Design of Quantum Circuits to Play Chess in a Quantum Computer

Anshuman Padhi [@],[1,*] Debankit Priyadarshi [@],[1,†] Harsh Bardhan Gupta,[2,‡] Khush Agrawal,[2,§] Bikash K. Behera,[3,4,¶] and Prasanta K. Panigrahi[4,**]

[1] *School of Physical Sciences, National Institute of Science Education and Research, HBNI, Jatni 752050, Odisha, India*
[2] *Indian Institute of Science Education and Research Kolkata, Mohanpur 741246, West Bengal, India*
[3] *Bikash's Quantum (OPC) Pvt. Ltd., Balindi, Mohanpur 741246, West Bengal, India*
[4] *Department of Physical Sciences, Indian Institute of Science Education and Research Kolkata, Mohanpur 741246, West Bengal, India*

Chess is an extremely ancient board game, which can be played using physical chess boards and can be enjoyed virtually by using classical computers. Gradually, as the quantum computers are being developed , their potential applications in various complex challenges are being realized. Upon the arrival of quantum computers at the scene, games can now be played on a quantum computer by designing quantum circuits on it. Though classical computers are enough to play the game virtually, quantum computers will always be better for tackling more and more complexity in a game. Since classical computation is used chess engines, in future, the engines can be made faster by using quantum computation, primarily because of their ability to use fewer steps to obtain at the same result as that of classical computers. And we can achieve the same with quantum computers by designing quantum circuits. However, an 8×8 chessboard is quite large and the existence of various pieces makes the circuit quite complex for us to build, so here we present quantum circuits for a 3×3 chessboard with only pawns in it. From here, it can be extrapolated to create, circuits for an 8×8 board and various other pieces. These quantum circuits can be designed on a quantum computer to play chess by any two users. We explain the working of the quantum circuits in detail with the algorithm behind the movement of pawns on the chessboard, in future, we can use this to later develop a quantum chess engine which will boost the quality of chess.

Keywords: Quantum Games, Quantum Chess, Quantum Circuit, IBM Quantum Experience

I. INTRODUCTION

Research in quantum game theory has been a huge matter of interest off late thanks to its various implications in major disciplines of sciences and social sciences [1]. Research fraternity of quantum computation and information has specifically showcased its immense potential to create and develop games that can be played on currently available quantum computers. Recently quantum tic-tac-toe game has been introduced by Leaw and Cheong [2]. Similarly, a quantum version of the game of Go, introduced by Ranchin [3]. Pal *et al.*, used a combination of classical and quantum computation to develop the algorithm for solving Sudoku puzzles [6]. Quantum Monty Hall game on a quantum computer has been developed by Paul *et al* [5]. A simple shooting game has also been designed by Roy *et al.* [4]. A group of researchers also proposed a generalization of the quantum prisoner's dilemma in the form of a quantum game in 2002 [10]. Diner's dilemma was successfully explained by Anand *et al.* in 2019 [7]. Similarly, an investigation on non-zero sum games was performed by Eisert *et al.* [11]. Splendid work was also done by Singh *et al.* [8] on demonstration of how Bingo games can be played in a quantum computer. The above-mentioned pioneering works have been a constant source of motivation for us to develop other games on quantum computers. In this work, we propose a very fundamental understanding of how we can play chess using quantum computers.

The origin of the game of chess has many conflicting theories. However, the most believed and accepted theory suggests that Chess originated from Indian soil around 6th Century A.D. and was carried to the Arabic countries and Europe by the Persians [14]. The name and the terminologies associated with the game underwent various adaptations from empires to empires. Even the cultural beliefs started affecting the rules of the game as it is widely believed that the immense power of queen in the chessboard roots back to the entrance of Chess in Europe, which has a traditional Monarchy with the Queen being at the helm of affairs [15]. Gradually, it started getting modernized in Europe and the first competitive chess tournament, the London Chess Tournament was hosted in 1851 and the first world championship was organized in 1886 [13]. And since then it has been a long and beautiful journey with a lot of competition between different players all over the world. The likes of Steinitz, Capablanca, Alekhine, Spassky, Fischer, Karpov, Kasparov revolutionized the way chess was played [14].

Chess is an extremely calculated mind game, with numerous possible moves, thus began the advent of mathe-

* anshuman.padhi@niser.ac.in; @ Those are the first authors and have equally contributed to this work
† debankit.priyadarshi@niser.ac.in; @ Those are the first authors and have equally contributed to this work
‡ harshbardhangupta818@gmail.com
§ ka19ms027@iiserkol.ac.in
¶ bkb18rs025@iiserkol.ac.in
** pprasanta@iiserkol.ac.in

maticians and computer scientists in realizing whether a machine could play chess just like a human being. Alan Turing, a British computer scientist, mathematician, and cryptanalyst, gave the first algorithm to play chess [12, 16, 19]. Later, his colleague Dietrich Prinz, produced the first programmed chess, which could find a series moves which leads to a win if the checkmate is achievable in 2-3 moves [16]. Since then, development of chess engines were highly pursued and in 1996, IBM's chess engine "Deep Blue" was successfully able to beat the reigning world champion and arguably one of the greatest of all time, Garry Kasparov [17]. After that remarkable feat, engines have never looked back and they have grown from strength to strength. Now they are almost unbeatable, and help in preparation of professional chess players in sharpening their game.

Chess is a classic board game where two players, assigned pieces of a specific color each (white or black), play out on an 8×8 board. Each player begins with sixteen pieces: one king, one queen, two rooks, two knights, two bishops, and eight pawns. The movement is different for different kinds of pieces and the hierarchy of power is set by the default rules, the notion being, the most powerful after the King is the Queen and then it follows down to the pawn. The objective is to beat the opponent by checkmating their opponent, where the vanquished side's king is under threat and there is no way to escape. During the game, different pieces are used to attack and capture the pieces of the other player while supporting each other [14].

Applying the principles of quantum computation, we can design circuits to play chess on a quantum computer. To avoid complexities while designing the circuits, here we just propose a method to move pawns, as they are the most fundamental pieces on a chessboard. From getting motivation from the algorithm behind the motion of pawns, we can set up circuits to move other pieces of the board which have relatively complex motion than that of pawns.

Here we first name the squares of the board in terms of qubits, assign some more qubits to imply if a pawn is present or not and if present, what color it is. Then we ask the user for two inputs, first being the square from which he/she wants to move a piece and the second being the square to which the piece is expected to move.

Here we majorly focus on the validity of the two inputs i.e., the given input should be consistent with the allowed set of moves. We check whether the input follows the allowed set of moves by putting conditions in the designed circuits.

The rest of the paper is organized as follows. Section II introduces the rules to be followed while moving a pawn. In section III, the logic behind the motion of pawns following the set of rules and the quantum circuits for the same algorithms has been discussed. Finally, we conclude in Section IV by discussing future implications and applications of the work.

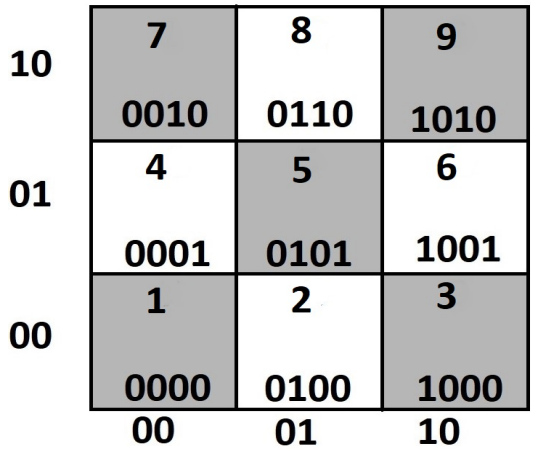

FIG. 1: Squares expressed in the form of coordinate qubits

Status assigned to a square	Implication about the the piece present
00	Vacant
01	Black Pawn
10	white Pawn

TABLE I: Convention used for denoting status of a square

II. RULES AND CONVENTIONS USED

Before developing the game we must know how pawns move in a chessboard. As per rules [18]-

- Pawns can move forward one square, if that square is unoccupied. By no means it can move backward.
- They can capture an enemy piece on either of the two spaces adjacent to the space in front of them (i.e., the two squares diagonally in front of them) but cannot move to these spaces if they are vacant.

Before moving a pawn on a chessboard, we must initialize the board at the beginning. For naming each square in a 3×3 chessboard, we use 4 qubits in total, 2 for denoting the column that square belongs to (to be called X-coordinate of the square henceforth) and 2 for denoting the row that square belongs to (to be called Y-coordinate of the square henceforth). We assign $|00\rangle$ for the first row starting from the bottom, $|01\rangle$ for the second row and $|10\rangle$ for the third row. Similarly, we assign $|00\rangle$ for the first column starting from left, $|01\rangle$ for the second column and $|10\rangle$ for the third column. This convention gives us the squares in terms of 4 qubits by just expressing the position of the squares in terms of columns (X-coordinate) and rows (Y-coordinate). From here on, we call these 4 qubits which are used to denote a square's position on the board, as 'coordinate qubits' (Fig. 1).

Before a move is played, a sample 3×3 chessboard with 3 pawns in white side and 3 pawns in black side

Square Number	Coordinate Qubits	Status of the particular square
1	0000	$S_1=\|10\rangle$
2	0100	$S_2=\|10\rangle$
3	1000	$S_3=\|10\rangle$
4	0001	$S_4=\|00\rangle$
5	0101	$S_5=\|00\rangle$
6	1001	$S_6=\|00\rangle$
7	0010	$S_7=\|01\rangle$
8	0110	$S_8=\|01\rangle$
9	1010	$S_9=\|01\rangle$

TABLE II: The initial state of the board, where each square, expressed in terms of 4 qubits is assigned some initialized value of status before the game starts.

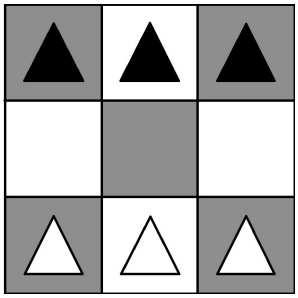

FIG. 2: Initial state of the board

Input for Current square	Input for Target square	Output D1	Output D2	Implication
0100	1010	0	0	Not Valid
0100	0001	0	1	Left Diagonal
0100	1001	1	0	Right Diagonal
0100	0101	1	1	Forward

TABLE III: A white pawn on 5th square of the board with coordinate qubits $|0100\rangle$ is given various target squares, and the truth table of the circuit denotes the validity of move, and also in which direction the pawn shall move

have been considered (Fig. 2). The board as a whole has a state comprising of the respective statuses of all squares (in total 9). The array of statuses of each square $|s1\rangle$,$|s2\rangle$.....$|s9\rangle$ is initialized (Table. II), to set the board before the game starts. As moves are played thereafter, the statuses of each square change.

After naming the squares, now we must have some convention to denote whether in a square a pawn is present or not and if present what color it is. (Henceforth mentioned in the article as 'status' of a given square). We assign two qubits to denote the status of a given square. The square has a status of $|00\rangle$ if no piece is present in it, $|01\rangle$ if a black pawn is present and status of $|10\rangle$ for a white pawn being there (Table. I). The status of all 9 squares constitutes the formation that is present on the board.

III. DESIGNING THE QUANTUM CIRCUITS

A. Checking the validity of inputs-

When we wish to move a piece, first we take note of the square at which the piece is present (to be called 'current' square henceforth). Then we choose the square to which we want the piece to be moved (to be called 'target' square henceforth). To make a move, we ask the user to give us two inputs i.e., the current square and the target square respectively, in terms of 4 qubits each. Next, we go on to check whether those given inputs are valid (i.e., if they adhere to the rules to be followed while moving a pawn). Then we act on the input further to allow necessary changes in the board setup after the move is played, if and only if the inputs are valid.

For successful completion of the move, the choice of the user must be according to rules allowed for the game. The basic rule is that the pawn should move forward. In no circumstance, it is allowed to move backwards.

For this condition (Fig. 3), we have used the fact that the Y coordinate of the source square and the target square should differ by a value of one. Therefore, we have implemented an adder circuit to increase the Y coordinate of the current square by one and used the comparator circuit to compare it with the Y coordinate of the target square. If the above condition is satisfied (meaning that the move is partially valid as the piece goes to next row), the X coordinates of both the squares are compared where the output from Y-coordinate comparison act as the controls. Otherwise, the move is simply not executed.

Let the X-coordinate of the source square be some x. Then according to the rules, the X coordinate of the target box is allowed the values of $x-1$, x and $x+1$ since the pawn is allowed to move either straight or to left diagonal or right diagonal. Therefore, we have used this condition to design the circuit. We have made use of a subtractor circuit, a comparator circuit and an adder circuit to achieve the necessary circuit [9].

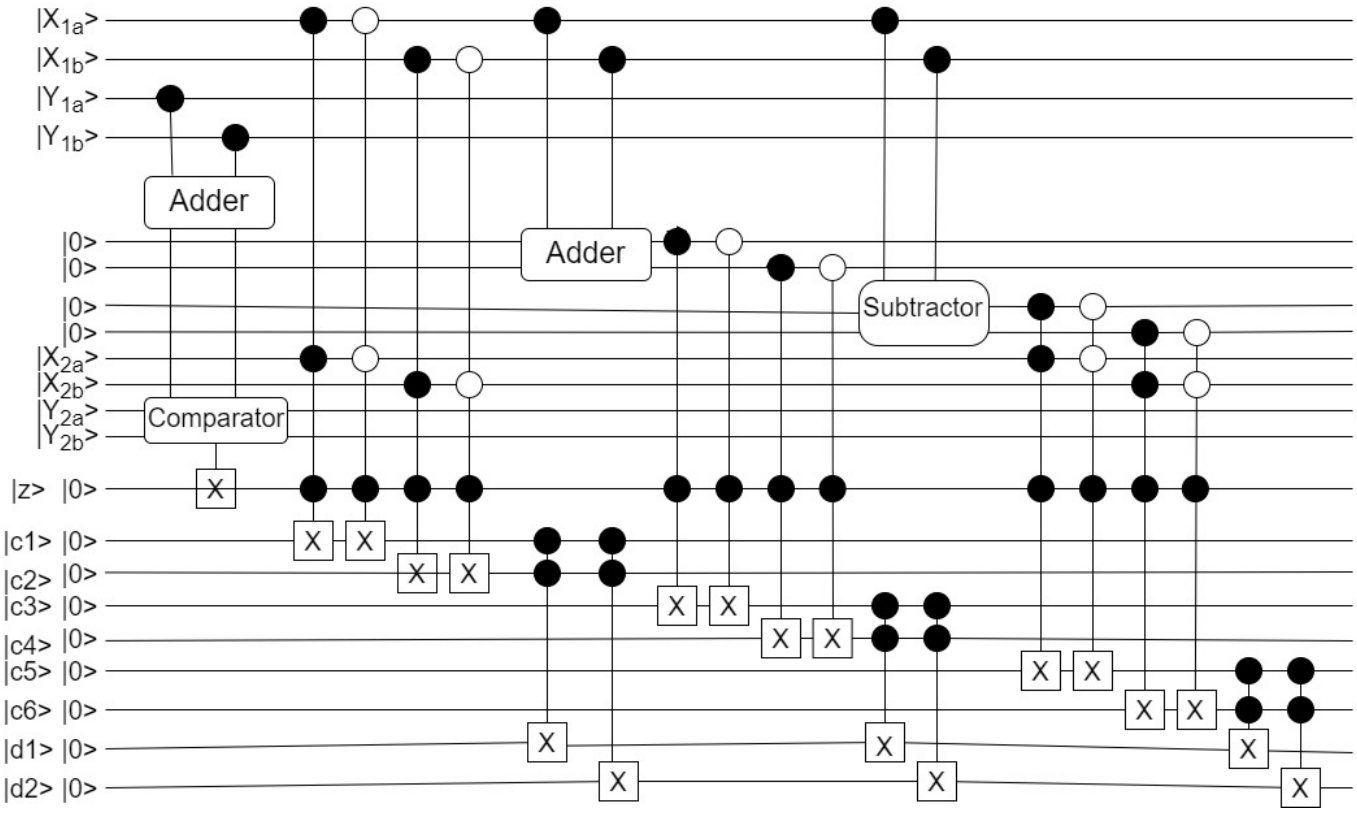

FIG. 3: Quantum circuit designed to find out whether a move is valid or not. The two input squares $|X_{1a}X_{1b}Y_{1a}Y_{1b}\rangle$ $|X_{2a}X_{2b}Y_{2a}Y_{2b}\rangle$ are fed into the circuit as shown. The Y coordinate of current square is treated with an adder and is compared with the Y coordinate of target square. Then it is checked whether the pawn has moved ahead or not. If this doesn't match, then the output is null. If they match, the circuit goes on to find the relation between the X coordinates of both the sqaures. Here, treating the X-coordinate of current square with an adder+comparator, a comparator and a subtractor+comparator circuit respectively, it finds what relation the X-coordinate of target square has with the x-coordinate of current square. Accordingly, the output $|d1\rangle\,|d2\rangle$ is determined.

In our circuit, we have compared values of the Y coordinate of inputs with the help of an adder circuit and a comparator circuit, and the z qubit stores the result. Then we compare the X coordinate for straight, left diagonal and right diagonal case and the qubits named as $|c1\rangle$, $|c2\rangle$, $|c3\rangle$, $|c4\rangle$, $|c5\rangle$ and $|c6\rangle$ stores the result respectively. Then we have taken a common output from these qubits and the qubits $|d1\rangle$ and $|d2\rangle$ (known as "direction qubits") stores this output. The direction qubits indicate whether the pawn moves straight or right corner or left corner. These direction qubits help in the next step, where we find the present status corresponding to the inputs. An example has been shown in Table III.

B. Finding the status of the input squares

When a piece moves from one square to the other, the status of both the squares gets changed. When the piece is moved, the 'current' square goes vacant and the target square now has the piece which was moved. So to move a piece, we need to operate on the statuses of 'current' and 'target' square. For that, we need to know the present statuses of both the squares and later perform operations on them.

Hence to know the present status of the input, first, we need to know which square the input implies to. We feed the inputs into a circuit (Fig. 5), which compares the given input qubits to all coordinate qubits of the 9 squares, and finds which square the input matches with. After finding the matching square, it copies the status of that matched squares to blank qubits $|z1\rangle$(a and b) and $|z2\rangle$ (a and b) where $|z1\rangle$ is the status of the 'current' square and $|z2\rangle$ is the status of the 'target' square. The circuit achieves all this just by feeding of the inputs, as we use the coordinate qubits of all 9 squares as the controls. When the input matches with one of the 9 coordinate qubits, that portion of the circuit get activated and only the corresponding status of that qubit gets copied. These 2 statuses go to the next circuit for further operation, which is controlled by the 'direction' qubits obtained in the previous step. Direction qubits determine which direction the piece will go i.e., whether forward or right diagonal or left diagonal.

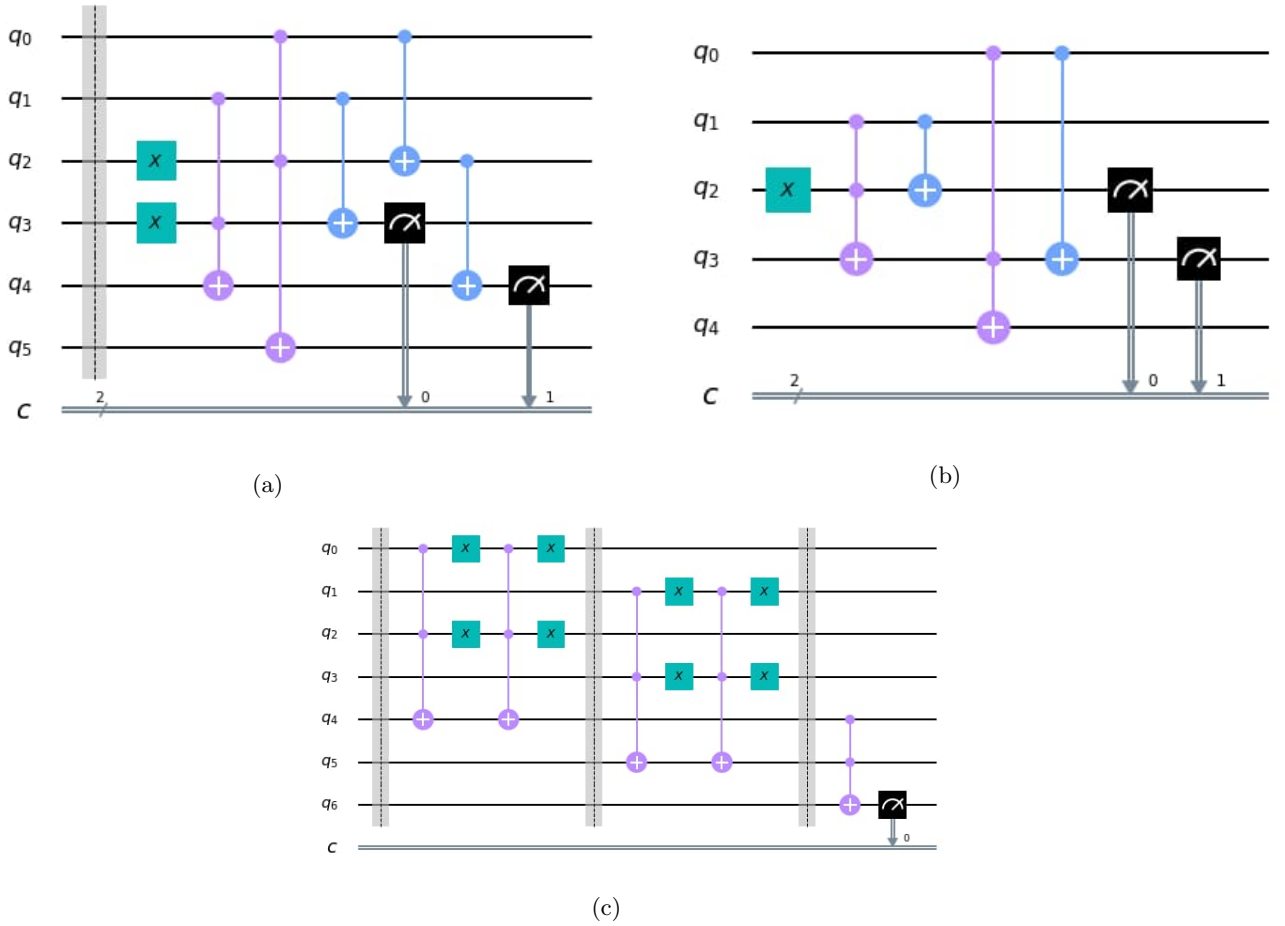

FIG. 4: **The circuits of subtractor, adder and comparator using IBM Q Experience.** Figure (a) corresponds to a subtractor. Here, two qubits are given as inputs and the other qubits are used to subtract two qubits from them by the method of 2's complement. Figure (b) corresponds to an adder circuit. Here, two qubits are given as inputs and two adder qubits are added to them by taking another as the carrier. Figure (c) corresponds to a comparator circuit. Here, four qubits are given as inputs where each two qubits corresponds to a set of input. The first qubit compares with the third qubit while the second qubit compares with the fourth qubit. If the qubits match, then the output i.e., $|q6\rangle$, is $|1\rangle$ or it is $|0\rangle$.

C. Operation on the statuses

This circuit (Fig. 6) is some sort of a gate, where the values of $|d1\rangle$ and $|d2\rangle$ determine which direction the piece will go i.e., what operation to be applied on $|z1\rangle$ and $|z2\rangle$. If direction qubits imply a straight motion of pawn, it implies the status of 'current' square becomes $|00\rangle$ as the piece moves out from that square and there is no piece on it anymore and the status $|z2\rangle$ becomes the same as that of the previous $|z1\rangle$ because the same piece has moved into the 'target' square right now.

As discussed earlier, $|d1\rangle$ and $|d2\rangle$ control what operation the both statuses undergo. However, is it the only control we require in the circuit? No. If $|d1\rangle$ and $|d2\rangle$ imply the piece to move forward, the status of the target square($|z2\rangle$) must be $|00\rangle$ i.e., it should be vacant. And if $|d1\rangle$ and $|d2\rangle$ imply a diagonal direction, then the status of the target square ($|z2\rangle$) has to be $|01\rangle$ and the status of the current square is $|10\rangle$ and vice versa. It means, for the pawn to move diagonally, the diagonal piece must be of an opposite color to that of the piece or else the move would be rendered invalid. Hence in this circuit along with $|d1\rangle$ and $|d2\rangle$, $|z1\rangle$ and $|z2\rangle$ also play the role of controls! So we, therefore, copy $|z1\rangle$ and $|z2\rangle$, and use one set of copies as the controls and the other set of copies (denoted as $|z1'\rangle$ and $|z2'\rangle$ as the qubits on which the operations for either straight or diagonal movement is performed. In the circuit given, we can see that $|z1\rangle$, $|z2\rangle$, $|d1\rangle$ and $|d2\rangle$ act as the controls, and they decide which operation is allowed on $|z1'\rangle$ and $|z2'\rangle$. If a pawn moves forward, then the status of the current square goes vacant and the status of the target square becomes the same as the status of the current square previously. Hence, $|z2\rangle$ turns into $|z1\rangle$ and we achieve that by first rendering $|z2\rangle$

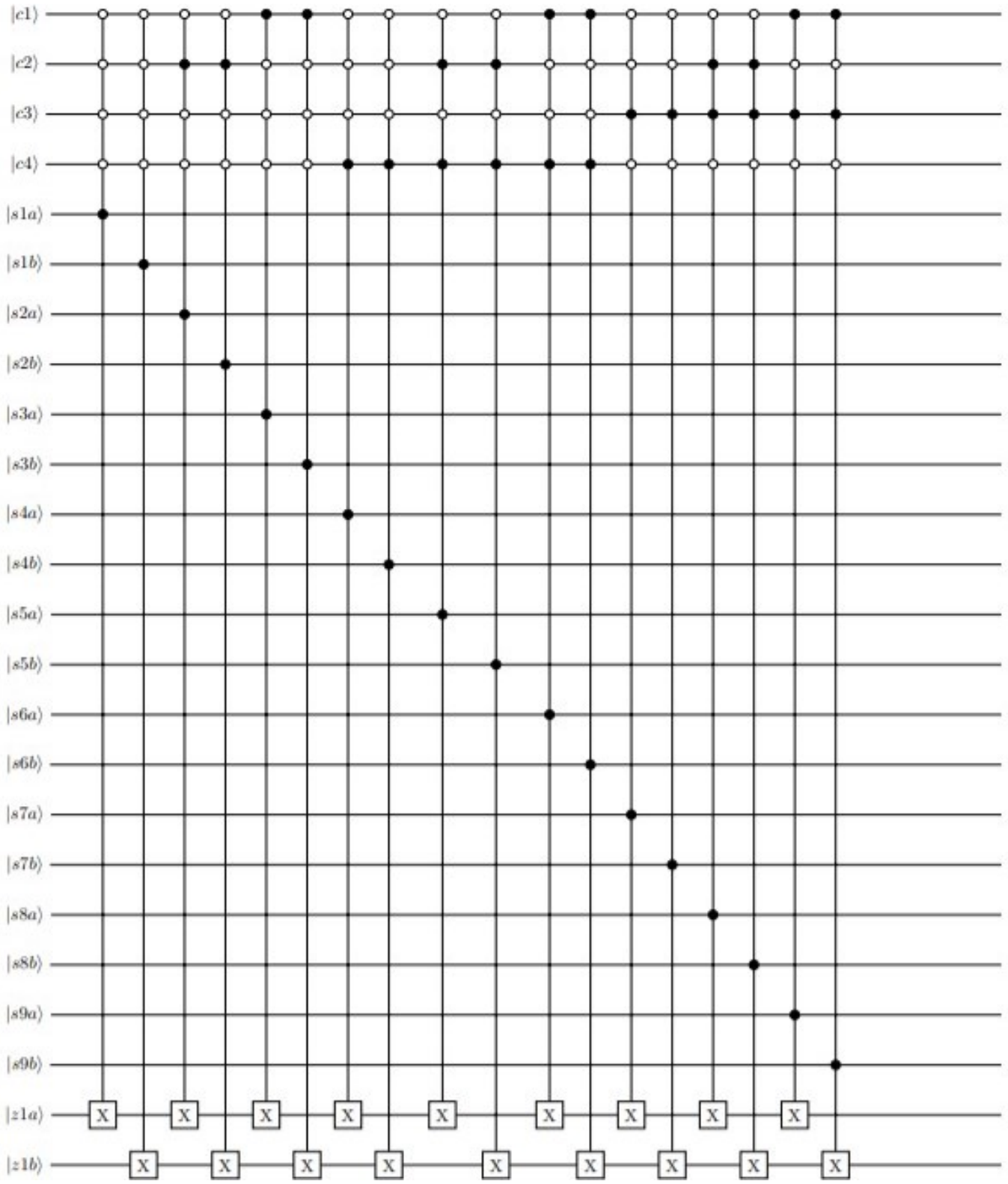

FIG. 5: Quantum circuit designed to find out the status of the two input sqaures. Here, we feed the input qubits into the circuit in the form of $|c1\rangle$, $|c2\rangle$, $|c3\rangle$ and $|c4\rangle$. It goes through a series of gates which corresponds to different position qubits that are used in the board. When the input qubit matches with one from the set of position(coordinate) qubits of the squres, the corresponding status of the square i.e., $|s1\rangle...|s9\rangle$, is copied to a pair of new qubits, $|z1a\rangle$ and $|z1b\rangle$, that is used in the next step. The two set of input qubits (source and target) is provided in the same manner to the circuit and the corresponding status is extracted in the qubits $|z1\rangle$ and $|z2\rangle$.

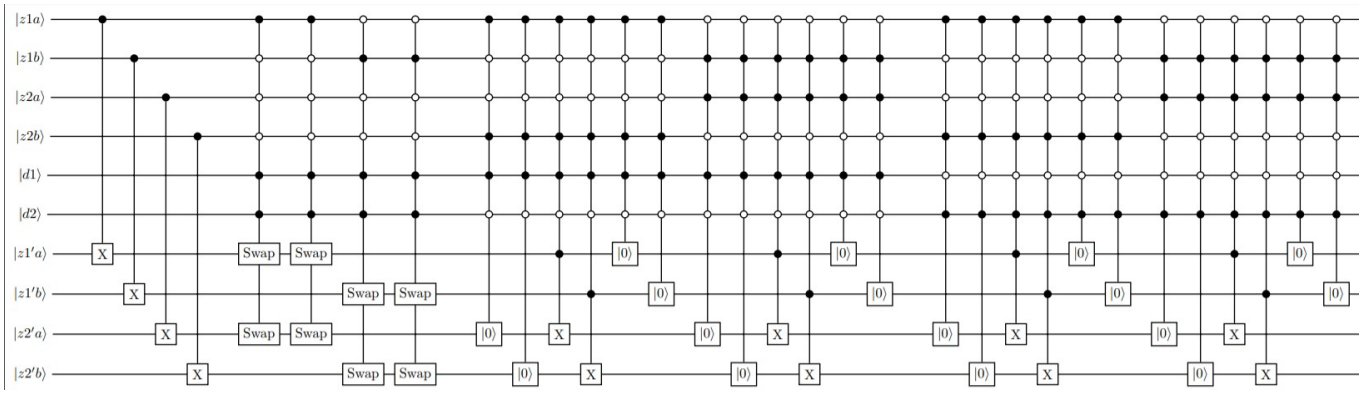

FIG. 6: In the above circuit, at first, the values of $|z1\rangle$ and $|z2\rangle$ is copied to $|z1'\rangle$ and $|z2'\rangle$. Now, $|z1\rangle$, $|z1\rangle$, $|z1\rangle$ and $|z1\rangle$ act as controls and the corresponding operations are done on $|z1'\rangle$ and $|z2'\rangle$. If the values of $|d1\rangle$ and $|d2\rangle$ is $|1\rangle$ and $|1\rangle$, and the value of $|z2\rangle$ is $|00\rangle$, then only the swap operation is carried out between $|z1'\rangle$ and $|z2'\rangle$. Similarly, if $|d1\rangle$ and $|d2\rangle$ corresponds to left diagonal or the right diagonal, then the corresponding operations are done. If the values of $|d1\rangle$ and $|d2\rangle$ are $|0\rangle$ and $|0\rangle$, then no operations are carried out and the status of the squares remain unchanged.

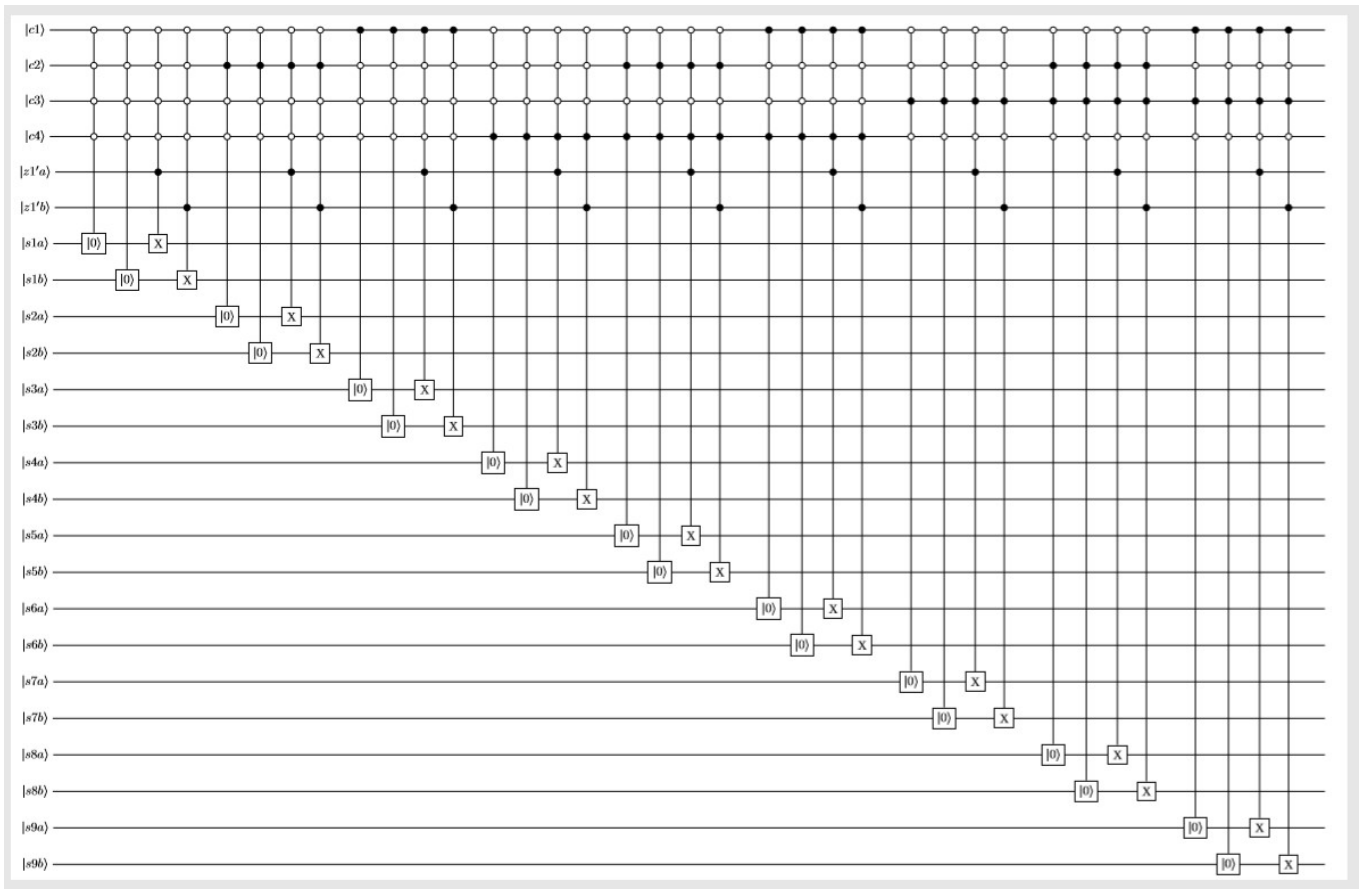

FIG. 7: In the above circuit, $|c1\rangle$, $|c2\rangle$, $|c3\rangle$, $|c4\rangle$, $|z1'a\rangle$ and $|z1'b\rangle$ qubits act as controls. Depending on the values of $|c1\rangle$, $|c2\rangle$, $|c3\rangle$ and $|c4\rangle$, the values of $|z1'a\rangle$ and $|z1'b\rangle$ are copied to the status of the corresponding square on the board.

into $|00\rangle$ and then copying $|z1\rangle$ on it. And later $|z1\rangle$ becomes $|00\rangle$ as now the current square is vacant after the move is played. Thus we obtain the new statuses of both the squares after the move has been played.

D. Updating the change in the initial state of the board

Now we must plug this new value to the status of both the squares whose status we copied and processed. After plugging them back, the respective statuses of all the coordinate qubits give the new state of the board i.e., the state of the board after the movement has been updated (Fig. 7). Thus, we were successfully able to move the pawn in a chessboard.

IV. CONCLUSION

In this article, we discussed moving a pawn in a 3×3 chessboard. Deriving motivation from this, we can design larger circuits to do the same in an 8×8 chessboard. For incorporating other pieces, we may use some more qubits to distinguish pieces from one another in terms of the type of movement they are allowed to. And then we can check the validity of inputs by doing some intricate designing of the validating circuit. Then to perform a movement, we might have to use a bit of programming for pieces with complex motions. Suppose, for a bishop who can move in a diagonal by any number of squares per move, we might break it into a series of pawn's diagonal movements. The only difference being that the bishop can move diagonally even if the diagonal square is absent, unlike pawn. To accommodate that we may use the conditions put in the circuit a bit differently than we did for a pawn. Similarly, for rook, we may break it into a series of pawn's forward movement, where it can capture a piece lying in front of it, unlike pawn. Circuit for the queen's movement can just be the integration of rook's and Bishop's circuit. Thus by realizing the algorithm behind the pawn's movement, we can go on to make circuits for other pieces as well, ultimately making an 8×8 full-fledged chess game as well!

ACKNOWLEDGMENTS

A.P. and D.P. would like to thank Bikash's Quantum (OPC) Pvt. Ltd. for providing hospitality during the course of this project. B.K.B. acknowledges the support of IISER-K Institute fellowship. The authors acknowledge the support of IBM Quantum Experience for producing the basic circuits. The views expressed are those of the authors and do not reflect the official policy of IBM or IBM Quantum Experience team.

[1] R. B. Myerson, Game Theory: An Analysis of Conflict (MIT Press, Cambridge, MA, 1991)
[2] J. N. Leaw and S. A. Cheong, Strategic insights from playing quantum tic-tac-toe, J. Phys. A: Math. Theor.43, 455304 (2010)
[3] A. Ranchin, Quantum Go, arXiv:1603.04751 (2016).
[4] B. B.Roy, B. K. Behera, and P. K. Panigrahi, Modelling A Simple Shooting Game Using Quantum Computation DOI:10.13140/RG.2.2.30976.07680 (2019).
[5] S. Paul, B. K. Behera, and P. K. Panigrahi, Playing Quantum Monty Hall Game in a Quantum Computer arXiv:1901.01136 (2019).
[6] A. Pal, S. Chandra, V. Mongia, B. K. Behera, and P. K. Panigrahi, Solving Sudoku Game Using Quantum Computation, DOI:10.13140/RG.2.2.19777.86885 (2018).
[7] A. Anand, B. K. Behera, and P. K. Panigrahi, Solving Diner's Dilemma Game, Circuit Implementation and Verification on the IBM Quantum Simulator, DOI:10.1340/RG.2.228940.05765 (2019)
[8] V. Singh, B. K. Behera, and P. K. Panigrahi, Design of Quantum Circuits to Play Bingo Game in a Quantum Computer, DOI:10.13140/RG.2.2.19777.86885 (2019).
[9] S. Singh, P. Rathnaparkhi, B. K. Behera, and P. K. Panigrahi, Demonstration of a Quantum Calculator on IBM Quantum Experience Platform and its Application for Conversion of a Decimal Number to its Binary Representation, DOI:10.13140/RG.2.2.12661.63209 (2018).
[10] J. Du, H. Li, X. Xu, M. Shi, J. Wu, X. Zhou, and R. Han, Experimental Realization of Quantum Games on a Quantum Computer, Phys. Rev. Lett. **88**, 137902 (2002).
[11] J. Eisert, M. Wilkens, and M. Lewenstein, Quantum Games and Quantum Strategies, Phys. Rev. Lett. **83**, 3077 (1999)
[12] M. Krol, Have we witnessed a real-life Turing Test?, IEEE Xplore **32**, Issue: 3 (1999).
[13] H. Iida, N. Takeshita, J. Yoshimura, A Metric for Entertainment of Boardgames: Its Implication for Evolution of Chess Variants. In: Nakatsu R., Hoshino J. (eds) Entertainment Computing. IFIP - The International Federation for Information Processing, vol 112. (2003)
[14] History of Chess, Wikipedia, https://en.wikipedia.org/wiki/History_of_chess.
[15] https://www.chesshere.com/resources/chess_history.php
[16] Brief History of computer chess, https://thebestschools.org/magazine/brief-history-of-computer-chess/
[17] History of Chess Engine, https://chessentials.com/history-of-chess-computer-engines/
[18] Rules of Chess, http://www.chesscoachonline.com/chess-articles/chess-rules
[19] Alan Turing Created a Chess Computer, https://www.history.com/news/in-1950-alan-turing-created-a-chess-computer-program-that-