JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

Innovative Approaches to Classical and Quantum Reflected Binary Code Generation using Pascal Triangle, Reversible N-Input C-Gate and Reversible N-Input Q-Gate

[1]Peter Nimbe, [2]Prosper Kandabongee Yeng, [3]Eric Okyere, [4]Dr. Benjamin Asubam Weyori, [5]Prof. Adebayo Felix Adekoya, [6]Dr. Peter Appiahene, [7]Dr. Owusu Nyarko-Boateng, [8]Dr. Isaac Kofi Nti, [9]Dr. Samuel Boateng, [10]Dr. Patrick Kwabena Mensah, [11]Faiza Umar Bawah, [12]Nicodemus Songose Awarayi, [13]Vivian Akoto-Adjepong, [14]Promise Ricardo Agbedanu, [15]Mighty Abra Ayidzoe, [16]Jacob Mensah, [17]Emmanuel Adjei Domfeh

[1]Department of Computer Science and Informatics, University of Energy and Natural Resources, Ghana, Email: peter.nimbe@uenr.edu.gh
[2]Department of Information Security and Communication Technology, Norwegian University of Science and Technology, Norway,
Email: prosper.yeng@ntnu.no
[3]Department of Computer Science and Informatics, University of Energy and Natural Resources, Ghana, Email: eric.okyere@uenr.edu.gh
[4]Department of Computer Science and Informatics, University of Energy and Natural Resources, Ghana, Email: benjamin.weyori@uenr.edu.gh
[5]Department of Computer Science and Informatics, University of Energy and Natural Resources, Ghana, Email: adebayo.adekoya@uenr.edu.gh
[6]Department of Computer Science and Informatics, University of Energy and Natural Resources, Ghana, Email: peter.appiahene@uenr.edu.gh
[7]Department of Computer Science and Informatics, University of Energy and Natural Resources, Ghana,
Email: owusu.nyarko-boateng@uenr.edu.gh
[8]Department of Computer Science and Informatics, University of Energy and Natural Resources, Ghana, Email: isaac.nti@uenr.edu.gh
[9]Department of Computer Science and Informatics, University of Energy and Natural Resources, Ghana, Email: samuel.boateng@uenr.edu.gh
[10]Department of Computer Science and Informatics, University of Energy and Natural Resources, Ghana, Email: patrick.mensah@uenr.edu.gh
[11]Department of Computer Science and Informatics, University of Energy and Natural Resources, Ghana, Email: faiza.bawah@uenr.edu.gh

ABSTRACT
There are numerous classical approaches employed by researchers in the generation of gray code sequences, however currently there is no straightforward method or model for generating quantum gray codes. This paper seeks to present three innovative approaches to generating quantum gray codes. The first approach employs the principle of the Pascal triangle and vectors which are very important concepts drawn from the field of algebra. The second approach known as the "N-input Reversible C-Gate" is based on a reversible XOR gate, a vital concept in classical circuit model of computation. The third approach known as "N-input Reversible Q-Gate" is based on a controlled NOT gate which is a key concept drawn from the field of quantum circuit model of computation. Finally, we assess the performance of the proposed and existing approaches by measuring execution time in terms of number of bits and comparing the results. The Pascal triangle approach to quantum gray code generation requires a longer time to execute as the number of bits rises, according to simulation data and results. The evaluation also shows that the N-input Reversible C-Gate and N-input Reversible Q-Gate gate performs faster than that of the Pascal triangle approach and some of the other existing algorithms.

Index Terms— Controlled NOT Gate, N-input Reversible C-Gate, N-input Reversible Q-Gate, Reversible XOR Gate,

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

I. INTRODUCTION

A binary reflected gray code [1] is a binary sequence in which only one bit position separates two consecutive sequences, and a Gray code is an infinite set of word lists with un-bounded word lengths in which the Hamming distance between any two consecutive words in any list is bounded regardless of word length [1][2]. Gray coding was created to prevent erroneous output from electromechanical switches. When Gray code sequence value is increased or decreased, only one bit is modified, regardless of the number of bits [3]. As a result, the impact of noise spikes is reduced [4].

Gray code counters only alter one bit for every increase or reduction in value, reducing the noise effect. Because the fingers aren't always exactly aligned, gray codes are employed instead of other binary sequences. If two bits changed at the same time, one finger would detect the change before the other, resulting in an unexpected malfunction [1]. For example, the binary reflected Gray code is a method of registering all n-bit binary numbers so that consecutive values differ by exactly one bit. A method like this is intended to provide a two-fold gain [5].

Gray codes are preferred over binary codes because they have the unique attribute of only changing one bit when a number is handed on to its successor or processor, removing the possibility of a false reading [6]. Gray codes are lists of instances of a combinatorial item that are different per a given closeness condition including consistent change, and are a vital part of combinatorial production [7]. Position sensors that are mechanical in nature employ them to transform the angular position of a disk to a digital format [8].

Gray codes are used in a variety of applications, including puzzle solving such as the Tower of Hanoi (Savage, 1997), permanent computation [9], circuit testing, image processing [10], hashing [11][12], storing extractions, and Venn diagram classification [10][13]. They are increasingly used to aid in digital communication error correction, such as digital terrestrial television.

Related works

A way of expressing numbers in base 2 is gray code. In the conventional system of base 2, the numbers are 000, 001, 010, 011, 100, 101, 110 and 111 for 3 bit binary gray code. This is in sharp contrast to the gray binary code where 3 bit equals 000, 001, 011, 010, 110, 111, 101, and 100. In the conventional system, from 001 we move to 010 where the unit’s place turn out to be 0 from 1 and the next digit becomes 1 from 0. Frank Gray used the gray code concept in his patent submittal on Pulse Code Communication [14]. A diagram of the pulse code communication is shown below in Figure. 1 below.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

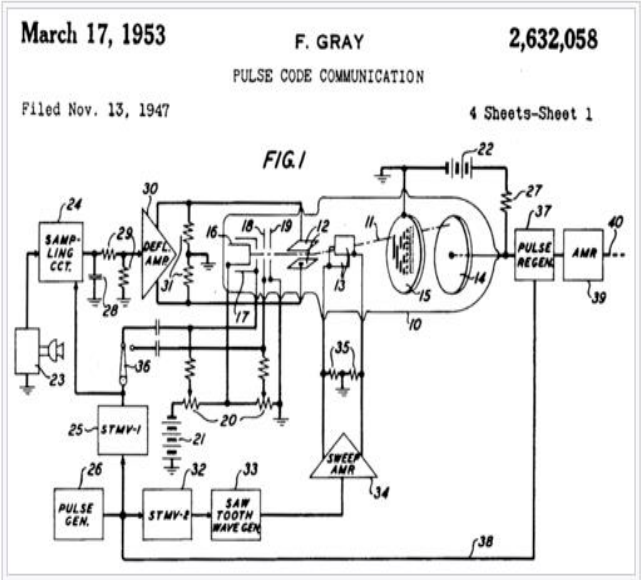


Figure 1: A portion of the front page of Gray's patent [14].

One application of gray codes is in communication where error correction is pivotal. It's vital that the legendary Donald Knuth used the gray code algorithm in the formulation of tuples in his Art of Computer Programming Vol4A. Gray codes of length n can be generated and converted into conventional binary [15].

Another application of gray codes is in encryption, databases, and puzzles. The binary reflected gray code is standard in nature even though there are other minimal change orderings on tuples that are binary in nature and combinatorial objects and they include monotonic, balanced, single-track and long-run gray codes. Gray codes exist in other forms or types, like Beckett-Gray code [16]. Gray codes are used in analog-to-digital encoders, where a rotating wheel's angular position is ascertained by encoding some values read off of nn concentrically-arranged tracks. Gray codes are used in Karnaugh maps, error detection and rotary and optical encoders [16]. The rotary encoder diagram is shown below in fig. 2 below.

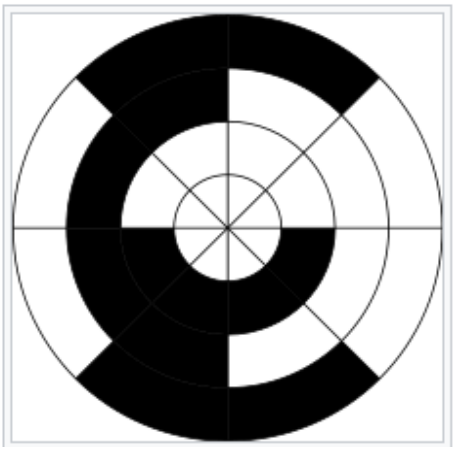


Figure 2: Rotary encoder (Wikipedia, n.d).

They are also employed in the interconnection network theory for the minimization of the dilation of linear arrays of processors [18]. Another application of gray codes is in p-sequences where binary trees are coded. This serves as an alternative representation to well-formed parentheses strings [19]. Gray codes are also

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

employed in quasi-gray codes where a sequence of bit strings of length d differ from the next in a constant number of bits [20]. Some other practical applications of gray codes include; formation of a Hamiltonian cycle on a hypercube, where each bit corresponds to one dimension, minimization of the errors in digital-to-analog signals conversion especially in sensors, solving the Towers of Hanoi problem and utilization in genetic algorithms theory [21].

Gray code generation is quite simple from binary because the length of the gray binary string is the same as the length of the conventional binary version. The leftmost bit remains unchanged, hence starting from the second position from the left, the formula is given by

$Gi = XOR(Bi,Bi-1)$

Where Bi is the bit in the i^{th} position of the gray code starting from the leftmost part, Gi is the bit in the i^{th} position of the gray code [22]. In time past, it was appropriate and convenient to utilize gray codes with dissimilar properties from binary reflected code. Short codes in the range of 32 to 256 element could be generated systematically but now a technique has been devised where both long and short gray codes are generated. Codes of various lengths are generated having desired properties of equal column change counts [23]

There are a number of ways gray binary codes have been generated which include algorithms, libraries, techniques etc. Two recursive techniques for creating binary reflected gray code were described by Er. The first technique simulates the binary reflected code, while the second expresses the bit transition positioning explicitly. The first algorithm operates at an O(2) per code word speed, while the second operates at an O(1) per code word pace. Other iterative techniques for creating binary reflected gray code cannot compare to the efficiency of these approaches [24].

There is a tiny library in Java for generating a gray code over n bits. This is however subject to a constraint where two adjacent bit word differ in only one position. There approaches to generating gray codes in C. One of such is where a palindrome number of length (2n)-1 is generated after elements are pushed into a stack [25].

In 1981, Robinson & Cohn described a recursive method for creating balanced gray codes. When two gray codes are obtained one from the other via the permutation and complementation procedures, they are said to be of the same type [26]. Gray codes are used to express a specific Hamiltonian path with permutation and complementation operations matching to symmetric cube operations [44].

Ali et al proposed two techniques, namely MOptimal and Backtracking that can form a gray code sequence of n bits. The technique based on backtracking produces a sub-tree. By the concatenation of 0 and 1 to the MSB position of the n-1 bit results, the backtracking method generates the n-bit reflection and sequence [13]. A variation to that known as MOptimal, is a variation of the time and space optimal approach that considers the inner loop executions and the total number of outer [13].

Phillips & Wick devised a recursive way to creating n-bit binary gray codes, in which a recurrence relation determines the gray code for the n bit system. However, there were certain inefficiencies because the recursive sub-problems answered were interdependent [27]. A dynamic programming approach for creating n bit gray code sequences was presented to minimize these inefficiencies and dependencies, where sub-problems of the main problem were solved just once.
Gray codes can also be segmented into N-ary reflected gray codes. To generate the sequence, two recursive techniques were provided. One approach is derived from the digit sequencing orders in N-ary reflected gray codes, while the other is derived straight from the definition. Further proof was constructed, and it was determined that the N-ary technique is cyclic when the radix is even, but not cyclic when the radix is odd, in general [28].

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

Vickers & Silverman provided a method for creating some specialized gray code with qualities such as equalized change numbers for general logic circuits. A block-insertion technique is used, which is led by a scalar quality measure [29].

Another method for creating a gray code for an odd-length sequence utilizing a virtual space was previously discussed [30]. The reflect and prefix methods can also be used to construct an n-bit gray code iteratively. The processes involved include; the creation of code for n=1 using the 0 and 1 codes, reversal of the preceding code: 0 and 1, addition of the following inverted codes to the list: 0, 1, 1, and 0 and then the addition of the 0 prefix for the old code and the 1 prefix for the new code: 00, 01, 11, and 10. Table 1 shows the gray codes for 1, 2 and 3-bit binary reflected codes.

Table 1: Gray codes for 1, 2 and 3-bit (John, 2019)

For n = 1 bit	Gray	000	001	011	010	110	111	101
	Binary	000	001	010	011	100	101	110
For n = 2 bit	Gray	00	01	11	10			
	Binary	00	01	10	11			
For n = 3 bit	Gray	0	1					
	Binary	0	1					

Ali et al. devised a programming technique and linked a storage efficient data structure with the capacity to generate a whole n-bit binary reflected gray code sequence in space and time. The technique took advantage of the innate redundancy in the gray code sequence elements to avoid another computation of repetitive subsequences. Their algorithm was found to suffer from the fact that the recursive sub-problems solved throughout the execution of the algorithm were dependent of one another [13].
Mütze & Nummenpalo stated that a gray code at the middle levels is a cyclical enumeration of all bit strings of length 2n+1 with either n or n+1 entries equal to 1 such that any two consecutive bit strings in the list differ in exactly one bit for any integer n . They also provided an efficient algorithm to compute a gray code at the middle levels that sort to produce an efficient space and time algorithm to compute a middle levels gray code [31]. Because of the limits of today's quantum hardware, creating algorithms that make the most of what's available is very critical.

Di Matteo et al examined an efficient encoding that uses a set of basis states, in which terms in the Hamiltonian are translated to qubit operators using a Hamiltonian that works on the basis states in gray code order. With a simulated Variational Quantum Eigensolver, this encoding is used in the commonly studied issue of calculating the ground-state energy of a deuteron (VQE) [32].

Ali et al. employed two techniques to generate a complete n-bit binary gray code sequence: backtracking and MOptimal. Backtracking generates the n-bit sequence and its reflection by joining a "0" and a "1" to the most significant bit position of the n-1 bit result created at each leaf node of the sub-tree, whereas MOptimal emphasizes space and time reduction. Finally, they measured the execution time in terms of the amount of bits and compared the findings to compare the performance of the proposed and current algorithms.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

Quantum Dot Cellular Automata (QCA) consider fault-tolerance as an important aspect while exploring and manufacturing quantum dots. For 2, 3, 4 or 5-bit QCA-based systems, [33] presented a gray code converter. The suggested designs minimize the number of cells (57.5 percent for 2-bit, 62.19 percent for 3-bit, and 64.28 percent for 4-bit) and area when compared to the existing circuit (50 percent for 2-bit, 63.63 percent for 3-bit, and 53.33 percent for 4-bit). The Area Utilization Factor, an integrated measure, was used to evaluate all of the designs (AUF).

Chang proposed a loopless technique for producing all of these Gray code Z-sequences. As a result, generating one Z-sequence had a time complexity of O(1), and their algorithm used 2n+O space (1). They also supplied algorithms for ranking and unranking depending on this ordering. The ranking method takes O (maxkmn, n2) time and takes up O (kmn) space, while the unranking algorithm takes O(kmn2) time and takes up O (kmn) space (kmn2). Their investigation did not reveal whether or not elements of the recursion table can be gotten in real time during the unranking process if the table is not prepared forehand [34].

He et al. developed a two-defocused binary pattern quaternary gray-code phase unwrapping technique. A weighted optimization approach for the construction of unusual binary code patterns was described in their study. With a defocused projector, the specified binary patterns were projected onto the item, resulting in quaternary patterns. After being taken by a camera, the deformed code patterns will be subjected to our suggested normalization-denoising-clustering technique to recover the ideal gray codes. Simulations and tests are used to illustrate the efficacy of the proposed strategy [35].

A general framework for generating gray codes for weak orders was created by Jacques et al. Their paper provided a simple framework-based technique that generates cyclic 2-gray codes for weak orders in constant amortized time per string, which are the first cyclic 2-gray codes for weak orders ever discovered. Their architecture might be tweaked to generate different gray codes for weak orders, and they show how to use it to formulate first shift Gray codes for weak orders in constant amortized time per string, when successive strings differ by a shift or a symbol change [36].

Konstantinova & Medvedev introduced a new idea of prefix-reversal gray codes based on independent cycles, which builds on Zaks and Williams' greedy prefix-reversal gray code constructions [37].

A novel form of rotary absolute encoder disk design based on N-ary cyclical gray coding has been proposed by Paul & Chang. In comparison to typical gray code tracks, the suggested coded disk design results in a downsizing of the coded track with better resolution. The suggested encoder coded track design allows the encoder resolution expression to include both the base and power terms of the denominator. However, their effort did not include the development of the single track N-ary gray code, as well as the sensing system for the coded disk and the data gathering system [38].

Gray code applications

By using a gray code generator, a test pattern creation system was created by Dilip et al. The usage of a gray code generation cuts the time and resources required to test a DUT in half. Gray code is employed in several areas such as FIFO technique and state diagram. This is possible due to the one-bit toggle nature likened to binary values. The suggested system creates test patterns that toggle by one bit, giving each pattern a one-bit difference. Overall, this study discovered no solutions to shorten the time it takes to generate test patterns [39].

Salgado-fuentes et al proposed a new way for determining a propositional formula that de-scribes a switching system problem by applying Gray Code techniques to generate numerous truth tables based on an original one. Each permutation was connected in a hyper volume, and each
node was represented by a bit combination. MATLAB was used to construct an algorithm and compared to results from the software Boole-Deusto to check and evaluate the method's applicability [40].

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

Gray codes' error-correction coding (ECC) performance in safeguarding spatial picture watermarks against lossy data compression was studied by Kimoto et al [41]. The variances between the Gray code word bot patterns are evaluated in depth. Based on the qualities, the Gray-ECC approach for storing watermark bits in Gray code words that indicate signal levels using a single-error-correcting (SEC) code was created. However there was no detail regarding how the system should adjust to different degrees of visual smoothness [41].

To improve network security, Sidhi recommended using a grey code converter at each connection edge in an internal network. If required, this mechanism can be employed by not just the gray code converter, but also another encryption and decryption system. Only on the physical layer and within the internal network can the upgrade guard against network breaches like phishing [42]. This enhanced security can help protect the data that travels through the cable, preventing bandwidth theft. The creation of a switch or hub with the option of data transfer via toggle for altering the format of data sent through the cable, on the other hand, was not included in this design [42].

Problem, Objective, Contribution

There are many classical approaches for gray code generation, however there is no approach that make use of the Pascal triangle concept. Furthermore there is no simple and established approach for quantum gray code generation. Quantum computing as a field is currently undergoing lots of standardizations with research ongoing. The objective of this paper is to propose an innovative approach to generating gray codes from a classical and quantum perspective.

2 Method

Quantum Perspective 1

To generate the n-bit gray code sequences, the ideas of non-deterministic finite state automata and Pascal triangle were used. Complete binary trees were used to create the non-deterministic finite state automata. The Pascal triangle and non-deterministic finite state automata are utilized because they can model non-deterministic states or objects. The principles that govern the quantum realm are regulated by a completely different reality. And this reality is intangible, non-deterministic, and invisible. In other words, quantum systems are inherently non-deterministic. Hence concepts that can guarantee or depict non-determinism and can generate gray codes at various levels of n are utilized. The non-deterministic finite state automata diagrams below in Figures 3-6 were created using JPLAP, a Java software for modeling and experimenting with formal language ideas such as automata, Turing machines, mealy machines, Moore machines, and transition systems.

Quantum Perspective 2

To generate the n-bit gray code sequences, an N-input reversible C-Gate is used. This gate is an innovative gate proposed by the authors in the generation of quantum gray codes together. The N-input reversible C-Gate gate used in this paper is an N-bit operation where the number of possible input states is equal to two to the power of the number of inputs i.e. number of input possible states is equal to 2N. In another perspective, an N-input reversible Q-Gate, which is an N-qubit operation is used to generate the n-bit gray code sequences.

2.1 Non-Deterministic Finite State Automaton

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

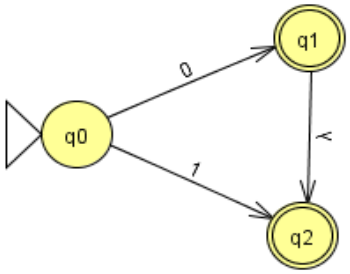


Figure 3: NDFA for 1 bit Binary Gray Code Sequence

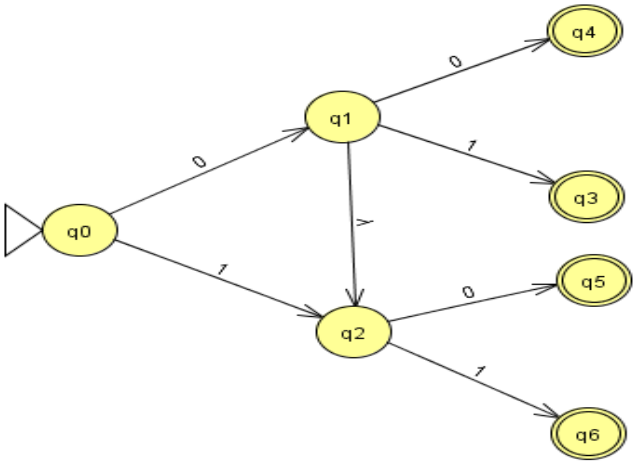


Figure 4: NDFA for 2 bit Binary Gray Code Sequence

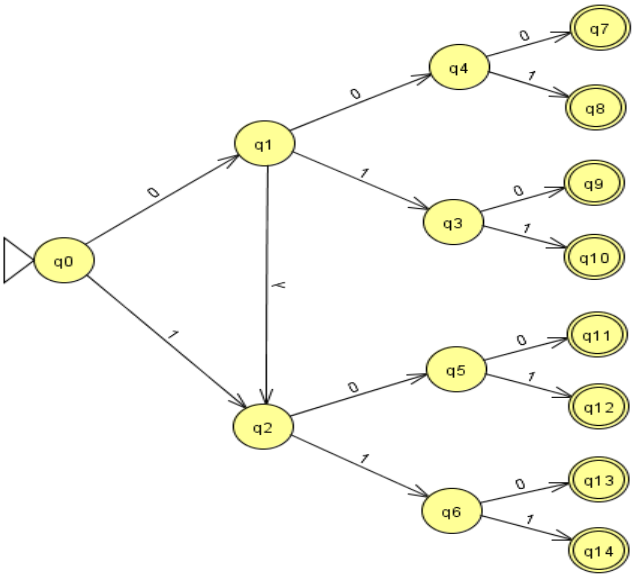


Figure 5: NDFA for 3 bit Binary Gray Code Sequence

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

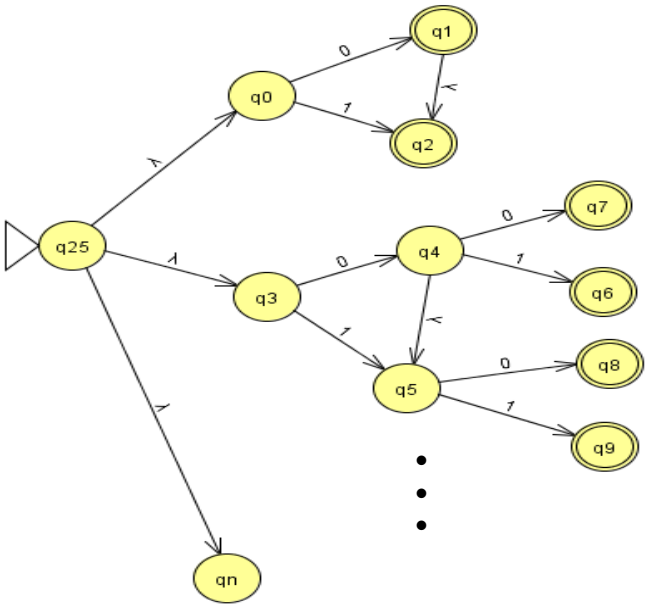


Figure 6: NDFA for n-bit Binary Gray Code Sequence

There are no minimizations for NDFA's due to the fact that there is no polynomial-time algorithm to minimize general NFAs unless P=PSPACE, an unsolved conjecture is resolved. However a brute force search might work nevertheless, there might be more efficient methods. The DFA and NFA's above generates both n-bit gray code and standard n-bit binary sequences, but the movements or traversals in the graph will result in either one of them.

2.3 Pascal Triangle Approach

A framework is presented for N-bit gray code generation and implemented using the C++ programming language. Proven mathematical concepts were used in the construction process, hence can ensure an effective generation algorithm. The mathematical concept of vectors is used in conjunction with the concatenation operator where each qubit is considered as vector. This concept is used for the presentation of both classical and quantum perspectives.

The basic unit of quantum data is a qubit which exist in a superposition of states. They are represented using Dirac notations. Quantum bits to classical bits relate by a formula 2^n, making the rate of growth to be exponential [43].

1 qubit = 2^1 = 2 bits = |0>, |1>
2 qubits = 2^2 = 4 bits = |00>, |01>, |10>, |11>
3 qubits = 2^3 = 8 bits = |000>, |001>, |010>, |011>, |100>, |101>, |110>, |111> [43].

Quantum Perspective 1:
Generating gray code using vectors and concatenation operator (&).
1 qubit = |1> = $\binom{0}{1}$ 0 qubit = |0> = $\binom{1}{0}$

For n=1, we have
$\binom{0}{1}, \binom{1}{0}$

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

For n=2, we have

$$\binom{0}{1} \& \binom{1}{0} = \begin{pmatrix} 01 \\ 00 \\ 10 \\ 11 \end{pmatrix}$$

This implies that 1 qubit & 0 qubit = 2 qubit
OR

$$\binom{1}{0} \& \binom{0}{1} = \begin{pmatrix} 10 \\ 11 \\ 00 \\ 01 \end{pmatrix}$$

This implies that 0 qubit & 1 qubit = 2 qubit

For n=3, we have

$$\binom{0}{1} \& \begin{pmatrix} 01 \\ 00 \\ 10 \\ 11 \end{pmatrix} = \begin{pmatrix} 001 \\ 000 \\ 010 \\ 011 \\ 101 \\ 100 \\ 110 \\ 111 \end{pmatrix}$$

This implies that 1 qubit & 2 qubit = 3 qubit
OR

$$\binom{1}{0} \& \begin{pmatrix} 01 \\ 00 \\ 10 \\ 11 \end{pmatrix} = \begin{pmatrix} 101 \\ 100 \\ 110 \\ 111 \\ 001 \\ 000 \\ 010 \\ 011 \end{pmatrix}$$

This implies that 0 qubit & 2 qubit = 3 qubit

Deduction

From the concept and approach above using vectors and concatenation, the following mathematical and logical deductions are made and presented below:

0 qubit & N qubit = N+1 qubit -------- A
1 qubit & N qubit = N+1 qubit -------- B

Equating A and B, we have
0 qubit & N qubit = 1 qubit & N qubit

Now, employing the concepts of vectors and concatenation in the Pascal triangle, we have the following below.

Figure 7 below is a Pascal triangle with 5 rows.

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

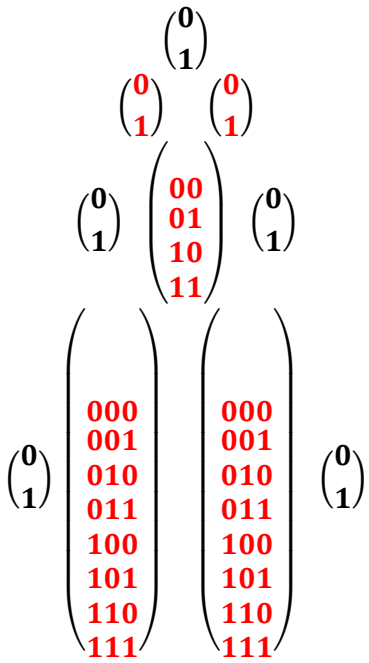
Figure 7: Classical Pascal triangle with 5 rows

Expressing it in the form of qubits, we have Figure 8 below

```
                |1>
            |1>      |1>
        |1>     |2>     |1>
    |1>     |3>     |3>     |1>
|1>     |4>     |6>     |4>    |1>
```

Figure 8: Quantum Pascal triangle with 5 rows

Mimicking the Pascal triangle above using qubit vectors, we have what is shown in Figure 9 below.



JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

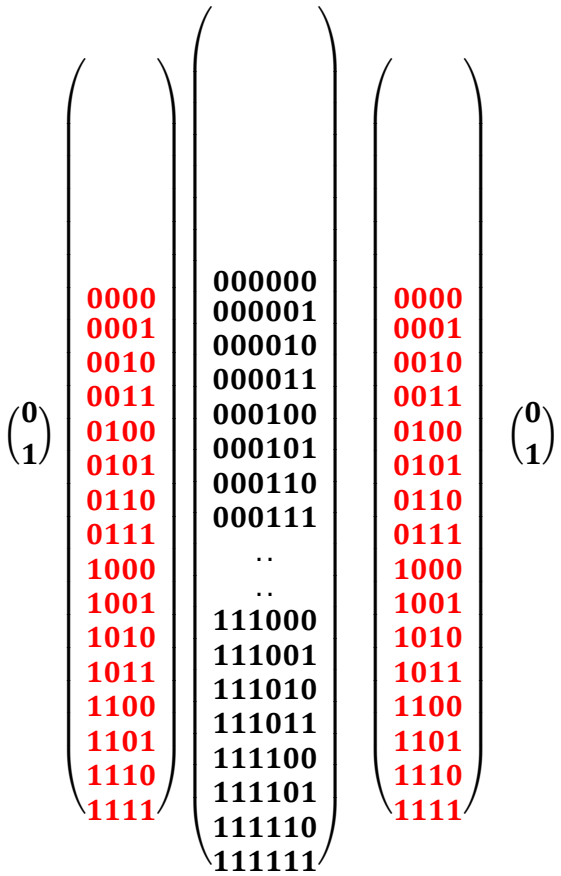


Figure 9: Pascal triangle using qubit vectors.

By observing the Gray code returned when n=1, 2, 3, we can find this rule:

1. The Gray code at n+1 is the Gray code of n & (0 qubit vector or 1 qubit vector).
2. The number of qubit strings corresponding to Gray code at n+1 is twice the length of the number of qubit strings corresponding to Gray code at n.
3. The Pascal triangle approach needs to be optimized for good performance and low execution times. This is because currently the performance in terms of running time tends to deteriorate when the value of n increases. Hence using either Equation A or Equation B for gray code generation for any given value of n is more optimal than using the Pascal triangle approach.

Optimized Algorithm

Set A = $\begin{pmatrix}0\\1\end{pmatrix}$

Set A = $(\ )$

For i = 0 to n-1
  C = Concate(C, A)
End For

Where Concate is a function for performing the concatenation of vectors.

2.4 Classical Approach

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

The classical approach to generating gray code employs an innovative technique based on an N-input Reversible C-Gate. This gate operates similar to the Reversible XOR Gate but with some variation pertaining to the number of inputs and outputs. Figure 10 below shows a diagram of the original reversible XOR Gate. The plus (+) symbol used in Tables 2-5 indicates an XOR Gate operation.

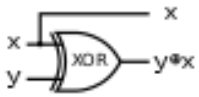


Figure 10: Reversible XOR Gate [46]

The classical gray codes is generated below for n=1 to 4 in Tables 2-5.

Table 2: Gray code generation for n=1

Input x	Output x
0	0
1	1

x → x

Gray for n=1 is 0, 1 and is obtained by repeating the value x. Figure 11 shows the circuit corresponding to Table 2.

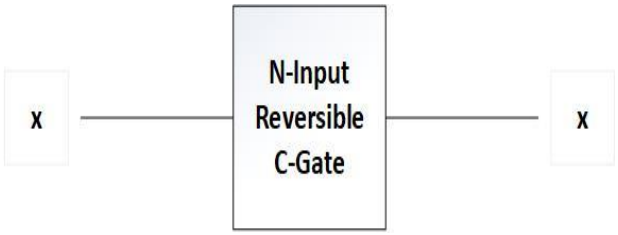


Figure 11: Circuit for transforming classical input to output for n=1

To generate gray code for n=2, the output x in Table 2 is passed to Table 3 as input. The same input is repeated for the 3rd and 4th row of Table 3. For the input values y, the 1st and 2nd row is 0 and 3rd and 4th row is 1.

Table 3: Gray code generation for n=2

Input		Output	
x	y	x	x+y
0	0	0	0
1	0	1	1
0	1	0	1
1	1	1	0

x y → x x+y

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

Gray code for n=2 is 00, 11, 01, 10 and is obtained by concatenating x and x+y in the output column of the table. Figure 12 shows the circuit corresponding to Table 3.

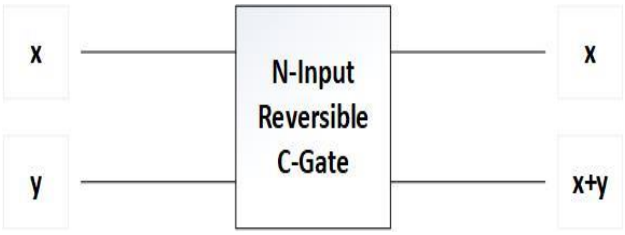


Figure 12: Circuit for transforming classical input to output for n=2

To generate gray code for n=3, the output x and x+y in Table 3 is passed to Table 4 as input where it becomes x and y respectively. The same input is repeated for the 5th, 6th, 7th, and 8th row of Table 4. For the input values z, the 1st, 2nd, 3rd and 4th row is 0 and 5th, 6th, 7th, and 8th row is 1.

Table 4: Gray code generation for n=3

Input			Output		
x	y	z	x	y	(x+y)+z
0	0	0	0	0	0
1	1	0	1	1	0
0	1	0	0	1	1
1	0	0	1	0	1
0	0	1	0	0	1
1	1	1	1	1	1
0	1	1	0	1	0
1	0	1	1	0	0

x y z→ x y (x+y)+z

Gray for n=3 is 000, 110, 011, 101, 001, 111, 010, 100 and is obtained by concatenating x, y and y+x in the output column of the table. Figure 13 shows the circuit corresponding to Table 4.

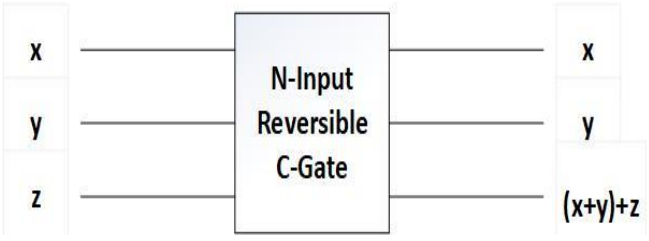


Figure 13: Circuit for transforming classical input to output for n=3

To generate gray code for n=4, the output x, y and x+y in Table 3 is passed to Table 4 as input where it becomes x, y and z respectively. The same input is repeated for the 9th, 10th, 11th, 12th, 13th, 14th, 15th, and 16th row of Table 4. For the input values z, the 1st, 2nd, 3rd, 4th, 5th, 6th, 7th and 8th row is 0 and 9th, 10th, 11th, 12th, 13th, 14th, 15th and 16th row 1.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

Table 5: Gray code generation for n=4

Input				Output			
x	y	z	a	x	y	z	((x+y)+z)+a
0	0	0	0	0	0	0	0
1	1	1	0	1	1	1	1
0	1	1	0	0	1	1	0
1	0	0	0	1	0	0	1
0	0	1	0	0	0	1	1
1	1	0	0	1	1	0	0
0	1	0	0	0	1	0	1
1	0	1	0	1	0	1	0
0	0	0	1	0	0	0	1
1	1	1	1	1	1	1	0
0	1	1	1	0	1	1	1
1	0	0	1	1	0	0	0
0	0	1	1	0	0	1	0
1	1	0	1	1	1	0	1
0	1	0	1	0	1	0	0
1	0	1	1	1	0	1	1

x y z a → x y z ((x+y)+z)+a

Gray for n=4 is 0000, 1111, 0110, 1001, 0011, 1100, 0101, 1010, 0001, 1110, 0111, 1000, 0010, 1101, 0100, 1011 and is obtained by concatenating x, y, z and x+y+z+a in the output column of the table. Figure 14 shows the circuit corresponding to Table 5.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

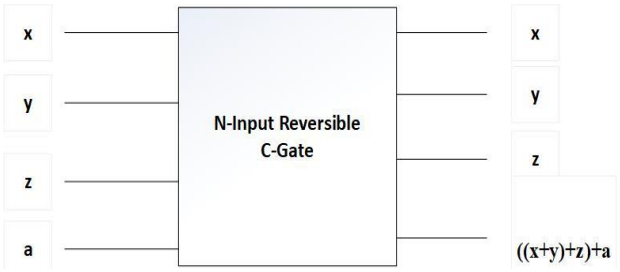


Figure 14: Circuit for transforming classical input to output for n=4

2.4 Quantum Approach

The quantum approach to generating gray code employs an innovative technique based on an N-input Reversible Q-Gate. This gate operates similar to the C-NOT Gate but with some variation pertaining to the number of inputs and outputs. Figure 11 below shows a diagram of the original C-NOT Gate. The plus (+) symbol used in Tables 6-9 indicates a C-NOT Gate operation.

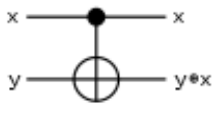


Figure 15: C-NOT Gate [46]

The quantum gray codes is generated below for n=1 to 4 in Tables 6-9.

Table 6: Gray code generation for n=1

Input x	Output x
|0>	|0>
|1>	|1>

x → x

Gray for n=1 is |0>, |1> and is obtained by repeating the value x. Figure 16 shows the circuit corresponding to Table 6.

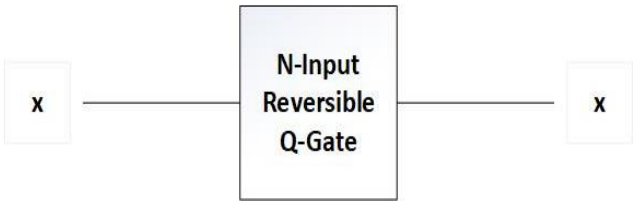


Figure 16: Circuit for transforming quantum input to output for n=1

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

To generate gray code for n=2, the output x in Table 6 is passed to Table 7 as input. The same input is repeated for the 3rd and 4th row of Table 7. For the input values y, the 1st and 2nd row is |0> and 3rd and 4th row is |1>.

Table 7: Gray code generation for n=2

Input x y	Output x x+y
|0> |0>	|0> |0>
|1> |0>	|1> |1>
|0> |1>	|0> |1>
|1> |1>	|1> |0>

x y → x y+x

Gray for n=2 is |00>, |11>, |01>, |10> and is obtained by concatenating x and x+y in the output column of the table. Figure 17 shows the circuit corresponding to Table 7.

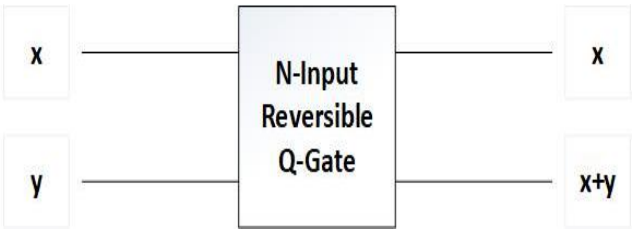


Figure 17: Circuit for transforming quantum input to output for n=2

To generate gray code for n=3, the output x and x+y in Table 7 is passed to Table 8 as input where it becomes x and y respectively. The same input is repeated for the 5th, 6th, 7th, and 8th row of Table 8. For the input values z, the 1st, 2nd, 3rd and 4th row is |0> and 5th, 6th, 7th, and 8th row is |1>.

Table 8: Gray code generation for n=3

Input x y z	Output x y (x+y)+z
|0> |0> |0>	|0> |0> |0>
|1> |1> |0>	|1> |1> |0>
|0> |1> |0>	|0> |1> |1>
|1> |0> |0>	|1> |0> |1>

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

|0> |0> |1>	|0> |0> |1>
|1> |1> |1>	|1> |1> |1>
|0> |1> |1>	|0> |1> |0>
|1> |0> |1>	|1> |0> |0>

x y z→ x y (x+y)+z

Gray for n=3 is |000>, |110>, |011>, |101>, |001>, |111>, |010>, |100> and is obtained by concatenating x, y and x+y in the output column of the table. Figure 18 shows the circuit corresponding to Table 8.

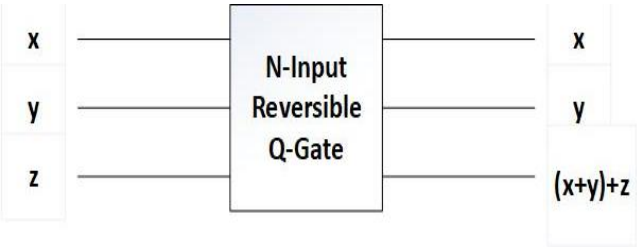


Figure 18: Circuit for transforming quantum input to output for n=3
To generate gray code for n=4, the output x, y and x+y in Table 8 is passed to Table 9 as input where it becomes x, y and z respectively. The same input is repeated for the 9th, 10th, 11th, 12th, 13th, 14th, 15th, and 16th row of Table 4. For the input values z, the 1st, 2nd, 3rd, 4th, 5th, 6th, 7th and 8th row is 0 and 9th, 10th, 11th, 12th, 13th, 14th, 15th and 16th row 1.

Table 9: Gray code generation for n=4

Input				Output			
x	y	z	a	x	y	z	((x+y)+z)+a
|0>	|0>	|0>	|0>	|0>	|0>	|0>	|0>
|1>	|1>	|1>	|0>	|1>	|1>	|1>	|1>
|0>	|1>	|1>	|0>	|0>	|1>	|1>	|1>
|1>	|0>	|0>	|0>	|1>	|0>	|0>	|0>
|0>	|0>	|1>	|0>	|0>	|0>	|1>	|1>
|1>	|1>	|0>	|0>	|1>	|1>	|0>	|0>
|0>	|1>	|0>	|0>	|0>	|1>	|0>	|0>
|1>	|0>	|1>	|0>	|1>	|0>	|1>	|1>
|0>	|0>	|0>	|1>	|0>	|0>	|0>	|1>

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

|1> |1> |1> |1>	|1> |1> |1> |0>
|0> |1> |1> |1>	|0> |1> |1> |0>
|1> |0> |0> |1>	||1> |0> |0> |1>
|0> |0> |1> |1>	|0> |0> |1> |0>
|1> |1> |0> |1>	|1> |1> |0> |1>
|0> |1> |0> |1>	|0> |1> |0> |1>
|1> |0> |1> |1>	|1> |0> |1> |0>

x y z a → x y z ((x+y)+z)+a

Gray for n=4 is |0000>, |1111>, |0110>, |1001>, |0011>, |1100>, |0101>, |1010>, |0001>, |1110>, |0111>, |1000>, |0010>, |1101>, |0100>, |1011> and is obtained by concatenating x, y, z and x+y+z+a in the output column of the table. Figure 19 shows the circuit corresponding to Table 9. Figure 19: Circuit for transforming quantum input to output for n=4.

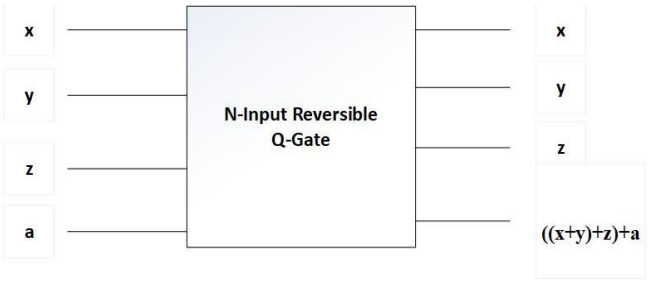


Figure 19: Circuit for transforming quantum input to output for n=4

Hence for the Input to Output transformation, the following algorithm below is used

For n=1: x → x
For n=2: x, y → x, (x+y)
For n=3: x, (x+y), z → x, (x+y), (x+y)+z
For n=4: x, (x+y), (x+y)+z, a → x, (x+y), (x+y)+z, ((x+y)+z)+a

The approach displays the values in a table or nxm dimensional array which affects performance in terms of running time. In an attempt to optimize the performance for this approach, some patterns or observations are drawn from table 2 to 9.

	1st Input	2nd Input	3rd Input	4th Input
N=1	0 1			
N=2	0 1 0	0 0 1		

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

	1	1		
N=3	0	0	0	
	1	1	0	
	0	1	0	
	1	0	0	
	0	0	1	
	1	1	1	
	0	1	1	
	1	0	1	
N=4	0	0	0	0
	1	1	1	0
	0	1	1	0
	1	0	0	0
	0	0	1	0
	1	1	0	0
	0	1	0	0
	1	0	1	0
	0	0	0	1
	1	1	1	1
	0	1	1	1
	1	0	0	1
	0	0	1	1
	1	1	0	1
	0	1	0	1
	1	0	1	1

3. Results

3.1 Gray code Implementation using Pascal triangle

```
// Generating n-bit gray codes
#include <iostream>
#include <bits/stdc++.h>
using namespace std;

// Generation of n-bit gray codes
void graycode_gen(int num)
{
    // power of 2
   for (int c = 0; c < (1 << num); c++)
   {
      // Using bitset to convert the decimal values of gray codes to binary
      int val = (c ^ (c >> 1));

      // Using bitset
      bitset<32> d(val);

      // Converting to string
      string h = d.to_string();
```

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

```
      if(c!=(1 << num)-1)
        cout <<"|"<< h.substr(32 - num) << ">,";
      else
        cout <<"|"<< h.substr(32 - num) << ">";
   }
}

//Function to generate gray code pascal triangle
void graycode_pascaltriangle()
{
 int hors,intial=1,tick,count1,count2;

 cout<<"Enter number of rows: ";
 cin>>hors;

 for(count1=0;count1<hors;count1++)
 {
  for(tick=1;tick<=hors-count1;tick++)
   cout<<"  ";
  for(count2=0;count2<=count1;count2++)
  {
   if (count2==0||count1==0)
    intial=1;
   else{
    intial=intial*(count1-count2+1)/count2;
   }

   cout<<setw(5);
   graycode_gen(intial);
  }
  cout<<endl;
 }
}

// Testing the gray code pascal triangle function
int main()
{
  graycode_pascaltriangle();

  return 0;
}
```

Grey Code Function

Running Time (Big-O): O(2n)
Auxiliary Space: O(n)

Pascal Triangle Function

Running Time (Big-O): O(n)*O(2n)
Auxiliary Space: O(n)
Total Time Complexity: O(2n) + O(n)*O(2n)

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

For n=1, we have:

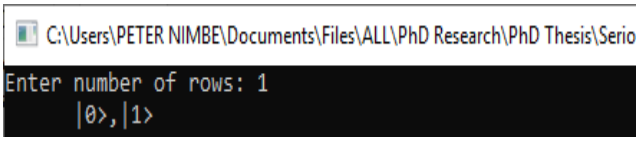


Figure 10: Qubit vector - Pascal Triangle when n equals 1

For n=2, we have:

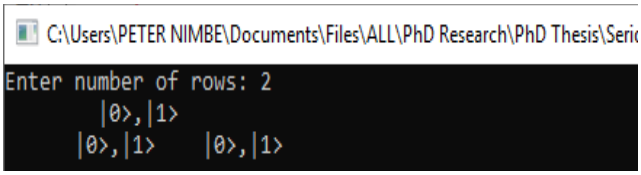


Figure 11: Qubit vector - Pascal Triangle when n equals 2

For n=4, we have:

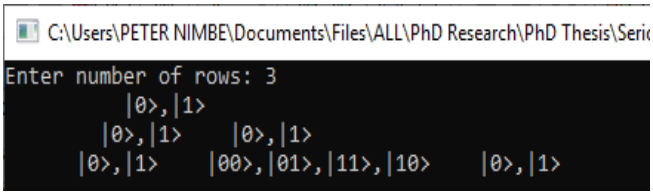


Figure 12: Qubit vector - Pascal Triangle when n equals 3

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

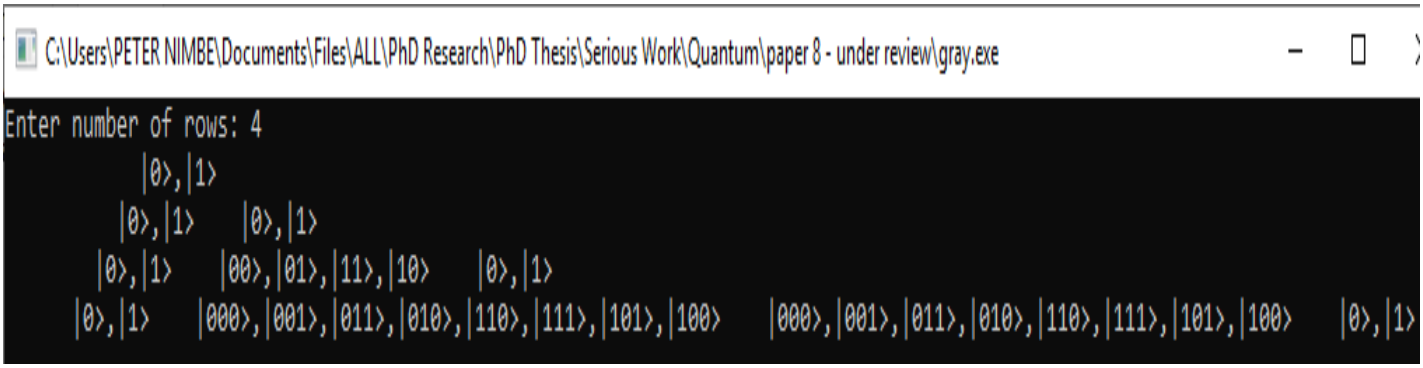


Figure 13: Qubit vector - Pascal Triangle when n equals 4

3.2 Gray Code Implementation using N-Input Reversible C-Gate and N-Input Reversible Q-Gate

3.4 Performance evaluation of gray code algorithms

The implementation of the innovative approaches using Pascal triangle, N-Input Reversible C-Gate and N-Input Reversible Q-Gate are done in C++ programming language on a Windows 10 Operating System installed PC with Advanced Micro Devices A10-9600P Radeon R5, 10 Compute Cores 4C+ 6G 2.4 GHz speed and 8 GB of RAM. Five runs are conducted for each of these algorithms, with the average obtained or computed in between. This procedure is repeated for n = 6 to 23.

Table 10 is based from Ali et al's work on generation of gray code sequence using time efficient approaches [13]. Figures

20-23 show the relevant graphs, which summarize the measured execution times of these algorithms for n = 6 to 23. Table 10 shows the execution time of different algorithms for n = 6 to n = 23 in milliseconds.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

Table 10: Execution time of different algorithms for n = 6 to n = 23 in milliseconds.

No. of bits (n)	Existing Approaches					Innovative Approaches			
	Recursive	Dynamic Programming	Optimal	Backtracking	MOptimal	NDFA	Pascal Triangle	N-Input Reversible C-Gate	N-Input Reversible Q-Gate
6	0	0	0	0	0	NA	6580	-	-
7	15	0	0	0	0	-	7676	-	-
8	31	15	15	15	15	-	8772	-	-
9	62	47	47	31	47	-	9868	-	-
10	109	94	94	78	94	-	10964	-	-
11	203	172	156	156	156	-	-	-	-
12	469	406	297	407	297	-	-	-	-
13	1016	875	594	984	594	-	-	-	-
14	2765	2453	1156	4485	1156	-	-	-	-
15	5188	4578	2282	20750	2282	-	-	-	-
16	10938	9625	4532	127656	4500	-	-	-	-
17	26219	22719	9563	466547	9453	-	-	-	-
18	56751	49984	19625	936412	19391	-	-	-	-
19	147761	109375	39859	----	39593	-	-	-	-
20	453635	305951	76844	----	76421	-	-	-	-
21	1287722	727854	170781	----	168172	-	-	-	-
22	----	----	358859	----	354844	-	-	-	-
23	----	----	721223	----	706140	-	-	-	-

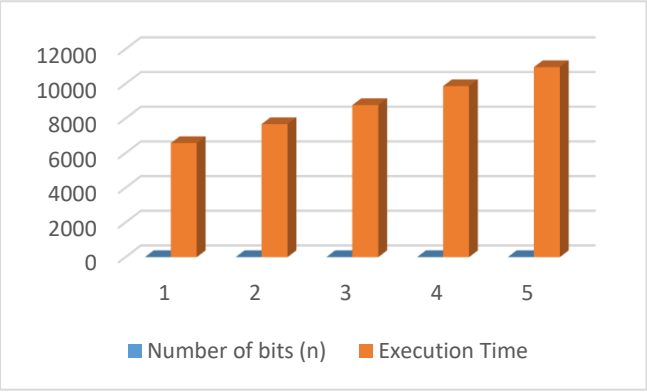


Figure 20: Number of bits Vs. Execution time of Pascal Triangle approach to gray code generation

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4708 - 4734
https://publishoa.com
ISSN: 1309-3452

Discussion

From Figures 3 - 6, it is observed that nondeterministic finite sate automata can be used to generate gray code sequences for any number n. Quantum systems are inherently non-deterministic. Hence concepts that can guarantee or depict non-determinism and can generate gray codes at various levels of n are utilized. Another concept employed is the Pascal Triangle (which can be derived from the binomial theorem). Its ability to generate various sequences at various levels or depths informed the authors, that it would be capable of generating qubit vector sequences at various levels, hence its utilization. It is a simple mathematical concept, yet involves a huge amount of processing as the depth of the Pascal triangle increases. From Figure 9 and the Gray code Implementation using Pascal triangle in section 3, it is seen that various qubit vectors are generated at various depths, and these vectors form various sequences forming a gray code. From Table 10 and Figure 20, it can be seen that there is an exponential increase in execution time as the number of rows or the depth of the Pascal triangle increases, nevertheless it still generates the grey codes at various levels. From Tables 11-14, we have classical gray code of n-bits generated using an N-input Reversible C-Gate– a modified XOR Gate. From Tables 16-19, we have quantum gray code of n-bits generated using an N-input Reversible Q-Gate- a modified C-NOT Gate. These approaches are effective and shown through the generation of gray code tables for n=1 to n=4 in Tables 2–9. Circuits corresponding to the tables are also shown in Figures 11 – 14 and Figures 16-19.

Conclusion and Future Work

This paper seeks to innovatively generate gray codes using concepts like Pascal triangle, N-Input Reversible C-Gate and N-Input Q- Gate. Future works will be to generate gray code sequences using other mathematical concepts. In addition, the authors will design and implement a quantum concatenation operator for concatenating qubits even though there is way to join or merge two quantum circuits to form a single circuit.

Conflicts of interest/Competing interests

The corresponding author declares that there is no conflict of interest on behalf of all authors.

Ethics approval

We, the authors, strictly followed the code of ethics for producing papers, guaranteeing that no plagiarism occurred and that the material was accurately referred to its original authors. There are no ethical difficulties or challenges with this work, and it follows the standard structure for producing manuscripts or articles. This work adheres to strict ethical guidelines.

Consent to participate

All authors consent to participating in this research or paper.

REFERENCES

[1] Asplund, J., & Keranen, M. (2020). TS (v, λ) with Cyclic 2-Intersecting Gray Codes: v≡ 0 or 4(mod12). Graphs and Combinatorics, 36(3), 415–436. https://doi.org/10.1007/s00373-019-02107-1

[2] Kumari, A., Pal, A., Singh, A., & Sharma, S. (2020). All-optical binary to gray code converter using non-linear material based MIM waveguide. Optik, 200, 163449. https://doi.org/10.1016/j.ijleo.2019.163449

[3] Gutierres, G., Mamede, R., & Santos, J. L. (2018). Gray codes for signed involutions. Discrete Mathematics, 341(9), 2590–2601. https://doi.org/10.1016/j.disc.2018.06.011

[4] Di Matteo, O., McCoy, A., Gysbers, P., Miyagi, T., Woloshyn, R. M., & Navrátil, P. (2021). Improving Hamiltonian encodings with the Gray code. Physical Review A, 103(4), 1–22. https://doi.org/10.1103/PhysRevA.103.042405
[5] Savage, C. (1997). A survey of combinatorial gray codes. SIAM Review, 39(4), 605–629. https://doi.org/10.1137/S0036144595295272
[6] Paul, S., & Chang, J. (2016). Design of absolute encoder disk coding based on affine n digit N-ary gray code. Conference Record - IEEE Instrumentation and Measurement Technology Conference, 2016-July. https://doi.org/10.1109/I2MTC.2016.7520384
[7] Jacques, M., Wong, D., & Woo, K. (2020). Generating Gray codes for weak orders in constant amortized time. Discrete Mathematics, 343(10), 1–10. https://doi.org/10.1016/j.disc.2020.111992
[8] Sawada, J., Williams, A., & Wong, D. (2017). Necklaces and Lyndon words in colexicographic and binary reflected Gray code order. Journal of Discrete Algorithms, 46–47(1), 25–35. https://doi.org/10.1016/j.jda.2017.10.002
[9] Rheinboldt, W. (1992). Combinatorial algorithms. In Choice Reviews Online (Vol. 29, Issue 05). https://doi.org/10.5860/choice.29-2751
[10] Yehezkeally, Y., & Schwartz, M. (2017). Limited-magnitude error-correcting gray codes for rank modulation. IEEE Transactions on Information Theory, 63(9), 5774–5792. https://doi.org/10.1109/TIT.2017.2719710
[11] Faloutsos, C. (1988). Gray Codes for Partial Match and Range Queries. IEEE Transactions on Software Engineering, 14(10), 1381–1393. https://doi.org/10.1109/32.6184
[12] Schwartz, M. (2014). Gray codes and enumerative coding for vector spaces. IEEE Transactions on Information Theory, 60(1), 271–281. https://doi.org/10.1109/TIT.2013.2286616
[13] Ali, M. M., Islam, M. N., & Foysal, A. B. M. (2009). Algorithms for generating binary reflected gray code sequence: Time efficient approaches. Proceedings - 2009 International Conference on Future Computer and Communication, ICFCC 2009, 79, 79–83. https://doi.org/10.1109/ICFCC.2009.41
[14] Gray, F. (1953). Pulse Code Communication (PDF). New York, USA: Bell Telephone Laboratories, Incorporated. U.S. Patent 2,632,058. Serial No. 785697. Archived (PDF) from the original on 2020-08-05. Retrieved 2020-08-05.
[15] Ramakrishnan, R. (2018). Algorithms: Gray Binary Code - A different way of ordering numbers. Retrieved from https://dev.to/rrampage/algorithms-gray-binary-code---a-different-way-of-ordering-numbers-14e3
[16] John, G. (2019). What is Gray code? Retrieved from https://www.tutorialspoint.com/what-is-gray-code Retrieved on 03/03/2021
[17] Wikipedia (n.d). Rotary encoder. Retrieved from https://en.wikipedia.org/wiki/Rotary_encoder
[18] Cormier-Iijima, S. (2010). Alternative Combinatorial Gray Codes. https://sciyoshi.com/2010/12/gray-codes/ Retrieved on 03/03/2021
[19] Vajnovszki, V. (2002). Generating a Gray Code for P-Sequences. Journal of Mathematical Modelling and Algorithms 1, 31–41. https://doi.org/10.1023/A:1015622720041
[20] Bose, P., Carmi, P., Jansens, D., Maheshwari, A., Morin, P., Smid, M. (2010). Improved Methods For Generating Quasi-gray Codes. In: Kaplan, H. (eds) Algorithm Theory - SWAT 2010. SWAT 2010. Lecture Notes in Computer Science, vol 6139. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-13731-0_22
[21] Gray code. (n.d.). Retrieved from https://cp-algorithms.com/algebra/gray-code.html
[22] Eberhart, R.C., Shi, Y. (2011). Computational Intelligence: Concepts to Implementations. Elsevier. ISBN 0080553834, 9780080553832
[23] Ludman, J. E., & Sampson, J. L. (1981). A technique for generating Gray codes. Journal of Statistical Planning and Inference, 5(2), 171–180. doi:10.1016/0378-3758(81)90027-6
[24] Er, M. C. (1985). Two Recursive Algorithms for Generating the Binary Reflected Gray Code. Journal of Information and Optimization Sciences, 6(3), 213–216. doi:10.1080/02522667.1985.10698822
[25] Code Review. (2018). A tiny Java library for generating Gray codes. Retrieved from https://codereview.stackexchange.com/questions/184299/a-tiny-java-library-for-generating-gray-codes Retrieved on 03/03/2021
[26] Robinson, J. P., and Cohn, M. (1981), Counting sequences, IEEE Trans. Comput. C-30, 17-23.

[27] Phillips, A.T., and Wick, M.R. (2005). A Dynamic Programming Approach to Generating a Binary Reflected Gray Code Sequence.
[28] Er, M. C. (1984). On Generating the N-ary Reflected Gray Codes. IEEE Transactions on Computers, C-33(8), 739–741. doi:10.1109/tc.1984.5009360
[29] Vickers, & Silverman. (1980). A Technique for Generating Specialized Gray Codes. IEEE Transactions on Computers, C-29(4), 329–331. doi:10.1109/tc.1980.1675573
[30] Pothireddy A., Basappa, J.A., Wheeler, D.G. (2008). Generating a gray code for an odd length sequence using a virtual space. https://patents.google.com/patent/US20090295608A1/en
[31] Mütze, T., & Nummenpalo, J. (2018). Efficient computation of middle levels gray codes. ACM Transactions on Algorithms, 14(2), 915–927. https://doi.org/10.1007/978-3-662-48350-3_76
[32] Di Matteo, O., McCoy, A., Gysbers, P., Miyagi, T., Woloshyn, R. M., and Navrátil, P. (2021). Improving Hamiltonian encodings with the Gray code. Phys. Rev. A 103, 042405. https://doi.org/10.1103/PhysRevA.103.042405
[33] Bhamra, K. S., Joshi, G., & Kumar, N. (2021). An Efficient Design of Binary to Gray Code Binary Converter using QCA. IOP Conference Series: Materials Science and Engineering, 1033(1). https://doi.org/10.1088/1757-899X/1033/1/012014
[34] Chang, Y. H., Wu, R. Y., Lin, C. K., & Chang, J. M. (2021). A loopless algorithm for generating (k, m)-ary trees in Gray code order. Optimization Letters, 15(4), 1133–1154. https://doi.org/10.1007/s11590-020-01613-z
[35] He, X., Zheng, D., Kemao, Q., & Christopoulos, G. (2019). Quaternary gray-code phase unwrapping for binary fringe projection profilometry. Optics and Lasers in Engineering, 121(February), 358–368. https://doi.org/10.1016/j.optlaseng.2019.04.009
[36] Jacques, M., Wong, D., & Woo, K. (2020). Generating Gray codes for weak orders in constant amortized time. Discrete Mathematics, 343(10), 111992. doi:10.1016/j.disc.2020.111992
[37] Konstantinova, E., & Medvedev, A. (2016). Independent Even Cycles in the Pancake Graph and Greedy Prefix Reversal Gray Codes. Graphs and Combinatorics, 32(5), 1965–1978. https://doi.org/10.1007/s00373-016-1679-x
[38] Paul, S., & Chang, J. (2016). Design of absolute encoder disk coding based on affine n digit N-ary gray code, IEEE International Instrumentation and Measurement Technology Conference Proceedings, 1-6.
[39] Dilip, P. S., Somanathan, G. R., & Bhakthavatchalu, R. (2020). Gray code for test pattern generation. AIP Conference Proceedings, 2222(April). https://doi.org/10.1063/5.0004319
[40] Salgado-fuentes, V., Moreno, G., Salgado-fuentes, V., & Moreno, G. (2021). Switching Systems Synthesis Method Using Permuted Gray Code Tables (PGC Method). Latin-American Journal of Computing (LAJC), VIII(1).
[41] Kimoto, T. (2013). Spatial Image Watermarking by Error-Correction Coding in Gray Codes. Journal of Signal and Information Processing, 04(03), 259–273. https://doi.org/10.4236/jsip.2013.43034
[42] Sidhi, T. A. P. (2016). Enhancing the Network Security with Gray Code. World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering, 10(3), 492–496.
[43] The National Academies Press (2019), Quantum computing: progress and prospects, https://doi.org/10.17226/25196
[44] Arazi, B. (1984). An Approach for Generating Different Types of Gray Codes. Information and Control 63, 1-10
[45] Dana J. (5 Oct, 2010). Improved Methods for Generating Quasi-Gray Codes
[46] Wikipedia. (2022). Controlled NOT gate. https://en.wikipedia.org/wiki/Controlled_NOT_gate