement, Nature Physics, 10, 256-258, 2014.
[15] B. M. Terhal, Detecting quantum entanglement, 287, 313-335, 2002.
[16] C. Simon, J. Kempe, Robustness of multiparty entanglement, Phys. Rev. A 65, 052327, 2002.
[17] S. Wolfram, A New Kind of Science, Wolfram Media, 2002.