



## RESEARCH ARTICLE

# Quantum 2-Player Games and Realizations with Circuits

Jinliang Zhang<sup>†</sup>, Tian Chen<sup>\*†</sup>, Wenyuan Deng, Xiaoxue Tong, and Xiangdong Zhang<sup>\*</sup>

Key Laboratory of Advanced Optoelectronic Quantum Architecture and Measurements of Ministry of Education, Beijing Key Laboratory of Nanophotonics & Ultrafine Optoelectronic Systems, School of Physics, Beijing Institute of Technology, 100081 Beijing, China.

\*Address correspondence to: [chentian@bit.edu.cn](mailto:chentian@bit.edu.cn) (T.C.); [zhangxd@bit.edu.cn](mailto:zhangxd@bit.edu.cn) (X.Z.)

†These authors contributed equally to this work.

Game theory problems are widely applied in many research areas such as computer science and finance, with the key issue being how to quickly make decisions. Here, we present a novel quantum algorithm for game theory problems based on a continuous quantum walk. Our algorithm exhibits quantum advantage compared to classical game algorithms. Furthermore, we exploit the analogy between the wave function of the Schrödinger equation and the voltage in Kirchhoff's law to effectively translate the design of quantum game trees into classical circuit networks. We have theoretically simulated the quantum game trees and experimentally validated the quantum functionality speedup on classical circuit networks. Due to the robust scalability and stability inherent in classical circuit networks, quantum game trees implemented within this framework hold promise for addressing more intricate application scenarios.

## Introduction

Game theory is a collection of mathematical models that study decision-making in situations involving conflict and cooperation, with the aim of abstracting key elements of various competitive scenarios and scientifically investigating their characteristics [1–5]. Recent research has integrated artificial intelligence into game theory problems, combining machine learning methods to excel in domains such as chess and Go. This convergence has resulted in the development of high-performance computer programs capable of playing at a superhuman level [6–13]. A typical example in the game theory is the 2-player game, which involves the decisions from 2 agents. This problem can be cast into a decision tree to determine which agent wins by calculating the optimal value function within this tree. This is accomplished through high-performance alpha-beta search techniques, which efficiently explore vast search spaces, or by utilizing a general-purpose Monte Carlo Tree Search (MCTS) algorithm. These methods allow researchers to subsequently provide the optimal solution. In such scenarios, the ability to make decisions quickly often proves to be a decisive factor in achieving success. Research has shown [14] that if 2-player game trees are viewed as AND-OR trees, then the minimax value of the game tree corresponds precisely to the value of the optimal solution in the AND-OR tree. Therefore, the evaluation of game problems hinges on efficiently obtaining the optimal solution to the AND-OR tree problems. In the assessment of AND-OR tree problems [15–17], classically, the value of a balanced binary AND-OR tree with zero error in expected time  $O\left(N^{\log_2\left[\left(1+\sqrt{33}\right)/4\right]}\right) = O(N^{0.754})$  can be

computed using a technique called alpha-beta pruning. Despite the fact that no algorithm has outperformed this classical zero-error algorithm for a long time, it remains quite slow.

On the other hand, quantum algorithms offer important advantages in addressing specific 2-player game problems. The simplest case is the Prisoners' Dilemma, which is the one-step decision problem. When quantum strategies are employed in the Prisoners' Dilemma, the decision-making process no longer presents a dilemma [18,19]. Not only the one-step decision problem, the many-step decision problem is also studied with quantum algorithms. Some works encode all possible solutions of a variety of game search problems into a Hilbert space. By using Grover's quantum search algorithm [20], they achieve a quadratic speedup over naive classical algorithms [21–24]. Then, the bounded-error quantum algorithms based on quantum circuit model have been developed for evaluating game trees [25–30] on a graph, which requires an  $O\left(\sqrt{N}\log N\right)$  query for evaluating AND-OR formulas with size  $N$ . However, it remains unclear how to experimentally implement these algorithms or whether tree structures can be used for this purpose.

In this paper, we propose a novel scheme to realize a 2-player game based on AND-OR tree structures. The contributions of our work are 2-fold. First, we employ a subgame design technique to develop a quantum algorithm for the Hamiltonian AND-OR tree using continuous-time quantum walk. Our proposed algorithms achieve a query time of  $O\left(\sqrt{N}\right)$  for evaluating preprocessed approximately balanced AND-OR trees.

Second, we validate the quantum speedup characteristics of this algorithm within circuit networks.

The organization of this paper is as follows. In the “Theoretical Scheme of Quantum 2-Player Zero-Sum Games” section, we introduce basic concepts related to game trees and present the theory of quantum 2-player games based on tree structures. In the “Circuit Designs of Quantum 2-Player Games” section, we demonstrate how to design the quantum algorithm in circuit networks, showcasing its gaming functionality with quantum speedup. The corresponding experimental results of quantum 2-player games in circuit networks have been addressed in the “Experimental Realizations of Quantum 2-Player Games” section. Finally, we discuss and provide future outlooks. The quantum speedup in game tree solving plays a crucial role in various aspects of society, particularly in fields like artificial intelligence and deep learning.

## Results

### Theoretical Scheme of Quantum 2-Player Zero-Sum Games

In this section, we propose a novel multi-step decision-making framework for a 2-player game based on a continuous-time quantum walk, demonstrating quantum speedup. Before providing the details of the quantum 2-player game, we initiate the process by constructing a quantum AND-OR tree using the quantum walk at first.

#### Quantum AND-OR tree based on quantum walk

Here, we demonstrate quantum speedup for the quantum AND-OR tree. First, we design and implement the most common basic 2-input OR gate and 2-input AND gate structures in the quantum game tree. These 2-input OR and AND gates are constructed by using the negative-AND (NAND) gate. The construction details of the NAND gate are shown in Section S1.

The design scheme for the quantum OR tree is shown in Fig. 1A. For a 2-input OR gate, the depth of the quantum OR tree is  $d = 2$ . Here, the parameter  $L$  is taken as  $L = 8\sqrt{N} \approx 12$  with  $N = 2$ . There are a total of  $M = 32$  nodes. It includes 25 nodes of the runway, 3 nodes of the tree structure, and 4 nodes of the input layer. The top row input has 2 states, with connections and disconnections between the first 2 rows of nodes corresponding to inputs 1 and 0, respectively. The lowest root node in the tree structure is connected to a runway at the node  $L + 1 = 13$ , in which the length of runway is  $2L + 1 = 25$ . Each node in Fig. 1A can be represented by a  $32 \times 1$  dimensional quantum state, where the state function  $|r\rangle = (0, \dots, 0, 1, 0, \dots, 0)^T$  represents that the  $r$ th element of the column vector is 1, and all other elements are 0. Thus, the entire quantum OR tree can be represented by a  $32 \times 32$  dimensional Hamiltonian  $H$ . The details of constructing a quantum OR tree are provided in Section S2.

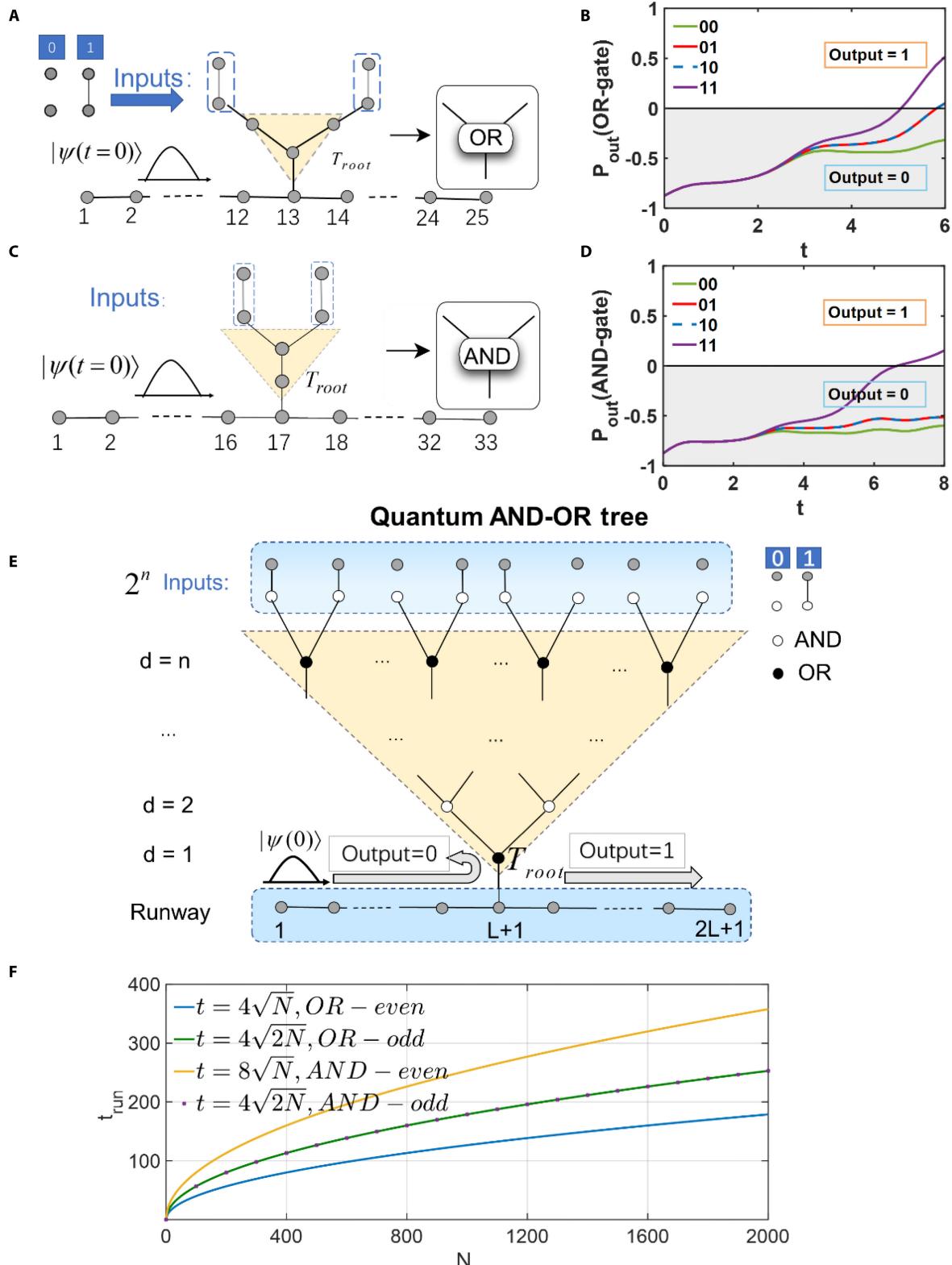
In our study, at time  $t$ , the wave function of the system can be represented as  $\psi(t) = (\psi_1(t), \psi_2(t), \dots, \psi_{32}(t))^T$ . We construct the initial state of the system at time  $t = 0$ ,  $|\psi(0)\rangle = \frac{1}{\sqrt{L}} \sum_{r=2}^{r=13} e^{ir\pi/2} |r\rangle$  [31,32], as shown in the bottom left corner of Fig. 1A. The initial state distribution is located on the left side of the bottom runway. At time  $t$ , the state function evolves into  $|\psi(t)\rangle = e^{iHt} |\psi(0)\rangle$ , and  $t_{run} = \frac{L}{2} = 6$ . During the evolution process, we focus on the probability of the initial wave packet

appearing on either side of the bottom runway. If the probability of the wave packet appearing on the left runway is greater than that on the right runway, the root node outputs 0. Conversely, if the probability of the wave packet appearing on the right runway is greater than that on the left runway, the root node outputs 1. To better illustrate the computation results, we define the output as  $P_{out} = P_{\in R} - P_{\notin R}$ , where  $P_{\in R} = \sum_{L+1 < r < 2L+2} |\langle r|\psi(t)\rangle|^2$  and  $P_{\notin R} = \sum_{r \notin [L+2, 2L+1]} |\langle r|\psi(t)\rangle|^2$ . It can be observed that when  $P_{out} > 0$ , the quantum OR tree evaluates the root node as 1, and when  $P_{out} < 0$ , the quantum OR tree evaluates the root node as 0. Detailed derivations can be found in Section S2.

Due to the symmetry of the 2-input quantum OR tree structure, it effectively encompasses 3 distinct inputs: 00, 01 (or 10), and 11. The temporal evolution of the calculation results  $P_{out}$  for the 2-input quantum OR tree is depicted in Fig. 1B. The various curves in Fig. 1B represent the evolution of the output of the root node over time for the input scenarios 00, 01, 10, and 11, respectively. It is evident that after time  $t > 6$ , the output results for inputs 01 (red solid line), 10 (blue dashed line), and 11 (purple solid line) are  $P_{out} > 0$ , which indicates that the wave packet is more likely to occupy nodes on the right side of the runway. In this scenario, the output of the root node of the tree is 1. In contrast, for the input scenario of 00 (green solid line),  $P_{out}$  is always less than 0. This means that the wave packet is more likely to occupy nodes on the left side of the runway and within the tree, consequently leading to an output of 0 at the root node. That is, we have realized a basic 2-input quantum OR tree.

Similarly, we can also construct a 2-input quantum AND tree, as shown in Fig. 1C. The depth of the quantum AND tree is  $d = 2$ , where  $L$  is chosen as  $L = 8\sqrt{2N} = 16$ . There are a total of  $M = 39$  nodes. It includes 33 nodes of the runway, 2 nodes of the tree structure, and 4 nodes of the input layer. The first 2 rows of nodes also have 2 states, representing the values of inputs 1 and 0, respectively. The root node of the tree is connected to a runway at node 17, in which the length of runway is  $2L + 1 = 33$ . Thus, the entire quantum AND tree can be represented by a  $39 \times 39$  dimensional Hamiltonian  $H$ . Details for the construction of quantum AND tree have also been shown in Section S2. In our study, at time  $t$ , the wave function of the system can be represented as  $\psi(t) = (\psi_1(t), \psi_2(t), \dots, \psi_{39}(t))^T$ . In such a case, the initial state of the system at time  $t = 0$  is expressed as  $|\psi(0)\rangle = \frac{1}{\sqrt{L}} \sum_{r=2}^{r=17} e^{ir\pi/2} |r\rangle$  [31,32]. Its distribution is located on the left side of the bottom runway. At time  $t_{run} = \frac{L}{2} = 8$ , we can obtain the output of the AND tree. The curves in Fig. 1D represent the change of  $P_{out}$  over time for inputs of 00, 01, 10, and 11, respectively. It can be seen that for the input scenario of 11 (purple solid line), after time  $t > 8$ ,  $P_{out} > 0$ , resulting in the output of the tree being 1. In contrast, for the other input scenarios (00, 01, 10),  $P_{out}$  is always less than 0, leading to the output of the root node of the tree being 0. This approach allows us to implement a fundamental 2-input quantum AND tree.

Based on the 2-input quantum AND tree and OR tree, we construct a general quantum AND-OR tree as shown in Fig. 1E, comprising 3 main components: the input layer, the AND-OR tree structure, and the runway. In the input layer (the light blue area in Fig. 1E), connections and disconnections between the 2 rows of nodes correspond to inputs of 1 and 0. They are then processed



**Fig.1.** (A) Diagram for the 2-input quantum OR tree. If the nodes within the blue dashed box at the top are connected (disconnected), the input is 1 (0). (B) Output results of the quantum OR tree. The result is 0 when the input is 00, and it is 1 for inputs 01, 10, and 11. (C) Diagram for the 2-input quantum AND tree. (D) Output result of the quantum AND tree. The output is 0 when the input is 00, 01, and 10, and it is 1 only for the input 11. (E) Schematic diagram of the quantum AND-OR tree. With a depth of  $n$ , the input layer has  $N = 2^n$  inputs, where the connections and disconnections between the nodes of the input layer are represented by 1 and 0, respectively; white nodes represent the AND nodes, while black nodes represent the OR nodes. As the depth of the tree increases, the nodes alternate between black and white. The root node ( $T_{root}$ ) of the tree structure is the OR node, which is connected to a runway of length  $2L+1$ , and the initial  $|\psi(0)\rangle$  is input from the left side of the runway. (F) Relationship between the output result time  $t_{run}$  and the input parameter  $N$  for the 4 different structures of AND-OR trees. These trees are classified by their root nodes (AND nodes or OR nodes) and their depth (odd or even layers). All of them exhibit quantum speedup.

within the AND-OR tree structure (the light yellow area in Fig. 1E). Consequently, the result of the root node is considered as the output of the tree structure. The root node in the bottom layer of the tree structure is connected to a runway of length  $2L + 1$  (the dark blue area in Fig. 1E), where there are  $L$  nodes on each side of the connected point. The nodes on the runway are numbered from 1 to  $2L + 1$ , while the nodes of the tree structure are sequentially numbered from the bottom to the top. Here, we set the value of  $L$  to be  $L = 8 \times 2^{n/2}$  for the quantum AND-OR tree structure of depth  $n$ . There are a total of  $M = 3 \times 2^n + 2^{n/2+4}$  nodes. Each node in Fig. 1E can be represented by a  $M \times 1$  dimensional quantum state, where the state function  $|\psi\rangle = (0, \dots, 0, 1, 0, \dots, 0)^T$  represents that the  $r$ th element of the column vector is 1, and all other elements are 0. Thus, the entire quantum AND-OR tree can be represented by a  $M \times M$  dimensional Hamiltonian  $H$ . The details for the construction of the quantum AND-OR tree have also been presented in Section S3.

Similarly to the quantum OR tree, the wave function of the system at time  $t$  can be represented as  $\psi(t) = (\psi_1(t), \psi_2(t), \dots, \psi_M(t))^T$ . The initial state of the system at time  $t = 0$  is  $|\psi(0)\rangle = \frac{1}{\sqrt{L}} \sum_{r=2}^{r=L+1} e^{ir\pi/2} |r\rangle$  [31,32], as shown in the bottom left corner of Fig. 1E, where the initial state distribution is located on the left side of the bottom runway. The initial state evolves according to the Schrodinger equation:  $i \frac{d}{dt} \psi(t) = H\psi(t)$ . We focus on the output result  $P_{out} = P_{\in R} - P_{\notin R}$  at the time  $t_{run} = \frac{L}{2}$ .

After discussing the fundamental functionalities of the 2-input AND tree and OR tree alongside the quantum AND-OR tree, we proceed to a detailed analysis of their quantum speedup properties. Assuming that all AND-OR trees are of a 2-input nature, 4 distinct structures emerge based on the characteristics of the root node (AND or OR) and the depth level (odd or even).

When the root node of an AND-OR tree is an OR node and the tree depth is even (where  $n$  represents the depth of the tree,  $n = 2k$ ,  $k = 0, 1, 2, \dots$ ), the corresponding quantum walk describing the tree (with depth denoted by  $d$ ) remains unchanged, that is,  $d = n = 2k$ . However, when the tree depth is odd ( $n = 2k + 1$ ,  $k = 0, 1, 2, \dots$ ), an additional layer of depth is added to the input layer of the corresponding quantum game tree, resulting in a tree depth of  $d = n + 1 = 2k + 2$ . Similarly, if the root node is an AND node, and tree depth is even ( $n = 2k$ ,  $k = 0, 1, 2, \dots$ ), both the root node and the input layer of the corresponding quantum tree increase by one layer, resulting in a depth of  $d = n + 2 = 2k + 2$ . In the case of odd ( $n = 2k + 1$ ,  $k = 0, 1, 2, \dots$ ) depth scenarios, the corresponding quantum tree adds one layer at the root node, yielding a tree depth of  $d = n + 1 = 2k + 2$ . Detailed descriptions of these 4 types of AND-OR tree structures based on quantum walk have been presented in Section S4.

Figure 1F illustrates the relationship between the output result time  $t_{run}$  and the input parameter  $N$  for these 4 types of structures. It is noted that this output time corresponds to the duration required for fully distinguishing different results, and the output result in 0 or 1 does not change with a longer time evolution. We examine the evolution trend of theoretical values with changes in the input scale  $N$  for different structures. For each structure, we can conclude that under the condition of input scale  $N$ , the output time satisfies  $t \propto \sqrt{N}$  with  $N$ , thereby demonstrating the quantum speedup.

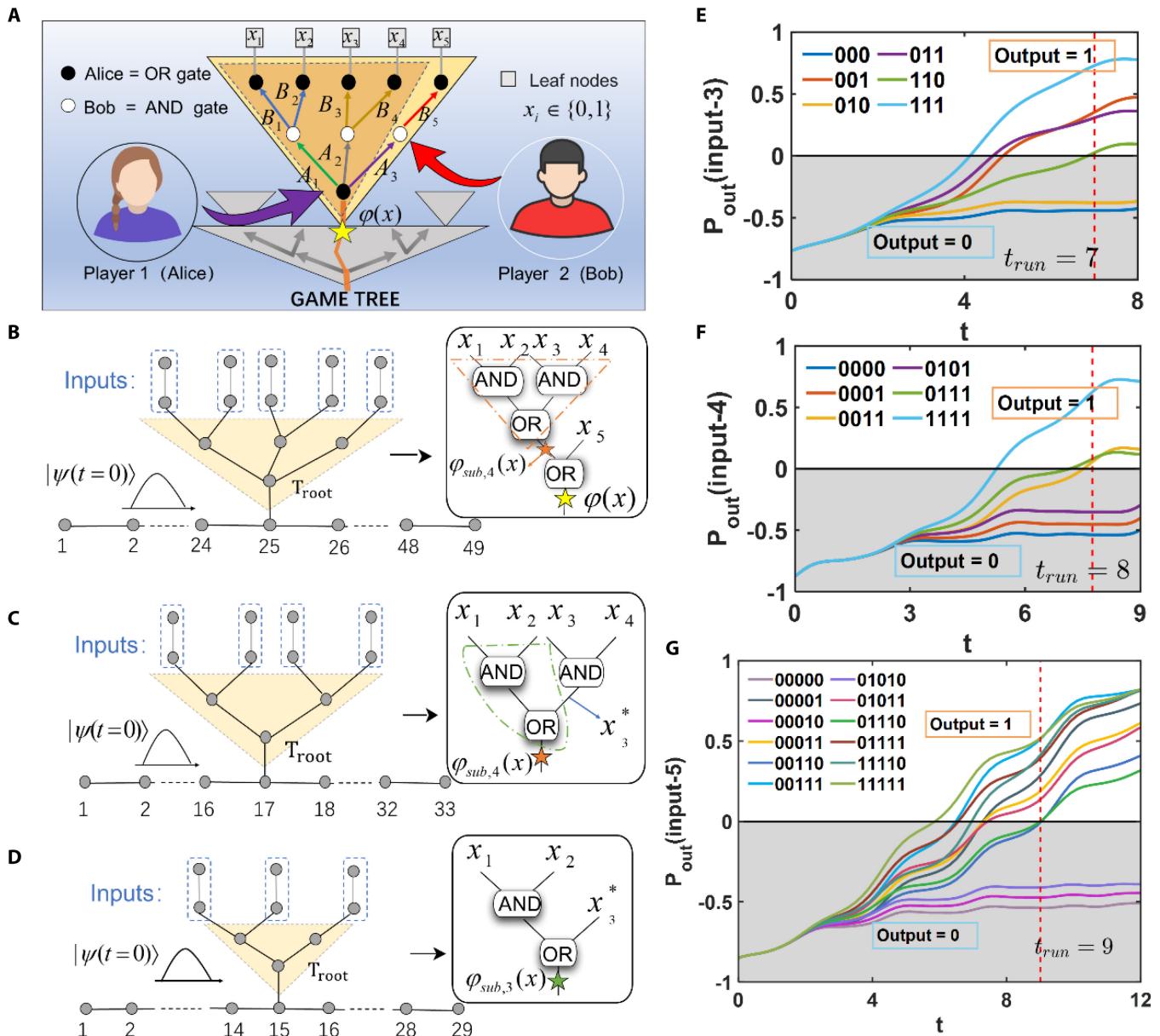
## Description of 2-player zero-sum games

As shown in the previous section, the quantum speedup in the quantum AND-OR tree has been demonstrated. In the following, we give the description of 2-player zero-sum games at first; then, we map this game problem to the quantum AND-OR tree and show the quantum speedup.

The game process is typically represented by the game tree illustrated in Fig. 2A. During the game, the first player (Alice) makes the “OR” operation of their child nodes, and the second player (Bob) does the “AND” operation of their child nodes. These are labeled by the black and white dots, respectively, in Fig. 2A. One round of game contains one “OR” and “AND” operations. In one round, Alice can make a choice at the beginning, and Bob wants to choose the strategy to win, no matter what moves made by Alice before. In this way, a new round of game is run. Alice needs to find a new strategy to win in the new situation, and the “OR” operation is made by Alice again. Therefore, the “OR” and “AND” operations (black and white dots in Fig. 2A) appear alternatively during the game process. The arrows in different colors in Fig. 2A represent different choices. When Alice chooses the decision  $A_1$  (the green arrow in Fig. 2A), Bob can choose between  $B_1$  and  $B_2$  (blue arrows); when Alice chooses  $A_2$  (gray arrow), Bob can choose between  $B_3$  and  $B_4$  (gold arrows); and when Alice chooses  $A_3$  (purple arrows), Bob can only choose  $B_5$  (red arrows). The game is terminated when reaching terminal leaf nodes (rectangles in the top of Fig. 2A). Each leaf node contains a variable  $x_i$  with the value of 1 or 0. Based on the inputs of the terminal leaf nodes, we can quickly determine the decision paths of Alice and Bob, and evaluate the output result of the root node (the yellow star in Fig. 2A) of the game tree. In the game between Alice and Bob, Alice wins if the root node value equals 1; conversely, Bob prevails when the root node value is 0.

The game problem addressed in Fig. 2A can be mapped to a 5-input quantum AND-OR tree, as illustrated in Fig. 2B, which is composed of dashed brown subtrees representing a 4-input subtree (the light yellow solid subtree in Fig. 2A) and a 2-input OR tree. Now, we employ a subgame design technique to develop a quantum algorithm for the Hamiltonian AND-OR tree using continuous quantum walk and then realize the quantum speedup in the gameplay process with a 5-input quantum AND-OR tree. The formula representation of this tree is as follows:  $\varphi(x) = (x_1 \wedge x_2) \vee (x_3 \wedge x_4) \vee x_5$ ,  $x_i \in \{0, 1\}$ , where  $\wedge$  and  $\vee$  denote AND nodes and OR nodes, respectively. The 5 input values are  $x_1, x_2, x_3, x_4$ , and  $x_5$ ;  $\varphi(x)$  represents the result of the calculation of the root node of the 5-input quantum AND-OR tree. Additionally, the AND-OR tree can further decompose into a smaller 4-input subtree and a 2-input OR tree,  $\varphi(x) = \varphi_{sub,4}(x) \vee x_5$ ,  $x_i \in \{0, 1\}$ . The operational formula for the 4-input subtree is as follows:  $\varphi_{sub,4}(x) = (x_1 \wedge x_2) \vee (x_3 \wedge x_4)$ ,  $x_i \in \{0, 1\}$ , as shown in Fig. 2B. The 4-input subtree can then be further decomposed into a smaller 3-input subtree and a 2-input AND tree (see the inset of Fig. 2C). The 3-input subtree, as depicted in Fig. 2D, is characterized by the following operational formula:  $\varphi_{sub,3}(x) = (x_1 \wedge x_2) \vee x_3^*, x_i \in \{0, 1\}$ , where  $x_3^*$  represents the result of the calculation for the input  $x_3$  and  $x_4$ .

For the 3-input subtree, here we take the value of  $L$  to be  $L = 8\sqrt{3} \approx 14$ . The top row of inputs in Fig. 2D can be in connected or disconnected states, corresponding to input values of 1 and 0, respectively. The lowest root node in the tree structure is connected to a runway of length 29, in which the connection



**Fig. 2.** (A) Alice and Bob double zero-sum game tree. The entire game tree is made up of numerous subtrees. Each gray triangle represents a subtree, while the orange-yellow line depicts the optimal path throughout the entire gameplay process. The black nodes represent the choice of Alice (OR node), and the white nodes represent the choice of Bob (AND node). The input values of the terminal nodes  $x_i$  are only 0 and 1. The green arrows denote the strategies available to Alice, with the strategy  $A_1|B_1, B_2$ ,  $\{A_2|B_3, B_4\}$ , and  $\{A_3|B_5\}$ . The dashed brown triangles and the solid yellow triangles represent the 4-input subtree and the 5-input subtree, respectively. (B) Schematic diagram of the 5-input AND-OR tree. On the right side is the corresponding classical game tree structure. (C) Schematic diagram of the 4-input quantum AND-OR tree. (D) Schematic diagram of the 3-input quantum AND-OR tree. (E) Results for a 3-input quantum AND-OR tree. The inputs 000 and 010 result in an output of 0, while the inputs 001, 011, 110, and 111 lead to an output of 1. (F) Results for the 4-input quantum AND-OR tree. The inputs 0000, 0001, and 0101 result in an output of 0. The inputs 0011, 0111, and 1111 result in an output of 1. (G) Results for the 5-input quantum AND-OR tree. The inputs 00000, 00010, and 01010 result in an output of 0. All other inputs result in an output of 1. The red dashed line represents the measurement time.

node is 15. This tree structure features 6 independent inputs: 000, 001, 010 (100), 011 (101), 110, and 111. The results of the calculation of the output of the 3-input subtree over time are depicted in Fig. 2E. From the graph, it is evident that after time  $t > t_{\text{run}} \left( \frac{L}{2} = 7 \right)$ , the input types: 110 (green line), 011 (purple line), 001 (orange-red line), and 111 (light blue line) have output  $P_{\text{out}} > 0$ . This signifies that under these input types, the wave packets of the input predominantly occupy the right side of

the runway, leading to a computation result of 1 at the root node. On the contrary, the results  $P_{\text{out}}$  for input 000 (blue line) and 010 (orange-yellow line) consistently remain less than 0. This suggests that the wave packets of the input are predominantly reflected onto the left side. Consequently, the corresponding inputs yield a result of 0. This implies that in the game played within this 3-input subtree, if the inputs are 110, 011, 001, or 111, Alice wins, whereas if the inputs are 000 or 010, Bob wins.

Furthermore, by integrating the 3-input subtree with a 2-input AND tree, a larger 4-input subtree is obtained, as depicted in Fig. 2C. We take the value of  $L$  to be  $L = 8\sqrt{4} = 16$ . Due to structural symmetry, there are only 6 distinct input combinations for the 4 inputs: 0000, 0001 (0010, 0100, 1000), 0011 (1100), 0101 (1010, 0110, 1001), 0111 (1011, 1101, 1110), and 1111. The time-varying calculation results of the output of the 4-input quantum AND-OR tree are shown in Fig. 2F. It is evident that after time  $t > 8$ , the output  $P_{out} > 0$  for 0111 (green line), 0011 (orange-yellow line), and 1111 (light blue line) indicate that under these input types, the wave packets predominantly occupy the right side of the runway. In these respective input scenarios, the calculation results of the 4-input quantum AND-OR tree is 1. For comparison, it is evident from the figure that the output  $P_{out}$  for 0000 (deep blue line), 0001 (orange-red line), and 0101 (purple line) consistently remain below 0. Under these input types, the wave packets are predominantly reflected to the left side of the runway, resulting in a corresponding calculation result of 0. This implies that in the larger 4-input subtree game, if the inputs are 0011, 0111, or 1111, Alice wins; however, if the inputs are 0000, 0001, or 0101, Bob wins. It should be noted that, compared to the 3-input subtree earlier, if Alice wins in the 3-input subtree game, then in the 4-input subtree game, regardless of the value of the fourth input, Alice will ultimately win.

Compared to the 4-input subtree, the final 5-input quantum AND-OR tree introduces an additional input while maintaining structural symmetry. There are a total of 12 independent combinations of inputs, namely, 00000, 00001, 00010, 00011, 00110, 00111, 01010, 01011, 01110, 01111, 11110, and 11111. For the 5-input AND-OR tree, we take the value of  $L$  to be 24. The small deviation away from  $8\sqrt{5} \approx 18$  is due to the asymmetric input in the middle (Fig. 2B).

The time-varying calculation results of the output for the 5-input quantum AND-OR tree are depicted in Fig. 2G. It is noteworthy that the output results  $P_{out}$  for 00000 (brown line), 00010 (pink line), and 01010 (purple line) consistently remain below 0. This means that under these input types, the wave packets predominantly concentrate on the left side of the runway, resulting in an output value of 0. In contrast, it is evident from the figure that, after time  $t > 9$ , the output results  $P_{out}$  for the remaining 9 cases are all greater than 0. Under these inputs, the wave packets predominantly occupy the right side of the runway, resulting in an output value of 1. Similarly to the previous description, for the 5-input game tree, if Alice wins in the 4-input subgame, regardless of the input of the fifth bit in the 5-input game tree, Alice will ultimately win the 5-input game.

Therefore, for the 3-input subtree, the 4-input subtree, and the 5-input game tree, from the study above, we can conclude that with the input scale  $N$ , the time to obtain the output result satisfies  $t \propto \sqrt{N}$  with  $N$ , which demonstrates the quantum speedup. Since then, we have introduced the design of a 2-player zero-sum game tree based on quantum walk and presented the corresponding results. In the following discussion, we explore how to design classical circuit networks to simulate these quantum game trees.

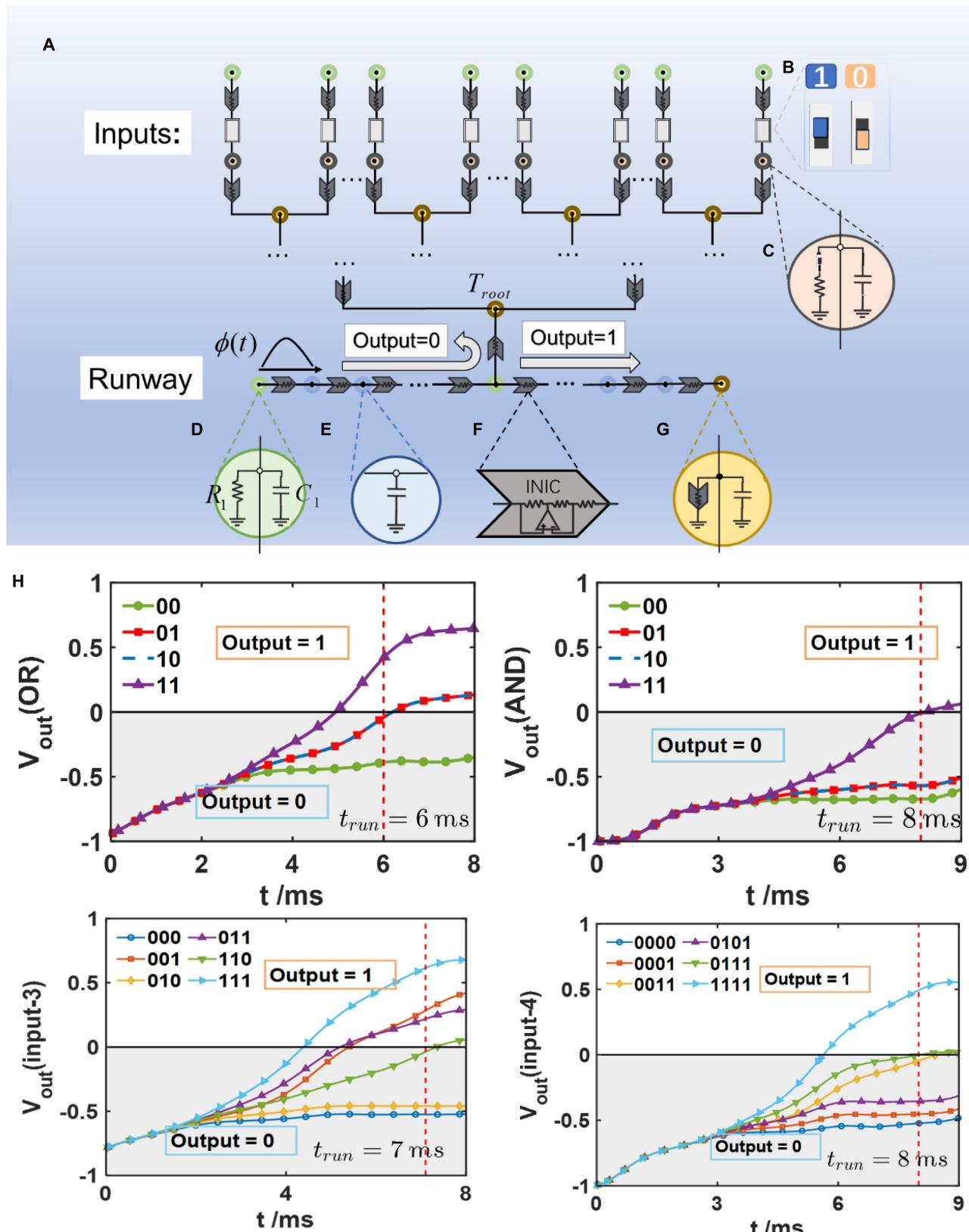
## Circuit Designs of Quantum 2-Player Games

In this section, we provide the circuit design for a general quantum AND-OR tree. As depicted in Fig. 3A, each circle corresponds to a node in the structural diagram shown in Fig. 1E, with a total of  $M = 3 \times 2^n + 2^{n/2+4}$  nodes in the entire circuit structure. Similarly to the numbering of nodes in Fig. 1E, we start by numbering the nodes on the runway and then proceed

to number the nodes from the bottom to the top of the tree structure. The voltage states on these nodes can be represented by column vector  $\phi(t) = (V_1(t), V_2(t), \dots, V_M(t))^T$ , where the voltage  $V_i(t)$  represents the voltage value at the  $i$ th circuit node at time  $t$ . In Fig. 3A, the top 2 rows of nodes in the circuit are connected via dual in-line package (DIP) switches. These DIP switches can be adjusted to be connected or disconnected, corresponding to input values of 1 or 0, as shown in Fig. 3B. In Fig. 3A, each node is grounded using a capacitor resistor, with the grounding method varying according to the design requirements. The nodes in the circuit network are grounded via capacitors and resistors or negative impedance converter (NIC) modules [33,34], as shown in Fig. 3C, D, E, and G. Each node is connected via a NIC, as depicted in Fig. 3F. This structure consists of an operational amplifier, 2 resistors connected to the positive and negative terminals of the operational amplifier, and a resistor  $R_{i,j}$  (representing the effective resistance from node  $i$  to node  $j$ ). The NIC module can reverse the direction of current flow from node  $i$  to node  $j$  and from node  $j$  to node  $i$  [34–36]. In addition, the descriptions of the Kirchhoff equation set for the  $i$ -th current node are provided in the Methods section, and the correspondence between these equations and the Schrödinger equation is shown in S5 of the Supplementary Materials.

At the initial moment, 1-V DC voltage is applied to the corresponding  $L$  nodes on the runway left side, with the remaining nodes grounded. After the evolution begins, we disconnect the initial voltage of 1 V and grounding from the remaining nodes. The voltage state  $\phi(t)$  evolves according to Kirchhoff's current law. During the evolution of the circuit, the operating time  $t_{circuit}$  of the circuit equations differs from the theoretical operating time  $t_{theory}$  of the system by only one constant term:  $t_{circuit} = t_{theory} * (R_i C_i)$  [32,37]. In our simulations and experiments, the resistance  $R_i$  is set to 10 k $\Omega$ , and the capacitance  $C_i$  is set to 100 nF. Therefore, in experiments, 1 ms corresponds to a theoretical time unit of 1. The output result can also be represented by subtracting the voltage of the remaining nodes from the voltage at the right end of the runway, that is,  $V_{out}(t) = \sum_{i \in R} |V_i(t)|^2 - \sum_{i \notin R} |V_i(t)|^2$ . This way, by measuring the voltage values of circuit nodes at different times using an oscilloscope, we can determine the computation result of the quantum AND-OR tree.

In Fig. 3H, we present the circuit simulation results of the 2-input quantum OR tree, 2-input quantum AND tree, 3-input subtree, and 4-input subtree relevant to the game process. The design diagrams of the corresponding circuit structures are included in Section S6. For the 2-input quantum OR tree, the results indicate that at time  $t > 6$  ms, the output is 1 ( $V_{out} > 0$ ) only when the inputs are 11 (purple upward-pointing triangle line), 10 (blue dotted line), and 01 (red square line), while for all other inputs, it remains 0 ( $V_{out} < 0$ ). This simulation result aligns with the theoretical results in Fig. 1B. For the 2-input quantum AND tree, at time  $t > 8$  ms, the output is 1 ( $V_{out} > 0$ ) only when the input is 11 (purple triangle upward-pointing line). This simulation result is also consistent with the theoretical results in Fig. 1D. In the 3-input subtree, the formula is represented as:  $\varphi_{\{sub,3\}}(x) = (x_1 \wedge x_2) \vee x_{3*}$ ,  $x_i \in \{0, 1\}$ . It is only after time  $t > 7$  ms that the output is 1 when the inputs are 001 (orange square line), 011 (purple upward-pointing triangle line), 110 (green downward-pointing triangle line), and 111 (blue rightward-pointing triangle line). For the 4-input subtree, the



**Fig.3.**(A) Circuit design of the quantum AND-OR tree. (B) The connections and disconnections of the switches at the top represent inputs 1 and 0, respectively. (C) The pink nodes in the middle are connected to the ground through capacitors and resistors in parallel. (D) When the switch in the pink node is connected to the child node (input is 1), the switch in the pink node should be closed, that is, the capacitor and resistor are in parallel with the ground. (E) When the switch is open, the pink node should only be disconnected from the capacitor and connected to the ground. (F) Schematic diagram of the negative impedance converter (NIC), which includes an operational amplifier, 2 positive and negative resistors, and an equivalent resistor. (G) Nodes are connected to the ground in parallel through NIC modules and capacitors. (H) Simulation results of the 2-input quantum OR tree, the 2-input quantum AND tree, the 3-input AND-OR tree, and the 4-input AND-OR tree, which are consistent with the theoretical calculations in Fig. 2.

formula is:  $\varphi_{\{sub,4\}}(x) = (x_1 \wedge x_2) \vee (x_3 \wedge x_4)$ ,  $x_i \in \{0, 1\}$ . At time  $t > 8$  ms, the output is 1 ( $V_{out} > 0$ ) only when the inputs are 0011 (yellow diamond line), 0111 (green pointing downward triangle line), and 1111 (blue pointing rightward triangle line). The circuit simulation results for the 3-input subtree and 4-input subtree correspond to the quantum theoretical results (see Fig. 2E and F). Similarly, for the more complex quantum AND-OR tree structure, we can also utilize the classical circuit designs mentioned above. In Section S6, we also provide the circuit design and results for the 5-input game tree. Upon completing the circuit design and simulation, we then proceed to discuss the experimental realization of the quantum game tree circuit.

## Experimental realizations of quantum 2-player games

Based on the aforementioned circuit designs, we discuss how to proceed with experimental implementation. First, we demonstrate the experimental implementation of a 2-input quantum OR tree. In Fig. 4A, the physical printed circuit board (PCB) of the 2-input quantum OR tree is displayed. The size of the board is 35 cm  $\times$  25 cm. The circuit structures prepared for the experiment correspond one to one with the theoretical design depicted in Fig. 3A.

In Fig. 4A,  $a_1$  denotes push button switches used to control the operation of relays, while  $a_2$  represents DIP switches. The switches in the open position signify input 0, while those in the closed position signify input 1.  $a_3$  represents relays (model G6K-2F-Y-5VDC), used to simultaneously disconnect each node after applying the initial voltage, allowing the system to evolve from the initial time.  $a_4$  represents 2 NICs, used to connect adjacent nodes. The capacitors are 100 nF, and the resistors are 10 k $\Omega$ . The capacitance and resistance parameters chosen for the experiment determine the values of the circuit matrix.  $a_5$  is a pin header used to input the initial voltage and the voltage values of the complex impedances to the circuit board. The silver dashed box represents the runway, the blue dashed triangles represent the quantum OR tree, and the red dashed circle represents the root node of the tree.

For the 2-input quantum OR tree, during the circuit evolution, the 4 different inputs (00, 01, 10, 11) can be controlled by opening or closing the DIP switches. Each node is connected to the initial voltage via relays. The initial state of the circuit

system is denoted as  $\phi = \frac{1}{\sqrt{L}} \begin{pmatrix} 0, 1, \underbrace{\dots, 1}_L, 0, \dots, 0 \end{pmatrix}^T V$ , where all

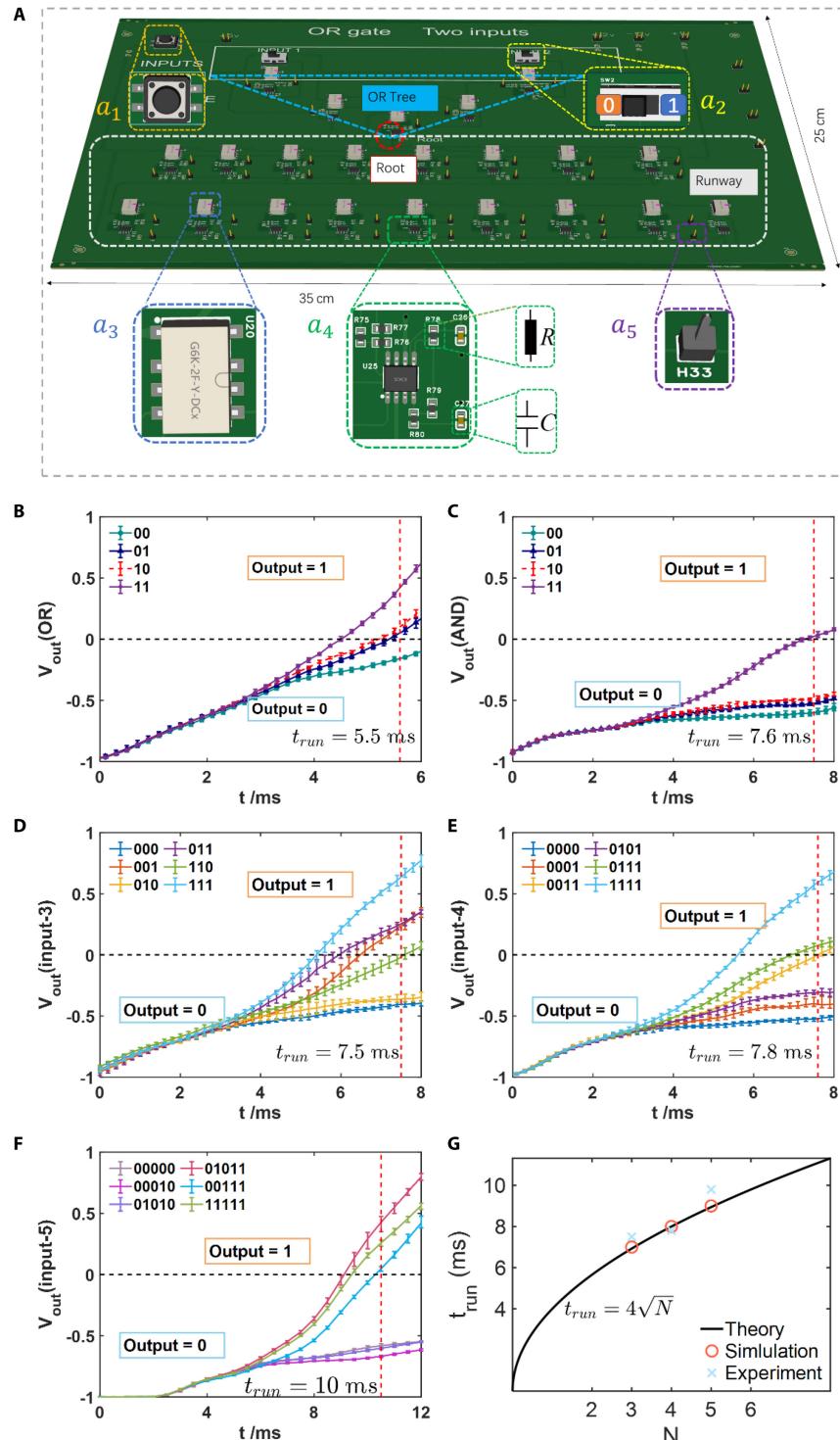
nonzero voltages are initially on the left side of the runway ( $r \leq L + 1$ ). When the push button switch is disconnected, the relays are opened, and each node is disconnected from the initial voltage, initiating the system's evolution. During the evolution, voltages gradually appear at the nodes on the right side of the tree in the experiment. Using an oscilloscope to measure the voltage values of the nodes and calculating the difference between the sum of the squares of voltages at the rightmost nodes and the sum of the squares of voltages at non-rightmost nodes ( $V_{out}(t) = \sum_{i \in R} |V_i(t)|^2 - \sum_{i \notin R} |V_i(t)|^2$ ), we can obtain the output value of the tree's root node.

In Fig. 4B, the experimental results for the 2-input quantum OR tree are depicted, where green, blue, red, and purple represent the evolution of output node voltages in the circuit

structure for inputs 00, 01, 10, and 11, respectively. Each line accompanied by error bars represents the results averaging 5 times in experiments (the following results in Fig. 4C to F are similar). It can be observed that after time  $t > 5.5$  ms, an output result of 1 ( $V_{out} > 0$ ) is obtained only when the input is 01 (10) or 11, while the other inputs remain at 0 ( $V_{out} < 0$ ). It is noted that components in the circuit network, such as capacitors, resistors, and operational amplifiers, may have inherent errors, whereas the LTspice simulation software employs ideal components. Consequently, there may be slight discrepancies in time between simulation and experiment results. The experimental evolution results are nearly in agreement with the theoretical simulations. We have discussed the experimental error in Section S7. Nevertheless, we have observed that these variances fall within the expected range. This means that the experimental results are reliable.

In Fig. 4C, the experimental results for the 2-input quantum AND tree are illustrated, where green, blue, red, and purple represent the evolution of voltages in the circuit structure for inputs 00, 01, 10, and 11, respectively. It can be observed that after time  $t > 7.6$  ms, an output result of 1 is obtained only when the input is 11, while the other inputs remain at 0. Such input results indeed correspond to the functionality of the AND tree. Furthermore, Fig. 4D presents the experimental results for the 3-input subtree. The deep blue, orange-red, orange-yellow, purple, light green, and light blue solid lines represent the evolution of voltages in the circuit structure for inputs 000, 001, 010, 011, 110, and 111, respectively. It can be observed that after time  $t > 7.5$  ms, an output result of 1 is obtained only when the inputs are 001, 011, 110, or 111, while for other inputs, the output remains at 0. We also present the experimental results for the 4-input subtree and the 5-input game tree in Fig. 4E and F. For the 4-input subtree, after time  $t > 7.8$  ms, an output result of 1 is obtained only when the inputs are 0011 (yellow line), 0111 (green line), or 1111 (light blue line), while for other inputs, the output remains at 0. Similarly, for the 5-input quantum AND-OR game tree, when the inputs are 00000 (gray line), 00010 (pink line), and 01010 (purple line), the output result at the root node remains at 0 ( $V_{out} < 0$ ). Conversely, for other inputs such as 01011 (red line), 00111 (blue line), and 11111 (green line), after time  $t > 10$  ms, the output result is 1 ( $V_{out} > 0$ ). The output results for other inputs are provided in Section S8. The experimental results described above correspond well to the circuit simulation results shown in Fig. 3H.

To showcase the quantum speedup of the AND-OR tree in circuit realization, we analyze the time required to distinguish between different outputs based on the tree's input. Specifically, we select the root node as the OR node and set the depth of the AND-OR tree to be even. The relationship between the input  $N$  and the output time  $t$  is provided in Fig. 4G. According to the quantum theory above (section "Theoretical Scheme of Quantum 2-Player Zero-Sum Games"), it needs the time as  $t \propto \sqrt{N}$  with the input bit number, which has been depicted as the solid black line in Fig. 4G. These theoretical results are obtained from the 3-input subtree, the 4-input subtree, and the 5-input game tree in Fig. 2. Here, the unit of time is chosen as millisecond, which corresponds to our circuit design. As shown in our circuit simulation (Fig. 3) and experiment (Fig. 4), the quantity  $P_{out}$  has been chosen to show the distribution of probability on the runway. For all cases showing the output of 1, we choose the critical time at which all  $P_{out}$  are becoming larger than zero. This time is viewed as  $t_{run}$ . In Fig. 4G, red circles are the simulation results



**Fig. 4.** (A) Printed circuit board (PCB) of the 2-input quantum OR tree structure. Detailed components of the quantum OR tree:  $a_1$ , button switch to control relay operation;  $a_2$ , DIP switch to select inputs (left for 0, right for 1), providing 3 independent inputs: 00, 01 (or 10), and 11;  $a_3$ , relay (model: G6K-2F-Y-5VDC) to control the initial state of the circuit board;  $a_4$ , two negative impedance modules (LT1013) within the gray box (resistor and capacitor highlighted with green dashed lines);  $a_5$ , pin headers to introduce initial voltages and provide impedance to the system, as well as for voltage measurements at each node. (B) Experimental results of the 2-input quantum OR tree: 00 (green line), 01 (blue line), 10 (red dashed line), and 11 (purple line) are depicted. Each line is accompanied by error bars representing the results obtained from 5 experiments. (C) Experimental results of the quantum AND tree of 2-input. Inputs are denoted by line colors similar to those in (B). (D) Experimental results of the 3-input quantum AND-OR tree: Among the outputs, 000 (blue line) and 010 (orange-yellow line) are 0. For other inputs, 001 (orange line), 011 (purple line), 110 (green line), or 111 (light blue line), the output is 1. (E) Experimental results of the 4-input quantum AND-OR tree. The outputs for 0000 (blue line), 0001 (orange-yellow line), and 0101 (purple line) are 0; all other output results are 1. (F) Experimental results of the 5-input quantum AND-OR tree. The outputs for 00000 (gray line), 00010 (pink line), and 01010 (purple line) are 0; for other inputs such as 01011 (red line), 00111 (blue line), and 11111 (green line), the output results are all 1. (G) Relationship between the input  $N$  and the output time  $t$  for the 3-input subtree, the 4-input subtree, and the 5-input game tree. The solid line represents the theoretical calculation value, the red circle represents the simulated time, and the blue cross represents the experimentally measured time.

that are obtained from the time for the 3-input subtree, the 4-input subtree, and the 5-input game tree in the circuit simulation in Section 3. Blue navy crosses denote the times from the experiment (Fig. 4D to F). As shown in Fig. 4G, it can be observed that the theoretical and simulated times strictly coincide with each other. However, due to the relatively small value of  $N$ , there is a slight discrepancy between the experimental and simulated time. The experimental results are distributed around the theoretic results (blue solid line), which also demonstrates the quantum advantage in our circuit platform.

## Discussion

Currently, the quantum circuit model can process multiple states simultaneously, potentially offering computational speedup in solving complex game problems. However, quantum computing is still constrained by noise and hardware limitations. Although the theoretical proposal of the balanced NAND-gate on the conjugated organic molecular structure is achieved [38], it is not easily implemented in reality due to the hardness in the accurate control of the internal structure of molecules and electron movement. Recent study shows that the balanced NAND-gate algorithm is tested in a photonic waveguide [39], but fixed inputs and a lack of tunability increased resource consumption and limited its application. In this way, the platforms with the conjugated organic molecules or the waveguides are not able to display the complicated functions required in the game problem easily. Compared with other platforms, the circuit has a high maturity and scalability, and can handle 2-player game problems stably and efficiently. In addition, the circuit is deterministic and robust in the calculation results, which is suitable for solving game problems that require high accuracy.

In addition, we have successfully implemented an 8-input AND-OR tree and conducted demonstrations on a PCB platform. Indeed, as the number of components increases, the required area of the PCB expands, and the accumulation of component errors increases, thereby affecting the output results. However, by mapping the implementation from a PCB to an integrated circuit (IC) chip, these issues can be effectively addressed. As shown in [40,41], the extremely low error rates in the chip enable the realization of low parasitic capacitance and inductance, thereby reducing signal distortion and error accumulation, and improving the precision and stability of the circuit. Implementing the game problem using IC chips offers the following advantages. For instance, the chip, which contains thousands of complementary metal-oxide semiconductor (CMOS) transistors, is fabricated using a 65-nm CMOS process technology, with a size of  $3,000 \times 3,500 \mu\text{m}^2$  [41]. Based on this component count, it is estimated that up to 100 inputs for the game problem can be realized. Because of these advantages, the implementation of the game problem on ICs can scale up to larger inputs while keeping the circuit size within the millimeter range.

In this work, we design a quantum algorithm based on a quantum walk and experimentally verify the quantum speedup on a classical circuit platform. We map the 2-player zero-sum game to the AND-OR tree and design the quantum AND-OR tree to solve the game problem. In our study, we choose the 5-input game tree as an example and divide it into a 4-input subtree and a 3-input subtree in sequence. Our results verify the consistency between the optimal path of a subtree and that

of the entire game tree. More importantly, the time for obtaining the output result satisfies  $t \propto \sqrt{N}$  with the  $N$  inputs, which has the good correspondence to the time in the quantum algorithm. Circuit simulations and experimental results realize a new type of classical computing based on the quantum intermediary construction. Furthermore, the solution for the 2-player zero-sum game can be obtained within the time as that from the quantum algorithm. Although we only discuss the 2-player zero-sum game problem here, it is also possible to use the concept of the game tree to demonstrate the quantum speedup for more complex game problems, including chess, economics, cybersecurity, computer science, and finance, among others. The quantum speedup in the game problem here provides a new idea for improving computing power in the era of big data and will be widely used in various fields.

## Methods

In the circuit, the Kirchhoff equation set for the  $i$ -th current node can be expressed as:

$$\begin{aligned} C_i \frac{dV_i(t)}{dt} + \frac{V_i(t)}{R_i} &= \sum_j \frac{V_j(t) - V_i(t)}{R_{i,j}} \\ \frac{dV_i(t)}{dt} &= \frac{1}{RC_i} \sum_j [V_j(t) - V_i(t)] \end{aligned} \quad (1)$$

Here, the capacitance value  $C_i$  and the resistance value  $R_i$  represent the capacitance and grounding resistance values of the  $i$ th circuit node, respectively. The node equation set for the entire circuit system can be expressed as:

$$\frac{d\phi(t)}{dt} = A\phi(t) \quad (2)$$

The matrix  $A$  is a real matrix of size  $M \times M$ , and its elements are determined by the capacitance and resistance between the nodes of the circuit. If we multiply Eq. 2 by the imaginary unit  $i$ , we obtain  $i \frac{d\phi(t)}{dt} = iA\phi(t) = \mathcal{H}\phi(t)$ . In our study, we choose the appropriate grounding resistor values so that the diagonal elements of the matrix  $A$  are all zero. Then, the effective resistance values of the NICs are adjusted to ensure that the matrix  $\mathcal{H}$  satisfies the condition of being a Hermitian matrix. The Matrix  $\mathcal{H}$  and the Hamiltonian  $H$  can be obtained through similarity:  $\mathcal{H} = PHP^{-1}$ . We can get:

$$i \frac{d}{dt} (P^{-1}\phi(t)) = H(P^{-1}\phi(t)) \quad (3)$$

In the equation,  $P$  represents the similarity transformation matrix, whose specific form is provided in Section S5. If we regard  $P^{-1}\phi(t)$  as the wave function  $\psi(t)$  in the Schrödinger equation, then the evolution of the circuit system exactly corresponds to the time evolution of the quantum system. It is

found that the initial state  $\phi(0) = \frac{1}{\sqrt{L}} \begin{pmatrix} 0, 1, 1, \dots, 1, 0, 0, \dots, 0 \end{pmatrix}^T$  in circuit evolution is similar to the initial wave function designed in quantum theory, where the voltage unit is set as  $\sqrt{LV^{-1}}$ .

## Acknowledgments

**Funding:** This work was supported by the National Key R&D Program of China under grant no. 2022YFA1404900 and the National Natural Science Foundation of China (12234004 and 12374323).

**Author contributions:** J.Z. provided the theoretical analysis cooperated with T.C. J.Z. performed the experiments and analyzed the data with the help of T.C., W.D., X.T., and X.Z. X.Z. initiated and designed this research project.

**Competing interests:** The authors declare that they have no competing interests.

## Data Availability

All data needed to evaluate the conclusions of the study are present in the paper and the Supplementary Materials. Additional data related to this paper may be requested from the author upon reasonable request.

## Supplementary Materials

Sections S1 to S8  
Figs. S1 to S8

## References

1. Lucas WF. An overview of the mathematical theory of games. *Manag Sci.* 1972;18(5-part-2):3–19.
2. Von Neumann J, Morgenstern O. *Theory of games and economic behavior*. 2nd ed. Princeton (NJ): Princeton University Press; 1947.
3. Ewerhart C. Backward induction and the game-theoretic analysis of chess. *Games Econ Behav.* 2002;39(2):206–214.
4. Alkheliwi T. Applying game theory rules to enhance decision support systems in credit and financial applications. In: *2014 Computer Games: AI Mobile, Multimedia, Educational and Serious Games (CGAMES)*. Louisville (KY): IEEE: Animation; 2014.
5. Pijls W, de Bruin A. Game tree algorithms and solution trees. *Theor Comput Sci.* 2001;252(1-2):197–215.
6. Hsu F-H. *Behind deep blue: Building the computer that defeated the world chess champion*. Princeton (NJ): Princeton University Press; 2002.
7. Coulom R. Efficient selectivity and backup operators in Monte-Carlo tree search. In: *International Conference on Computers and Games*. Berlin, Heidelberg: Springer Berlin Heidelberg; 2006.
8. Kocsis L, Szepesvári C. Bandit based Monte-Carlo planning. In: *European Conference on Machine Learning*. Berlin, Heidelberg: Springer Berlin Heidelberg; 2006.
9. Coulom R. Computing ‘Elo ratings’ of move patterns in the game of go. *ICGA J.* 2007;30(4):198–208.
10. Baudiš P, Gailly J, Pachi: State of the art open source Go program. *Adv Comput Games.* 2011:24–38.
11. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, van den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, et al. Mastering the game of Go with deep neural networks and tree search. *Nature.* 2016;529(7587):484–489.
12. Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A, et al. Mastering the game of Go without human knowledge. *Nature.* 2017;550(7676):354–359.
13. Silver D, Hubert T, Schrittwieser J, Antonoglou I, Lai M, Guez A, Lanctot M, Sifre L, Kumaran D, Graepel T, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science.* 2018;362(6419):1140–1144.
14. Stockman GC. A minimax algorithm better than alpha-beta? *Artif Intell.* 1979;12(2):179–196.
15. Saks M, Wigderson A. Probabilistic Boolean decision trees and the complexity of evaluating game trees. In: *27th Annual Symposium on Foundations of Computer Science (SFCS 1986)*. Toronto (ON, Canada): IEEE; 1986.
16. Campbell MS, Marsland TA. A comparison of minimax tree search algorithms. *Artif Intell.* 1983;20(4):347–367.
17. Elnaggar AA. A comparative study of game tree searching methods. *Int J Adv Comput Sci Appl.* 2014;5(5):68–77.
18. Eisert J, Wilkens M, Lewenstein M. Quantum games and quantum strategies. *Phys Rev Lett.* 1999;83(15):3077.
19. Du J. Experimental realization of quantum games on a quantum computer. *Phys Rev Lett.* 2002;88(13): Article 137902.
20. Grover LK. A fast quantum mechanical algorithm for database search. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. Philadelphia (PA): ACM; 1996. p. 212–219.
21. Ambainis A. Quantum search algorithms. *ACM SIGACT News.* 2004;35(2):22–35.
22. Cleve R, Gavinsky D, Yonge-Mallo D.L., Yonge-Mallo DL. Quantum algorithms for evaluating min-max trees. In: *Workshop on Quantum Computation, Communication, and Cryptography*. Berlin, Heidelberg: Springer Berlin Heidelberg; 2008.
23. Barnum H, Saks M. A lower bound on the quantum query complexity of read-once functions. *J Comput Syst Sci.* 2004;69(2):244–258.
24. Childs AM, Cleve R, Deotto E, Farhi E, Gutmann S, Spielman DA. Exponential algorithmic speedup by a quantum walk. In: *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*. San Diego (CA): ACM; 2003.
25. Childs AM. Universal computation by quantum walk. *Phys Rev Lett.* 2009;102(18):Article 180501.
26. Ambainis A, Childs AM, Reichardt BW, Špalek R, Zhang S. Any AND-OR formula of size  $N$  can be evaluated in time  $N^{1/2+o(1)}$  on a quantum computer. *SIAM J Comput.* 2010;39(6):2513–2530.
27. Meyer DA, Wong TG. Connectivity is a poor indicator of fast quantum search. *Phys Rev Lett.* 2015;114(11):Article 110503.
28. Weng C-X. Beating the fault-tolerance bound and security loopholes for byzantine agreement with a quantum solution. *Research.* 2023;6:0272.
29. Reichardt BW. Faster quantum algorithm for evaluating game trees. In: *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*. San Francisco (CA): Society for Industrial and Applied Mathematics; 2011.
30. Chakraborty S. Spatial search by quantum walk is optimal for almost all graphs. *Phys Rev Lett.* 2016;116(10):Article 100501.
31. Farhi E, Goldstone J, Gutmann S. A quantum algorithm for the Hamiltonian NAND tree. *Theory Comput.* 2007;4: 169–190.
32. Zhang Y. A new type of classical logic circuit with exponential speedup. *Adv Intell Syst.* 2023;5(3):2200232.
33. Pan N, Chen T, Sun H, Zhang X. Electric-circuit realization of fast quantum search. *Research.* 2021;2021:Article 9793071.

34. Liu S, Gao W, Zhang Q, Ma S, Zhang L, Liu C, Xiang YJ, Cui TJ, Zhang S. Topologically protected edge state in two-dimensional Su–Schrieffer–Heeger circuit. *Research*. 2019;2019:8609875.
35. Zhang H. Electric-circuit simulation of quantum fast hitting with exponential speedup. *Adv Quantum Technol*. 2022;5(4):2100143.
36. Luo K, Yu R, Weng H. Topological nodal states in circuit lattice. *Research*. 2018;2018:6793752.
37. Tong X. Quantum combinational logics and their realizations with circuits. *Adv Quantum Technol*. 2024;7(1):2300251.
38. Jensen PWK. Molecular realization of a quantum NAND tree. *Quantum Sci Technol*. 2019;4(1):Article 015013.
39. Wang Y. Integrated quantum-walk structure and NAND tree on a photonic chip. *Phys Rev Lett*. 2020;125(16):Article 160502.
40. Cao W, Wang C, Chen W, Hu S, Wang H, Yang L, Zhang X. Fully integrated parity–time-symmetric electronics. *Nat Nanotechnol*. 2022;17(3):262–268.
41. Deng W. Ultrasensitive integrated circuit sensors based on high-order non-Hermitian topological physics. *Sci Adv*. 2024;10(38):eadp6905.