Quantum Information Processing (2026) 25:17
https://doi.org/10.1007/s11128-025-05022-y

QGoL: Quantum Game of Life

Daniel Escanez-Exposito[1] · Jorge Garcia-Diaz[1] · Daniel del Castillo[1] · Pino Caballero-Gil[1] · Eduardo Sáenz-de-Cabezón[2]

Received: 4 September 2025 / Accepted: 6 December 2025

Abstract
This paper proposes a novel probabilistic version of the Game of Life and a quantum model for it. Based on the study and formalisation of the traditional deterministic case, the quantum paradigm is presented as a good candidate for modelling probabilistic behaviour, providing a convenient description of the problem and efficient calculation of the necessary probabilities in an intrinsic way. To illustrate the advantages of quantum computing for this calculation, two implementations are presented: a classical version developed in Rust and a quantum version built using IBM's Qiskit quantum circuit software. The particularities of this model are presented, its applications are discussed, and optimisations are suggested for the general case, where different neighbourhood sizes and geometries or an increased number of dimensions can be considered. For the particular case of the traditional Game of Life and for all these extended cases, competitive complexity is demonstrated through the quantum paradigm, with respect to the exhaustive classical calculation of probabilities or approximate methods. This allows cases with many dimensions or complex conditions to be executed efficiently.

Keywords Quantum computing · Game of Life · Hamming weight quantum counter

✉ Daniel Escanez-Exposito
jescanez@ull.edu.es

Jorge Garcia-Diaz
jgarcidi@ull.edu.es

Daniel del Castillo
daniel.del.19@ull.edu.es

Pino Caballero-Gil
pcaballe@ull.edu.es

Eduardo Sáenz-de-Cabezón
eduardo.saenz-de-cabezon@unirioja.es

[1] University of La Laguna, La Laguna 38200, Tenerife, Spain

[2] University of La Rioja, Logroño 26006, La Rioja, Spain

Published online: 12 January 2026

1 Introduction

The Game of Life (GoL) [1, 2] is a zero-player game, since its evolution is only determined by its initial state, which is defined as a 2-dimensional table of cells, each of which may be alive or dead. To generate the next state of the game, it is necessary to compute the number of live neighbours for each cell (using the Moore neighbourhood: horizontally, vertically, or diagonally adjacent [3]). A dead cell is reborn if exactly 3 of its neighbours are alive (B3, to denote the Birth size set) and a live cell survives in the next generation if it has 2 or 3 live neighbours (S23, to denote the Survival size set). In any other case, the cell will be dead in the next generation. With these simple rules (B3/S23), this system forms a Turing complete cellular automaton that can simulate any other Turing machine or a universal constructor [4]. Patterns that occur spontaneously are singular, such as: still lives, which do not change between generations; oscillators, which are enclosed in a finite loop of states; and spaceships, which behave like the previous ones but include a displacement across the board. The use of cellular automata has, for instance, interesting relationships with diffusion models (concretely modelling the Game of Life) [5], cryptography [6] and artificial intelligence [7].

After its initial introduction, many variants of the game have appeared, exploring alternatives to its rules, spatial geometry, and dimensions. HighLife (B36/S23) extends the original rules by adding a birth condition for cells with six neighbours, enabling self-replication phenomena [8], while Seeds (B2/S0) and Life Without Death (B3/S0), respectively, introduce explosive, transient dynamics and irreversible, ever-growing patterns by eliminating survival in distinct ways [9, 10]. Larger than Life generalises the automaton by enlarging the interaction neighbourhood, producing macroscopic, life-like structures [11], whereas Generations incorporates multiple cell states to capture transient "life histories" that yield complex, multi-phase evolutions [12]. Day & Night (B3678/S34678) is distinguished by its self-complementary symmetry, allowing both dense and sparse configurations to be stable [13]. Furthermore, extending GoL to non-square lattices, such as hexagonal, triangular, and quasiperiodic tilings, demonstrates that complex behaviour persists across diverse geometries despite differences in local connectivity [14, 15]. The three-dimensional Game of Life adapts these concepts into a volumetric space, requiring modified thresholds for stability [16]. Besides, probabilistic variants integrate stochastic updates, thereby connecting cellular automata with statistical mechanics and phase transition theory [17], fields that use physical simulations far removed from traditional GoL. Another probabilistic approach advocates applying constant probability filters to a deterministic live and dead cell map, simplifying the methods and reserving non-determinism only for the transition between classical board states [18]. Additionally, continuous generalisations such as SmoothLife and Lenia replace discrete binary states and grid-based updates with continuous fields and convolution operations, yielding fluid, life-like patterns that bridge digital simulation and natural phenomena [19, 20]. Finally, previous quantum versions of the Game of Life reinterpret cell states as quantum superpositions to achieve reversible, entangled dynamics, in contrast to the classical irreversible rules [21, 22]. While the proposed approach to the quantum Game of Life offers the exploration of complex quantum dynamics using a time-dependent Hamiltonian, it remains

constrained by the limitations inherent in its simulation methodology. Game theory problems and other quantum games have also been modelled previously from the perspective of quantum circuits [23, 24].

The quantum circuit-based approach proposed in this paper offers a more comprehensive and computationally robust framework for studying the evolution and stability of probabilistic cellular automata. Note that quantum cellular automata are not simulated; instead, quantum amplitudes are employed to accelerate the evaluation of a Poisson-binomial distribution. Expressing a probabilistic version of the Game of Life as a quantum circuit is a novel aspect within the scientific literature. Furthermore, an original Hamming weight quantum counter design is also introduced and used in the proposed circuit. The approach presented in the article aims to provide a description of the traditional model within a quantum framework, specifically considering probabilities restricted to values of either 0 or 1. Due to the linearity of the model, this formulation naturally extends to behave consistently for all intermediate probability values. Additionally, the proposed algorithm enables efficient simulation of the probabilistic model through quantum Monte Carlo methods, achieving the same complexity with respect to classical approximation methods. Thanks to this, it is possible to efficiently simulate some of the aforementioned cases in their probabilistic versions and with a higher number of dimensions. Notably, two implementations, one using Qiskit [25] and another in Rust, have been developed to implicitly check the proposed design, facilitating various performance comparisons and validating the robustness of the approach.

This paper is organised as follows: Section 2 introduces the proposed probabilistic approach to the Game of Life. Section 3 describes the implementation developed using quantum circuits. Section 4 clarifies the details of the circuit design, while also justifying the optimisations made. Section 5 exhibits an experimental study, highlighting how the original rules (B3/S23) behave in the probabilistic case, as well as rule changes that achieve interesting behaviours. Section 6 seeks to enhance the value of the proposal and discusses its possible limitations. Finally, Sect. 7 concludes the document and suggests future works.

2 Probabilistic Game of Life formalisation

In the traditional Game of Life, the function δ for determining whether a given cell c is alive or not in the next generation is given by Eq. (1), depending on its neighbour cells N_c set. The Iverson bracket [ ] returns the veracity of a given predicate [26, 27], and the symbol # is used to denote the cardinality of a set. Also $A_c \subseteq N_c$ is the subset of neighbours that are alive, and the main rule set $\Gamma = \{\Gamma_{\text{births}}, \Gamma_{\text{survivors}}\}$ is formed by the two sets of cardinalities of A_c that define births ($\Gamma_{\text{births}} = \{3\}$) and survivors ($\Gamma_{\text{survivors}} = \{2, 3\}$).

$$\delta(c) = \begin{cases} [\#A_c \in \Gamma_{\text{births}}], & \text{if } c \text{ is dead} \\ [\#A_c \in \Gamma_{\text{survivors}}], & \text{otherwise} \end{cases} \tag{1}$$

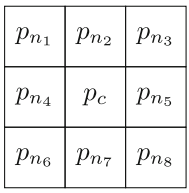

Fig. 1 Probability of life grid for cell c and its neighbours in traditional GoL geometry

In this way, the next state for a cell c is alive in the next generation:

- if c is currently dead and the number of live neighbours is contained in Γ_{births}, or
- if c is currently alive and the number of live neighbours is contained in $\Gamma_{\text{survivors}}$.

A probabilistic version of the Game of Life is proposed below, expressed in terms similar to those described above. Assuming that a cell c could have a given probability p_c of being alive (and hence a probability $1 - p_c$ of being dead), the probability of being alive in the next generation is determined by the cell c itself and the current probabilities of its neighbours being alive $P = \{p_{n_i} : \forall n_i \in N_c\}$ as shown in Fig. 1.

Assuming c is completely dead ($p_c = 0$), as in the traditional case, p_c will be the probability that Γ_{births} contains the number of live neighbours of c. However, in this case, it is not as simple as calculating $\#A_c$ for Eq. (1), because A_c is not defined in this probabilistic context. Only the probabilities included in P are available. It is not possible to determine the exact number of live neighbours, but it is possible to calculate the probability that a given number k of live neighbours exist, based on the probabilities defined in P. To explain the behaviour of the corresponding random variable, analogous to $\#A_c$ and denoted by Y, Eq. (2) is established. This equation is defined by the sum of the product of the probabilities for all possible subsets of live cells of a given size k. This follows a Poisson-binomial distribution [28], so the expression shows the sums of the product of the probabilities of being alive of all k-sized subsets of N_c times the product of all other cells' probabilities of being dead not included in this subset. Note that the number of summands is given by the number of the different subsets K of N_c with size k, that is $\#\{K \subseteq N_c : \#K = k\} = \binom{\mu}{k}$, where $\mu = \#N_c$.

$$\Pr(Y = k) = \sum_{\substack{K \subseteq N_c \\ \#K = k}} \left(\prod_{n_i \in N_c \setminus K} (1 - p_{n_i}) \right) \left(\prod_{n_j \in K} p_{n_j} \right) \tag{2}$$

On the other hand, when c is absolutely alive ($p_c = 1$), the same applies to the set $\Gamma_{\text{survivors}}$. Weighing these two possibilities correctly with the probabilities that these events occur ($1 - p_c$ and p_c, correspondingly), the traditional function δ described in Eq. (1) could be adjusted to obtain the probabilistic function ϕ shown in Eq. (3). Note that if $\forall c,\ p_c \in \{0, 1\}$, then ϕ behaves exactly as in the traditional case.

$$\begin{aligned}\phi(c) = (1 - p_c) \sum_{b \in \Gamma_{\text{births}}} \Pr(Y = b) \\ + p_c \sum_{s \in \Gamma_{\text{survivors}}} \Pr(Y = s)\end{aligned} \tag{3}$$

3 Overview of the quantum circuit implementation

Algorithm 1 models the pseudocode that represents the function δ in Eq. (1) for the traditional Game of Life. First, the number of live neighbours $\#A_c$ is computed and stored in a, for which the neighbours N_c are traversed exhaustively and a counter is incremented when neighbour n_i is alive. Once that counter has been correctly set, the rules are applied, checking if c is alive and if the value in the counter exists within one of the sets defined by the rules Γ.

Algorithm 1 Next generation in traditional Game of Life

Require: c
Ensure: $\delta(c)$
1: $a \leftarrow \sum_{n_i \in N_c} \text{isAlive}(n_i)$ ▷ $O(\mu)$
2: **if** isAlive(c) **then**
3: **return** isContainedIn(a, $\Gamma_{\text{survivors}}$) ▷ $O(\#\Gamma_{\text{survivors}})$
4: **else**
5: **return** isContainedIn(a, Γ_{births}) ▷ $O(\#\Gamma_{\text{births}})$
6: **end if**

The proposal below implements the probabilistic version described in Sect. 2, based on the traditional description and the principles of quantum computing. In this way, defining the traditional Game of Life within the quantum paradigm, the probabilistic version will be obtained by linearity, and the probabilities exposed in Eq. (3) will be fulfilled.

3.1 Initialisation

For the implementation of a quantum cell, a qubit will be used. This mathematical object is the quantum version of the bit and can be defined according to $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, where $\sum_{i \in \{0,1\}} ||\alpha_i||^2 = 1$ and $\alpha_i \in \mathbb{C}$, $\forall i \in \{0, 1\}$. For probabilistic purposes, this description is quite convenient, since $||\alpha_i||^2$ represents the probability of measuring the qubit in the state $|i\rangle$. Therefore, it is possible to encode within a qubit the probability p_c that cell c is alive, as shown in Eq. (4).

$$|\psi_c\rangle = \sqrt{1 - p_c}\,|0\rangle + \sqrt{p_c}\,|1\rangle \tag{4}$$

As has been shown, the quantum model allows one to express the proposed probabilistic behaviour of the Game of Life in its most basic unit in an inherently way. From the quantum operations or quantum gates, which are essentially unitary matrices that

act over qubits, the rules will be modelled. First, the gate that prepares the qubits in the state exposed in Eq. (4) will be defined. For this, it must be taken into account that the qubits are initially, at the beginning of the computation, in the $|0\rangle$ state. Therefore, a unitary operation $U(p_c)$ that performs the following application is needed: $U(p_c)\,|0\rangle = \sqrt{1-p_c}\,|0\rangle + \sqrt{p_c}\,|1\rangle$. A quantum gate $U(p_c)$ that achieves this goal is given by Eq. (5), which is an unitary matrix. Note that $U(p_c)$ is a real and symmetric matrix, therefore it is also hermitian, i.e. $U(p_c)^\dagger = (U(p_c)^*)^T = U(p_c)$. Finally, the unitarity of the matrix is proven by checking $U(p_c)U(p_c)^\dagger = U(p_c)^2 = I$.

$$U(p_c) = \begin{pmatrix} \sqrt{1-p_c} & \sqrt{p_c} \\ \sqrt{p_c} & -\sqrt{1-p_c} \end{pmatrix} \tag{5}$$

The quantum circuit proposed in this section aims to compute the function ϕ. With these gates, it is possible to start defining the circuit with the initialisation of the system, loading the probabilities of c and all its neighbours N_c. Following the algebraic notation of quantum circuits [29], the initialisation of this algorithm can be represented as $\prod_{x \in N_c \cup \{c\}} [U(p_x)]_x$.

The quantum complexity of the proposed circuit is evaluated in terms of Quantum Cost (QC) [30] and circuit depth [31]. The QC is defined as the total number of elementary quantum operations required to implement the circuit. In this case, all single-qubit gates and the $CNOT$ gate are considered elementary operations. The depth corresponds to the maximum number of sequential gate layers along any path from input to output, assuming gates on disjoint qubits can be executed in parallel. The QC for this part of the circuit is 1 for each initialisation gate, i.e. $\mu + 1$ (recall that $\mu = \#N_c$, the number of neighbours of each cell), and its depth is 1 (constant).

3.2 Hamming weight quantum counter

Once the probabilities defining Eq. (5) have been established on the qubits corresponding to the current cell and its neighbours, it will be easy to count the number of live neighbours using the novel Hamming weight quantum counter (HWQC) proposed in this paper, developed through a controlled increment gate. Recall that the Hamming weight of a string is the number of nonzero symbols, which in the present case are the bits set to one. For this purpose, a qubit register $a = (a_0, \ldots, a_{m-1})$, with $m \leq \lceil log_2(\mu) + 1 \rceil$ (details of the selection of the correct value for this parameter will be addressed in Sect. 4.1), will be allocated to store the binary expression of the number of live neighbours (as in line 1 of Algorithm 1). For each neighbouring cell, an increment gate [32] (which adds one unit to the register) is applied on the live neighbour counter register, and this is controlled by the qubit relative to the corresponding neighbouring cell. Controlled gates are executed on the target qubits only for those possibilities where the control qubits are set to the state $|1\rangle$. In this way, the neighbouring cell that is in superposition will activate the increaser for the $|1\rangle$ possibility and will not activate it for the $|0\rangle$ possibility. In algebraic notation, the Hamming weight quantum counter is implemented as the application of the controlled-INC gate for each neighbour of c and can be expressed as $\prod_{n_i \in N_c} INC_a^{n_i}$. Therefore, when all of

these gates are applied, the resulting quantum state of the registers n and a, named $|\Psi_{\text{HWQC}}\rangle$, is shown in Eq. (6). Note that the iterator decreases with each factor of the inner tensor product and the use of iverson bracket. The detailed explanation for how to achieve this state by using the HWQC is explained in Sect. 4.2 and Appendix A.

$$|\Psi_{\text{HWQC}}\rangle = \sum_{k\in\{0,\ldots,\mu\}} \left[\sqrt{\Pr(Y=k)} \left(\sum_{\substack{K\subseteq N_c \\ \#K=k}} \bigotimes_{i=\mu}^{1} |[n_i \in K]\rangle \right) \otimes |k\rangle \right] \tag{6}$$

The above quantum state forms a correctly weighted superposition of the neighbourhood N_c and the live neighbour counter a, keeping each possibility of this counter with the coefficient that ensures to measure it with the probability described in Eq. (2). Controlled-INC gate has the cost of adding one more control to each of the gates that form it. Analysing the quantum cost of this circuit, an i-controlled NOT gate is required for each $i \in \{1,\ldots,m\}$, which has a linear QC with respect to the number of controls i, using a linear number of auxiliary qubits (also with respect i) [33, 34]. Therefore, the QC (and the depth, since it is not possible to apply in parallel these gates) of a controlled-INC gate for m qubits is given by a function in $O(log^2(\mu))$. As μ incremental gates are needed to implement the HWQC, the complexity function of this component is in $O(\mu \cdot log^2(\mu))$.

3.3 Rules application

Finally, by applying the rules of the δ function (see Eq. (1) and lines 2–6 of Algorithm 1) to the circuit, the counter register value is used to determine the state of the *output* cell, i.e. that of the next-generation cell (*next_gen*). To implement the gate behaviour that models the use of the rule set Γ, the following reasoning will be followed: one set of gates models the birth of the *output* cell for those cases where appropriate, and the other set of gates implements the cell survival if applicable. Therefore, the set of gates is described by Eq. (7).

- The first set of gates will consist of an X gate for each $b \in \Gamma_{\text{births}}$ with target in *output*. This is a controlled operation, which activates when c is dead ($c = 0$) and a is equal to one of the numbers of live neighbours required for the cell birth ($a = b$). It can be expressed as $\prod_{b\in\Gamma_{\text{births}}} X_{output}^{c=0,a=b}$.
- The second set of gates, the ones controlling the survival of the cell, operate the same way, but using $\Gamma_{\text{survivors}}$ and $c = 1$. It can be expressed as $\prod_{s\in\Gamma_{\text{survivors}}} X_{output}^{c=1,a=s}$.

$$R(\Gamma)_{a,c,output} = \prod_{b\in\Gamma_{\text{births}}} \left(X_{output}^{c=0,a=b} \right) \prod_{s\in\Gamma_{\text{survivors}}} \left(X_{output}^{c=1,a=s} \right) \tag{7}$$

It should be noted that this set of gates can be optimised as illustrated in Sect. 4.3. Even so, estimating the QC and depth of this part of the circuit in its worst case in a

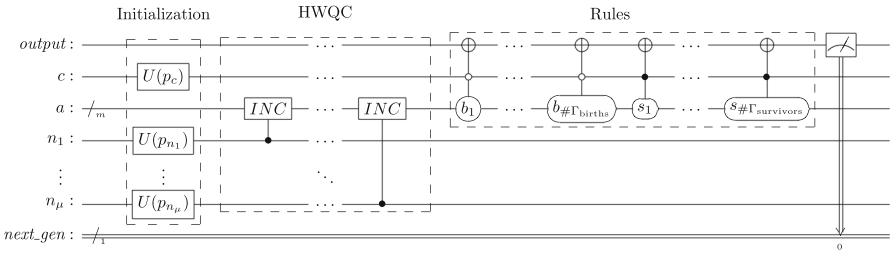

Fig. 2 Full quantum circuit implementation of general QGoL

similar way to the previous component, given by $\#(\Gamma_{\text{births}} \cup \Gamma_{\text{survivors}}) \cdot m^2$, which is bounded above by a function in $O(\mu \cdot log^2(\mu))$.

3.4 Overall implementation

After each component of the circuit has been introduced, the full expression that represents a generic Game of Life (for any set of rules, number of neighbours, number of considered dimensions or geometry), named $QGoL$, can be found in Eq. (8), where $M_{x,y}$ represents the measurement instruction over the qubit x, saving its result in the classical bit y. The graphical representation of this circuit for the traditional Game of Life instance can be visualised in Fig. 2.

$$QGoL = \overbrace{\prod_{x \in N_c \cup \{c\}} [U(p_x)]_x}^{\text{Initialization}} \underbrace{\prod_{n_i \in N_c} INC_a^{n_i}}_{\text{HWQC}} \overbrace{R(\Gamma)_{a,c,output}}^{\text{Rules}} M_{output,next\_gen} \tag{8}$$

It should be noted that the classical computational complexity of evaluating the exact output probability, in the worst-case scenario (where $\#(\Gamma_{\text{births}} \cup \Gamma_{\text{survivors}}) = \mu + 1$ and $\forall G \in \Gamma : \sum_{g \in G} \binom{\mu}{g} \approx 2^{\mu-1}$) requires computing Eq. (2) by brute force $\forall k \in \{0, \ldots, \mu\}$ when using a naive implementation. For each k, the evaluation involves summing over $\binom{\mu}{k}$ terms, each consisting of a product of μ factors. Since $\sum_{k=0}^{\mu} \binom{\mu}{k} = 2^{\mu}$, the overall complexity is in $O(2^{\mu} \cdot \mu)$. However, better algorithms exist to compute the Poisson-binomial distribution, if the exact values are not needed, by using FFT-based convolution [35]. Some approximate methods achieve to reduce the complexity to $O(\mu^2)$ in practice [36], even to $O(\mu \cdot log^2(\mu))$ in some theoretical proposals [37], which matches the proposed quantum version.

For every cell across all generations of a given board, the proposed quantum circuit must be executed a constant number $M = \frac{1}{\varepsilon^2}$ of times (also known as the number of shots, which is solely dependent on ε, the desired maximum tolerated standard error), to perform a quantum Monte Carlo method [38]. It uses $\mu + m + 2$ qubits (and a linear amount, with respect to m, of auxiliary qubits, in the event that the quantum processor does not have the proposed gates), so the total number of qubits is in $O(\mu)$, and its QC and depth grow as $O(\mu \cdot log^2(\mu))$. Note that the number of qubits and the

Table 1 QGoL circuit's Quantum Cost and depth

	QC	Depth
Initialisation	$O(\mu)$	$O(1)$
HWQC	$O(\mu \cdot log^2(\mu))$	$O(\mu \cdot log^2(\mu))$
Rules	$O(\mu \cdot log^2(\mu))$	$O(\mu \cdot log^2(\mu))$
Total quantum bound	$O(\mu \cdot log^2(\mu))$	$O(\mu \cdot log^2(\mu))$
Classical brute force complexity	$O(2^\mu \cdot \mu)$	
Classical approximate complexity	$O(\mu \cdot log^2(\mu))$	

complexities shown do not depend on the size of the board. Table 1 shows a summary of the upper bound estimates QC and depth.

4 Characterisation of implementation details

Some mathematical details on the construction of certain parts of the proposed circuit, as well as the possible optimisations that can be made, have been left out of the previous section in order to clarify the main development. These details are displayed and discussed below.

4.1 Optimisation of the live neighbours counter size *m*

The size of the counter of live neighbours m of any cell c, which corresponds to the number of target qubits in the HWQC, is bounded at the top by $\lceil log_2(\mu) + 1 \rceil$, where $\mu = \#N_c$. This is the number of bits that are necessary to represent any Hamming weight of a string of size μ. However, m can take a value much smaller than that upper bound, which is especially interesting for the implementation of the HWQC circuit. This section describes the process to obtain the optimal value of m. For any given GoL rules $\Gamma = \{\Gamma_{\text{births}}, \Gamma_{\text{survivors}}\}$, there always exists a minimum value of m for which all larger sizes of a (the HWQC target register) lead to the same GoL output.

For instance, consider Table 2, which describes the traditional Conway's GoL. In this representation, g is a possible size of the live neighbour set, and the symbol χ is used for "not applicable" or "don't-care" cases. Each possible value of m splits the rows into two parts, which will be referred to as `start` and `end`. This table shows that 3 is the smallest value of m after which, by periodically repeating the `start` of the last two rows of the table, it is possible to generate the `end` of those two rows (ignoring χ).

Another example of a GoL variant, known as Fredkin or Replicator 2, is defined by B1357/S02468. This is one of the most elegant and symmetric rules in the "Life-like" universe, because every pattern is replicated indefinitely, which enables the study of replicators in cellular automata, simulations of reversible systems and minimalist artificial life, and experimental cryptography using chaotic or pseudo-random patterns [39]. Since births occur at odd numbers and survivals at even numbers, this is a very

Table 2 Graphical optimisation for m value in traditional GoL rules

m	0	1	2	**3**					4							
g	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$[g \in \Gamma_{\text{births}}]$	0	0	0	1	0	0	0	0	0	✗	✗	✗	✗	✗	✗	✗
$[g \in \Gamma_{\text{survivors}}]$	0	0	1	1	0	0	0	0	0	✗	✗	✗	✗	✗	✗	✗
	start								end							

Table 3 Graphical optimisation for m value in Replicator 2 GoL

m	0	**1**	2		3			4								
g	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$[g \in \Gamma_{\text{births}}]$	0	1	0	1	0	1	0	1	0	✗	✗	✗	✗	✗	✗	✗
$[g \in \Gamma_{\text{survivors}}]$	1	0	1	0	1	0	1	0	1	✗	✗	✗	✗	✗	✗	✗
	start		end													

Table 4 Graphical optimisation for m value in a GoL with 12 neighbours

m	0	1	**2**		3			4								
g	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$[g \in \Gamma_{\text{births}}]$	0	1	0	1	0	1	0	1	0	1	0	1	0	✗	✗	✗
$[g \in \Gamma_{\text{survivors}}]$	1	0	1	1	1	0	1	1	1	0	1	1	1	✗	✗	✗
	start				end											

intuitive example of what this section aims to show. As exposed in Table 3, the value m=1 represents the minimum for which the `ends` of the last two rows can be generated by periodically repeating their `starts`, disregarding ✗.

Finally, it is worth noting the improvement achieved by this technique in cases with a larger number of neighbours. For this purpose, an instance with 12 neighbours is introduced. In this case, the 4 orthogonal cells at distance 2 (in the directions north, south, east and west) are added to the 8 standard neighbours. In Table 4, a GoL variant that uses this neighbourhood and the transition rules B1,3,5,7,9,11/S0,2,3,4,6,7,8,10,11,12 is considered. It can be observed that, for m=2, periodically repeating the `starts` suffices to reconstruct the `ends`, ignoring ✗.

The observations made for the previous examples can be confirmed for any GoL variant. Therefore, formally, the optimal value of the parameter m can be calculated using Eq. (9). This minimisation function returns the smallest m that satisfies that for each row (with its corresponding $G \in \Gamma$ set) and, for each row element g in the `start` sequence, all its congruent elements g' in the `end` sequence must belong to the set G if and only if g belongs to it. In this way, an optimisation method can be

defined to minimise the quantum register size for the live neighbour counter in the implementation of the proposed QGoL, which is applicable in any general case.

$$\begin{gathered}\min m:\ \forall G \in \Gamma, \forall g < 2^m, \forall g' \leq \mu : g' \equiv g \pmod{2^m}, \\ g \in G \iff g' \in G\end{gathered} \tag{9}$$

4.2 Using HWQC to count live neighbours

The use of HWQC for live neighbours counting starts with the circuit already initialised, so the neighbour qubits have the probabilities of initiation. In addition, there is a register of size m set to zero to store the value of the counter. For each initialised neighbour qubit, the controlled incrementer must be applied. The expression that represents this situation is described by Eq. (10). Note the use of $(j)_{2,i}$ to express the i-th bit of the decimal number j expressed in binary notation.

$$|\Psi_1\rangle = \left[\sum_{j=0}^{2^\mu-1}\left(\prod_{i=1}^{\mu}\sqrt{(1-p_{n_i})^{1-(j)_{2,i}}}\sqrt{{p_{n_i}}^{(j)_{2,i}}}\right)|j\rangle\right] \otimes |0\rangle^{\otimes m} \prod_{n_t \in N_c} INC_a^{n_t} \tag{10}$$

The application of this cascade of controlled-INC gates trivially leads to the state that entangles each possibility of the neighbour register with its corresponding Hamming weight (see Appendix A), which previously was the register consisting of m zeros. This is the case since the increase by one unit for each qubit in state one for each possibility achieves this effect. Therefore, the above expression is equivalent to Eq. (11).

$$|\Psi_2\rangle = \sum_{j=0}^{2^\mu-1}\left[\left(\prod_{i=1}^{\mu}\sqrt{(1-p_{n_i})^{1-(j)_{2,i}}}\sqrt{{p_{n_i}}^{(j)_{2,i}}}\right)|j\rangle \otimes |W((j)_2)\rangle\right] \tag{11}$$

In this way, it is possible to draw a common factor k for the summands where they share the same Hamming weight value, grouping the above expression into groups that share this characteristic, as shown in Eq. (12).

$$|\Psi_3\rangle = \sum_{k\in\{0,\ldots,\mu\}}\left[\left(\sum_{\substack{K\subseteq N_c \\ \#K=k}}\left(\prod_{n_i\in N_c\setminus K}\sqrt{(1-p_{n_i})}\right)\left(\prod_{n_i\in K}\sqrt{p_{n_i}}\right)\bigotimes_{l=\mu}^{1}|[n_l \in K]\rangle\right)\otimes|k\rangle\right] \tag{12}$$

After doing this, the square root of $\Pr(Y = k)$, which is defined in Eq. (2), appears as the coefficient of each probability associated with each case. This allows a final change to be made to the expression to make this fact even more explicit, leaving the expression $|\Psi_{\text{HWQC}}\rangle$ (see Eq. (6)).

4.3 Optimisation of rule gates

To model the rules, the δ' function described in Eq. (13) is used to perform the same work as δ in Eq. (1). In this expression, the c binary variable specifies if the cell is alive, and the a binary register of size m represents the live neighbours counter. These values parametrise the minterms or minimum sums of a rule set $G \in \Gamma$ denoted by $\Sigma_a(G)$ [40].

$$\begin{aligned} \delta'(c,a) &= \bar{c} \cdot \Sigma_a(\Gamma_{\text{births}}) + c \cdot \Sigma_a(\Gamma_{\text{survivors}}), \\ \text{where } \Sigma_a(G) &= \sum_{g \in G} \prod_{j=1}^{m} \overline{a_j \oplus (g)_{2,j}} \end{aligned} \tag{13}$$

For instance, the resulting expression of the traditional GoL rules is shown in Eq. (14).

$$\begin{aligned} \bar{c} \cdot \Sigma_a(\{3\}) + c \cdot \Sigma_a(\{2,3\}) = \bar{c} \cdot \bar{a_3} \cdot \bar{a_2} \cdot a_1 \cdot a_0 \\ + c \cdot \bar{a_3} \cdot \bar{a_2} \cdot a_1 \cdot \bar{a_0} + c \cdot \bar{a_3} \cdot \bar{a_2} \cdot a_1 \cdot a_0 \end{aligned} \tag{14}$$

Each of the minterms in this Boolean expression corresponds to the controls of a multicontrolled not gate, where the target is the output qubit, justifying Eq. (7). From this, it is possible to intuit some properties about the quantum gates involved in the modelling of the rules. For example, it follows that they commute with each other (just as the addends commute). Moreover, any optimisation made in this expression represents a more efficient quantum circuit for this purpose. These optimisations can be achieved through the rules defined by the algebraic notation for quantum circuits [29], which would allow a quantum gate transpiler to perform these improvements without regard to the underlying optimisation of the Boolean expression. Some examples of this procedure can be found in the Appendix B.

Finally, if $\exists\, G \in \Gamma : \#G > \frac{\mu}{2}$, then could be convenient to apply the property expressed in Eq. (15), with $y = 0$ if $G = \Gamma_{\text{births}}$ or $y = 1$ if $G = \Gamma_{\text{survivors}}$, by using the relative complement of N_c with respect to G, in order to minimise the circuit depth.

$$\prod_{g \in G} X_{output}^{c=y, a=g} = X_{output}^{c=y} \prod_{g' \in N_c \setminus G} X_{output}^{c=y, a=g'} \tag{15}$$

5 Experimental study

Once various aspects of the circuit design have been justified, the complete circuit for the traditional GoL instance, optimising its rules using Eq. (B.5), is shown in Fig. 3. As can be seen, it requires a small amount of qubits for the NISQ era [41], 13 qubits, not counting the linear amount of auxiliary qubits needed for the implementation of the controlled incrementers and rule gates. More examples of complete optimisation design can be found in Appendix C.

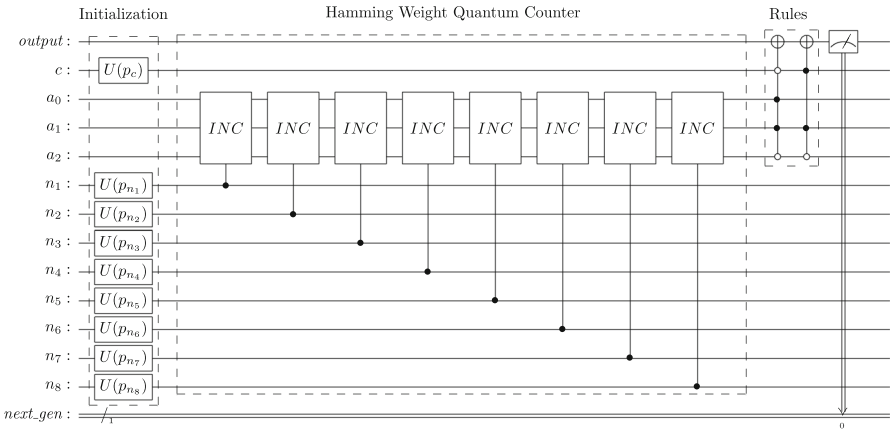

Fig. 3 Full quantum circuit for the probabilistic version of the traditional GoL

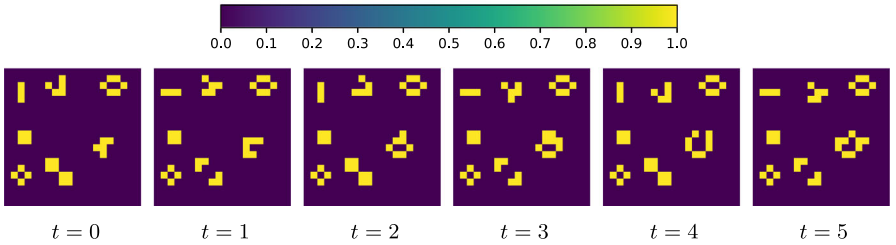

Fig. 4 Colour legend and B3/S23 with traditional behaviour example in generation t

Therefore, for this instance, it is efficient to simulate this circuit in a classical way, as well as to calculate the probabilities exhaustively. To justify this empirically, the quantum version described in the paper using Qiskit in Python and a Rust implementation of the computation of the probabilities in the classical way have been carried out [42]. The necessary documentation for the execution of the programs is contained in the repository itself. Thanks to these implementations, which give us exactly the same experimental results, the following graphs have been obtained.

First, an execution was carried out by experimentally verifying that the results of the traditional rules for the case of $\forall p_c \in \{0, 1\}$, were identical to those of the original Conway's GoL. The expected result can be seen in Fig. 4, where the cells that have more probability of being dead will have a darker purple colour, and those that are more likely to be alive will have a lighter yellow colour. For this example, some well-known structures have been placed on a 20x20 board with periodic boundary conditions and their correct behaviour has been verified.

The second example in Fig. 5 is inspired by the previous one, since it takes the same structures, changing those probabilities that were at 1 to 0.9 and applying the same process. In this case, patterns formed by structures that tend to fade or reach a stable position can be observed, as in the case of the light yellow square.

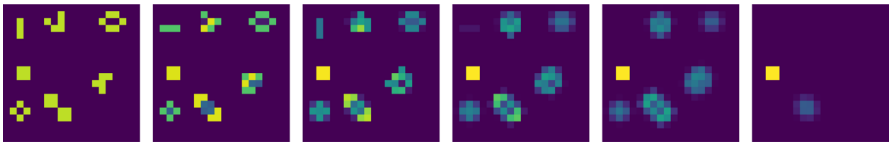

Fig. 5 B3/S23 with probabilistic behaviour example in generation t

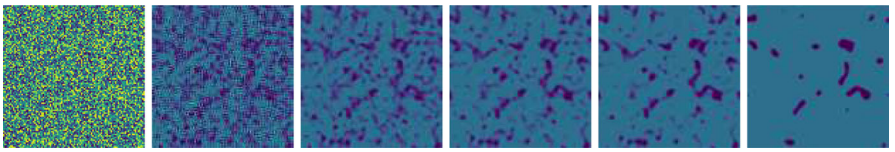

Fig. 6 B3/S23 random probabilities example in generation t

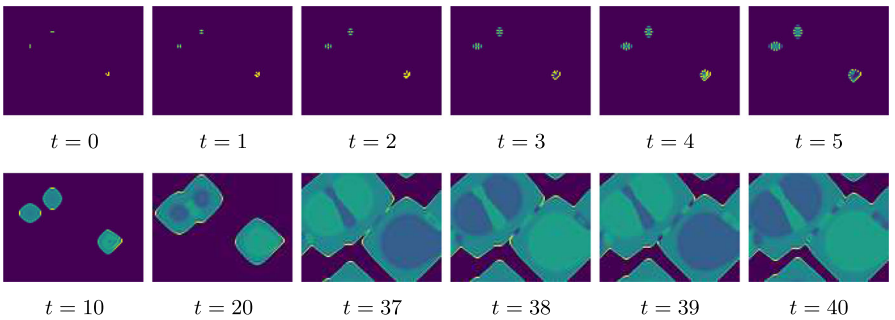

Fig. 7 B234/S3 example in generation t

A more ambitious board of 100x100 cells with random probabilities was formed to see its evolution. In this case, illustrated in Fig. 6, it can be seen how Turing pattern-like structures are generated by cells with a probability near zero of being alive, which tend to gradually fade away in a sea of cells with a value ∼0.37.

Tests were performed by changing the rule sets, against a board of cells with zero probability of being alive, and two structures of bar and glider with values 0.8 and 0.9, respectively. Formations of rhombus structures can be observed with rules B234/S3 in Fig. 7, rectangular with rules B123/S3 in Fig. 8, and octagonal using rules B345678/S0123456 in Fig. 9. In the boards of correlated generations, it can be seen how the interior of the structures flickers between two different states (one clearly with more average probability of being alive than the other), while the edge with the shape grows indefinitely.

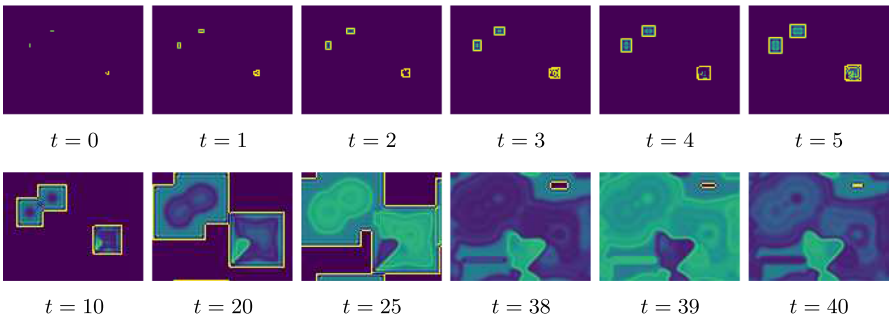

Fig. 8 B123/S3 example in generation t

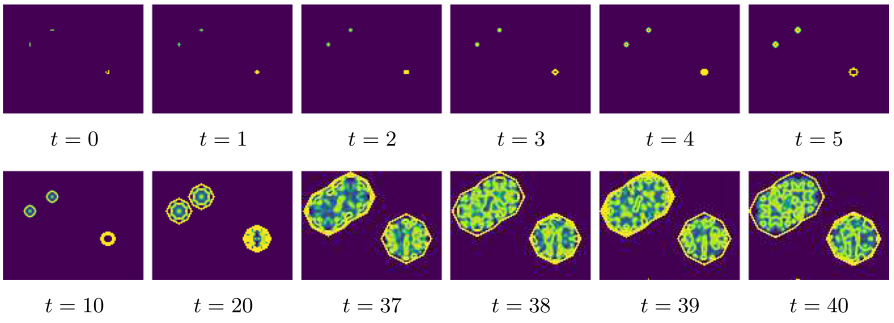

Fig. 9 B345678/S0123456 example in generation t

6 Discussion

The obtained results demonstrate the potential of quantum computation for the efficient simulation of probabilistic cellular automata, with particular emphasis on the Game of Life. By reformulating the classical model within a quantum framework, it becomes possible to evaluate the system's probabilistic evolution with significantly competitive computational complexity. This is especially relevant in scenarios involving higher-dimensional spaces, larger neighbourhoods, or more complex update conditions. The linear nature of the quantum formulation ensures that the model behaves consistently across the full range of probability values between 0 and 1. While the circuit is constructed to reflect binary transitions, the resulting behaviour remains valid for intermediate probabilities, broadening the scope of the simulation without requiring structural modifications to the circuit. Although the study has focused on a specific probabilistic extension of the Game of Life, the proposed framework is generalisable. The modular structure of the quantum circuit supports adaptation to alternative cellular automaton rules, topologies, or probabilistic update functions.

The behaviour of the proposed system may resemble that of a weighted filter applied over the lattice, where the aggregation of neighbour contributions can lead to a smoothing effect that inhibits the formation of classical patterns typically observed in the deterministic version of the Game of Life. This is a direct consequence of the uniform consideration of all neighbourhood contributions when calculating tran-

Table 5 Qiskit transpilation of m-controlled NOT

m	QC ($U3$)	QC ($CNOT$)	Depth
1	0	1	1
2	8	6	11
3	19	14	28
4	41	36	65
5	90	84	130
10	514	452	674
20	2508	2316	4030
30	3467	3608	5022
40	4595	4808	6666
50	5715	6008	8406
100	11315	12008	17106
200	22515	24008	34506
300	33715	36008	51906
400	44915	48008	69306
500	56115	60008	86706
1000	112115	120008	173706
2000	224115	240008	347706
3000	336115	360008	521706
4000	448115	480008	695706
5000	560115	600008	869706

sition probabilities. To address this, a possible improvement is to introduce models with neighbourhood position dependent weights. In the quantum implementation, this could be achieved by selectively increasing the influence of certain neighbours through repeated applications of the corresponding controlled-INC gates. Conceptually, this is equivalent to implementing a controlled-$INC^{\gamma n_j}$, where γ represents the number of repetitions that determines the relative weight of each neighbour n_j, allowing local dynamics to emerge.

Another issue that needs to be addressed in this section is the practical feasibility of one of the most significant contributions of this paper: the HWQC circuit. It has been treated from a theoretical point of view, with which its complexity in terms of QC and depth were found to be $O(\mu \cdot log^2(\mu))$. It was described that a controlled-INC gate was required for each neighbour (μ) and that each of these gates required a number of m-controlled $CNOT$ gates that grows as $log(\mu)$. Moreover, $log(\mu)$ was also the complexity of the function which describes the growing of the number gates required to build an m-controlled $CNOT$. It is this last point that may seem most implausible from a practical perspective. To verify this, a transpilation simulation was performed using the default version in Qiskit `2.2.3`, with the results compiled in Table 5, where linear growth with respect to the number of controls (and therefore logarithmic with respect to the number of qubits needed to represent the number of neighbours) can be verified.

Furthermore, the current proposal focuses on a quantum implementation approach to modelling probabilistic cellular automata, relying on classical measurement of quantum amplitudes to extract probabilities. A more purely quantum formulation, such as one that exploits the phase information of qubit states, remains an open avenue for future research. Such an approach would involve encoding probability amplitudes using the polar form of quantum states and would require careful design to maintain coherence and enable meaningful post-measurement interpretation. However, the limitations imposed by quantum measurement, which inherently collapses superpositions and limits access to full state information, represent a key challenge in this direction. Studying these limitations and finding ways to circumvent these partial measurements may allow for richer dynamical behaviours and new classes of cellular automata beyond the probabilistic regime.

7 Conclusion

This work presents a novel quantum framework for simulating a probabilistic version of the Game of Life, demonstrating the feasibility and computational advantage of modelling probabilistic cellular automata through quantum circuits. By reformulating the classical model in a quantum context, the approach enables efficient computation of transition probabilities. The proposed design is modular, scalable, and generalisable, laying the groundwork for the quantum simulation of a wide variety of cellular automata rules and configurations. The introduction of a Hamming weight quantum counter, along with dual implementations in Qiskit and Rust, supports both theoretical development and empirical validation.

Future work may explore position-weighted neighbourhoods to enable the emergence of patterns more similar to those found in the traditional GoL, as well as more intrinsically quantum formulations that exploit phase information and coherence for richer dynamical behaviours. Another line of future work may concern the study of well-known stable structures in the classical Game of Life. Studying their behaviour in the probabilistic setting raises several key questions, including the existence of stable structures that persist indefinitely even under homogeneous probabilities; the identification of structures that are stable only within certain probability ranges, as well as the characterisation of those ranges; and the analysis of the behaviour of asymptotic structures, particularly the mechanisms by which they converge to specific values. Many similar questions may be posed to study in more depth the behaviour of this probabilistic version of the Game of Life (GoL) using the proposed Quantum Game of Life (QGoL).

Appendix A HWQC Proof

Before proofing the operation of the HWQC circuit, some details of the INC gate are established. The internal implementation of the incrementer gate [32] can be represented as $INC_a = \prod_{i=m-1}^{0} X_{a_i}^{a_0:a_{i-1}}$, defining negative ranges ($i{:}j$, with $i > j$) as the empty set Ø. Note that the iterator decreases with each factor. The graphical

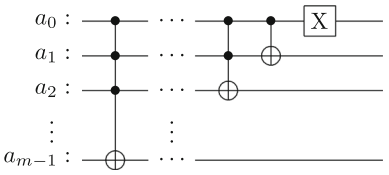

Fig. 10 Incrementer circuit for m qubits

representation of this circuit can be visualised in Fig. 10. Therefore, the behaviour of this gate can be expressed as $|x\rangle INC_a = |x+1 \mod 2^m\rangle$, where m is the size of the register a.

Lemma 1 *Let the neighbourhood N_c of size μ and quantum register a the counter initialised to $|0\rangle^{\otimes m}$. The invariant $I(i)$ is defined as follows:* "After stage i, the register a contains the correct value corresponding to the neighbourhood N_c of size i." *Then, $I(i)$ holds for all $i \in \{1, \ldots, \mu\}$.*

$$\Big(\sum_{j=0}^{2^i-1} \alpha_j |j\rangle\Big) \otimes |0\rangle^{\otimes m} \prod_{n_t \in N_c} INC_a^{n_t} = \sum_{j=0}^{2^i-1} (\alpha_j |j\rangle \otimes |W(j)_2\rangle) \tag{A.1}$$

Proof The proof proceeds by induction on i.

Base case ($i = 1$):

$$\begin{aligned}\Big(\alpha_0|0\rangle + \alpha_1|1\rangle\Big) \otimes |0\rangle^{\otimes m} INC_a^1 &= \alpha_0|0\rangle \otimes |0\rangle^{\otimes m} + \alpha_1|1\rangle \otimes |0\rangle^{\otimes m-1} \otimes |1\rangle \\ &= \alpha_0|0\rangle \otimes |W(0)\rangle + \alpha_1|1\rangle \otimes |W(1)\rangle\end{aligned} \tag{A.2}$$

Therefore, $I(1)$ holds.

Induction hypothesis: Assume that $I(k)$ holds, that is, after stage k the register a stores the correct value corresponding to the neighbourhood N_c of size k.

Inductive step: Consider stage $k+1$.

$$\begin{aligned}&\left(\sum_{j=0}^{2^{k+1}-1} \alpha_j |j\rangle\right) \otimes |0\rangle^{\otimes m} \prod_{n_t \in N_c \cup \{k+1\}} INC_a^{n_t} \\ &= \left(\sum_{j=0}^{2^{k+1}-1} \alpha_j |j\rangle\right) \otimes |0\rangle^{\otimes m} \left(\prod_{n_t \in N_c} INC_a^{n_t}\right) INC_a^{k+1} \\ &= \sum_{r=0}^{2^k-1} \Big(\gamma_r |r\rangle \otimes (\beta_0|0\rangle + \beta_1|1\rangle) \otimes |W(r)_2\rangle\Big) INC_a^{k+1} \\ &= \sum_{r=0}^{2^k-1} \Big((\gamma_r\beta_0|r0\rangle + \gamma_r\beta_1|r1\rangle) \otimes |W(r)_2\rangle\Big) INC_a^{k+1}\end{aligned}$$

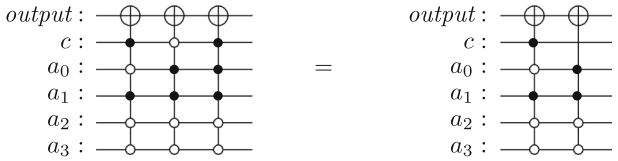

Fig. 11 Graphical representation of the rule optimisation in Eq. (B.4)

$$= \sum_{j=0}^{2^{k+1}-1} \alpha_j |j\rangle \otimes |W(j)_2\rangle \tag{A.3}$$

Conclusion: By induction, the invariant $I(i)$ holds for all i. □

Appendix B: Rule gates optimisation examples

An example of the procedure over the set of gates for the implementation of the traditional Conway's Game of Life rule set is shown in Eq. (B.4). Note that the notation $\bar{i}$ will be used when the control i must be zero to apply the gate [43]. Using graphical representation, the realised optimisation can be visualised in Fig. 11.

$$\begin{aligned}
\prod_{b\in\Gamma_{\text{births}}=\{3\}} \left(X_{output}^{c=0,a=b}\right) \prod_{s\in\Gamma_{\text{survivors}}=\{2,3\}} \left(X_{output}^{c=1,a=s}\right) &= X_{output}^{c=0,a=3} X_{output}^{c=1,a=2} X_{output}^{c=1,a=3} \\
&= X_{output}^{c=1,a=2} X_{output}^{c=0,a=3} X_{output}^{c=1,a=3} \\
&= X_{output}^{c,\bar{a_3},\bar{a_2},a_1,\bar{a_0}} X_{output}^{\bar{c},\bar{a_3},\bar{a_2},a_1,a_0} X_{output}^{c,\bar{a_3},\bar{a_2},a_1,a_0} \\
&= X_{output}^{c,\bar{a_3},\bar{a_2},a_1,\bar{a_0}} X_{output}^{\bar{a_3},\bar{a_2},a_1,a_0}
\end{aligned} \tag{B.4}$$

Another equivalent circuit, performing different algebraic operations, can be visualised in Eq. (B.5) and in Fig. 12.

$$\begin{aligned}
\prod_{b\in\Gamma_{\text{births}}=\{3\}} \left(X_{output}^{c=0,a=b}\right) \prod_{s\in\Gamma_{\text{survivors}}=\{2,3\}} \left(X_{output}^{c=1,a=s}\right) &= X_{output}^{c=0,a=3} X_{output}^{c=1,a=2} X_{output}^{c=1,a=3} \\
&= X_{output}^{\bar{c},\bar{a_3},\bar{a_2},a_1,a_0} X_{output}^{c,\bar{a_3},\bar{a_2},a_1,\bar{a_0}} X_{output}^{c,\bar{a_3},\bar{a_2},a_1,a_0} \\
&= X_{output}^{\bar{c},\bar{a_3},\bar{a_2},a_1,a_0} X_{output}^{c,\bar{a_3},\bar{a_2},a_1}
\end{aligned} \tag{B.5}$$

It should be noted that a proposed optimisation is, generally, not unique. There may be several equivalent improvements against a certain criterion to be considered (e.g. circuit depth). However, it might be useful to consider all options taking into account aspects of the actual quantum processor on which the circuit would run, such as the quality of the connections between the qubits involved or its coupling map limitations. For instance, ignoring the transpilation details and supposing that the c

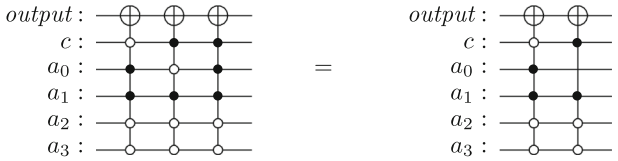

Fig. 12 Graphical representation of the rule optimisation in Eq. (B.5)

Table 6 Graphical optimisation for m value in HighLife rules

m	0	1	2	**3**					4							
g	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$[g \in \Gamma_{\text{births}}]$	0	0	0	1	0	0	1	0	0	✗	✗	✗	✗	✗	✗	✗
$[g \in \Gamma_{\text{survivors}}]$	0	0	1	1	0	0	0	0	0	✗	✗	✗	✗	✗	✗	✗
				start								end				

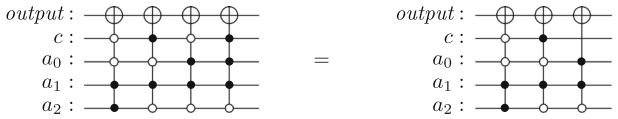

Fig. 13 Graphical representation of the HighLife rule optimisation

qubit has poorer quality connections with the rest of the qubits than the a_0 qubit has, the circuit shown in Fig. 11 would be preferable to the one illustrated in Fig. 12.

Appendix C: General optimisation examples

In the following, the construction of the quantum circuit corresponding to two GoL variants will be optimised as examples. In the case of HighLife (B36/S23) [8], the optimal value of $m = 3$ is determined using Table 6. Consequently, the value of a_3 has been discarded for the following operations. Consequently, a_3 has been discarded for the gate optimisation manipulations in Eq. (C.6) and Fig. 13.

$$
\begin{aligned}
&\prod_{b\in\Gamma_{\text{births}}=\{3,6\}} \left(X_{output}^{c=0,a=b}\right) \prod_{s\in\Gamma_{\text{survivors}}=\{2,3\}} \left(X_{output}^{c=1,a=s}\right) \\
&\qquad = X_{output}^{c=0,a=3} X_{output}^{c=0,a=6} X_{output}^{c=1,a=2} X_{output}^{c=1,a=3} \\
&\qquad = X_{output}^{c=0,a=6} X_{output}^{c=1,a=2} X_{output}^{c=1,a=3} X_{output}^{c=0,a=3} \\
&\qquad = X_{output}^{\bar{c},a_2,a_1,\bar{a_0}} X_{output}^{c,\bar{a_2},a_1,\bar{a_0}} X_{output}^{c,\bar{a_2},a_1,a_0} X_{output}^{\bar{c},\bar{a_2},a_1,a_0} \\
&\qquad = X_{output}^{\bar{c},a_2,a_1,\bar{a_0}} X_{output}^{c,\bar{a_2},a_1,\bar{a_0}} X_{output}^{\bar{a_2},a_1,a_0} \qquad \text{(C.6)}
\end{aligned}
$$

Table 7 Graphical optimisation for m value in Day & Night rules

m	0	1	2		3				**4**							
g	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$[g \in \Gamma_{\text{births}}]$	0	0	0	1	0	0	1	1	1	✗	✗	✗	✗	✗	✗	✗
$[g \in \Gamma_{\text{survivors}}]$	0	0	0	1	1	0	1	1	1	✗	✗	✗	✗	✗	✗	✗

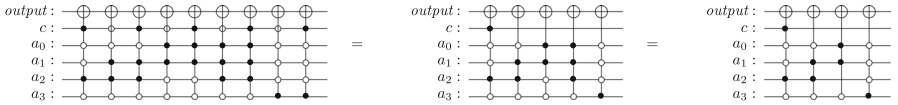

Fig. 14 Graphical representation of the Day & Night rule optimisation

For the second case, Day & Night (B3678/S34678) [13], the value of m was not improved, the attempt can be seen in Table 7. However, subsequent optimisation of the gates was successful, since most of the elements in both sets of rules coincided, allowing the number of gates to be reduced by almost half. This is illustrated in Eq. (C.7) and Fig. 14.

$$
\begin{aligned}
&\prod_{b\in\Gamma_{\text{births}}=\{3,6,7,8\}} \left(X_{output}^{c=0,a=b}\right) \prod_{s\in\Gamma_{\text{survivors}}=\{3,4,6,7,8\}} \left(X_{output}^{c=1,a=s}\right) \\
&= X_{output}^{c=1,a=4} \prod_{x\in\{3,6,7,8\}} \left(X_{output}^{a=x}\right) \\
&= X_{output}^{c=1,a=4} X_{output}^{a=3} X_{output}^{a=6} X_{output}^{a=7} X_{output}^{a=8} \\
&= X_{output}^{c,\bar{a_3},a_2,\bar{a_1},\bar{a_0}} X_{output}^{\bar{a_3},\bar{a_2},a_1,a_0} X_{output}^{\bar{a_3},a_2,a_1,\bar{a_0}} X_{output}^{\bar{a_3},a_2,a_1,a_0} X_{output}^{a_3,\bar{a_2},\bar{a_1},\bar{a_0}} \\
&= X_{output}^{c,\bar{a_3},a_2,\bar{a_1},\bar{a_0}} X_{output}^{\bar{a_3},a_2,a_1,\bar{a_0}} (X_{output}^{\bar{a_3},\bar{a_2},a_1,a_0} X_{output}^{\bar{a_3},a_2,a_1,a_0}) X_{output}^{a_3,\bar{a_2},\bar{a_1},\bar{a_0}} \\
&= X_{output}^{c,\bar{a_3},a_2,\bar{a_1},\bar{a_0}} X_{output}^{\bar{a_3},a_2,a_1,\bar{a_0}} X_{output}^{\bar{a_3},a_1,a_0} X_{output}^{a_3,\bar{a_2},\bar{a_1},\bar{a_0}}
\end{aligned} \tag{C.7}
$$

Author Contributions All authors, especially Jorge Garcia-Diaz, Pino Caballero-Gil and Eduardo Sáenz-de-Cabezón, actively contributed to the extensive discussions that guided the research steps and reviewed the article. Daniel del Castillo led the optimisation of the number of qubits and improvements to the Rust implementation. Daniel Escanez-Exposito made efforts in the design and formalisation of the proposal, the implementation of the quantum circuits with Qiskit and the writing of the article.

Funding This work has been possible thanks to the C065/23 Cybersecurity Chair of the University of La Laguna, funded by INCIBE, with the Recovery, Transformation, and Resilience Plan (Next Generation) financed by the European Union. It is also part of the PID2022-138933OB-I00, PID2024-157733NBI00, and 2023DIG28 IACTA research projects funded by MCIN/AEI/10.13039/501100011033/FEDER EU, and the CajaCanarias la Caixa Foundation.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Competing Interests The authors declare no competing interests.

Ethics approval and consent to participate Not applicable.

Consent for publication Not applicable.

Materials availability Not applicable.

Code availability The source code is available at https://github.com/jdanielescanez/qgol. The GitHub repository includes the quantum simulation, the classical brute-force probability computation, and the GIF generator. It also provides usage instructions and is distributed under the MIT license.

References

1. Gardner, M.: Mathematical games. Scientific american **222**(6), 132–140 (1970)
2. Berlekamp, E.R., Conway, J.H., Guy, R.K.: Winning ways for your mathematical plays, volume 4. AK Peters/CRC Press (2004)
3. Moore, E.F., et al.: Gedanken-experiments on sequential machines. Automata studies **34**, 129–153 (1956)
4. Von Neumann, J., Burks, A.W.: Theory of self-reproducing automata. University of Illinois Press Urbana (1966)
5. Hernández-Montoya, A.R., Tapia-McClung, H.: A Diffusion Approach to the Dynamics of Conway's Game of Life (GoL): Emergence of Multiple Power Law Fluctuation Regimes. Journal of Cellular automata **14** (2019)
6. Nandi, S., Kar, B.K., Chaudhuri, P.P.: Theory and applications of cellular automata in cryptography. IEEE Trans. Comput. **43**(12), 1346–1357 (1994)
7. Chen, Q.: Cellular automata and artificial intelligence in ecohydraulics modelling. CRC Press (2004)
8. Eppstein, D.: Growth and Decay in Life-Like Cellular Automata. In: Adamatzky, A. (ed.) Game of Life Cellular Automata, pp. 71–97. Springer, London (2010)
9. Martínez, G.J., Seck-Tuoh-Mora, J.C., Zenil, H.: Computation and Universality: Class IV versus Class III Cellular Automata. J. Cell. Autom. **7**(5–6), 393–430 (2013)
10. Griffeath, D., Moore, C.: Life without Death is P-complete. Complex Systems **10**(6), 437–447 (1996)
11. Evans, K.M.: Larger than Life: Digital Creatures in a Family of Two-Dimensional Cellular Automata. In: Proc. Discrete Math. Theor. Comput. Sci. (DM-CCG Conference), pp. 177–192. Maison de l'Informatique et des Mathématiques Discrètes, Paris, France (2001)
12. Toffoli, T., Margolus, N.: Cellular Automata Machines: A New Environment for Modeling. MIT Press, Cambridge, MA (1987)
13. Peña, E., Sayama, H.: Life Worth Mentioning: Complexity in Life-Like Cellular Automata. Artif. Life **2**, 105–112 (2021)
14. Bays, C.: The Game of Life in Non-square Environments. In: Adamatzky, A. (ed.) Game of Life Cellular Automata, pp. 319–329. Springer, London (2010)

15. Goucher, A.P.: Gliders in Cellular Automata on Penrose Tilings. J. Cell. Autom. **7**(5–6), 385–392 (2012)
16. Bays, C.: A New Game of Three-Dimensional Life. Complex Systems **5**, 15–18 (1991)
17. Schulman, L.S., Seiden, P.E.: Statistical mechanics of a dynamical system based on Conway's Game of Life. J. Stat. Phys. **19**(4), 293–314 (1978)
18. Aguilera-Venegas, G., Galán-García, J.L., Egea-Guerrero, R., Galán-García, M.Á., Rodríguez-Cielos, P., Padilla-Domínguez, Y., Galán-Luque, M.: A probabilistic extension to Conway's Game of Life. Adv. Comput. Math. **45**(4), 2111–2121 (2019)
19. Rafler, S.: Generalization of Conway's "Game of Life" to a continuous domain-SmoothLife. arXiv preprint **1111.1567** (2011)
20. Chan, B.W.-C.: Lenia: Biology of artificial life. Complex Systems **28**(3), 251–286 (2019)
21. Bleh, D., Calarco, T., Montangero, S.: Quantum Game of Life. Europhys. Lett. **97**(2), 20012 (2012)
22. Ney, P.-M., Notarnicola, S., Montangero, S., Morigi, G.: Entanglement in the quantum game of life. Phys. Rev. A **105**(1), 012416 (2022)
23. Bugu, S., Ozaydin, F., Kodera, T.: Surpassing the classical limit in magic square game with distant quantum dots coupled to optical cavities. Sci. Rep. **10**(1), 22202 (2020)
24. Altintas, A.A., Ozaydin, F., Bayindir, C., Bayrakci, V.: Prisoners' dilemma in a spatially separated system based on spin–photon interactions. Photonics **9**(9) (2022)
25. Javadi-Abhari, A., Treinish, M., Krsulich, K., Wood, C.J., Lishman, J., Gacon, J., Martiel, S., Nation, P.D., Bishop, L.S., Cross, A.W., et al.: Quantum computing with qiskit. arXiv preprint arXiv:2405.08810 (2024)
26. Iverson, K.E.: A programming language. In: Proceedings of the May 1-3, 1962, Spring Joint Computer Conference, pp. 345–351 (1962)
27. Knuth, D.E.: Two notes on notation. Am. Math. Mon. **99**(5), 403–422 (1992)
28. Wang, Y.H.: On the number of successes in independent trials. Statistica Sinica, 295–312 (1993)
29. Escanez-Exposito, D., Caballero-Gil, P., Rodriguez-Vega, M., Costa-Cano, F., Sáenz-de-Cabezón, E.: Algebraic language for the efficient representation and optimization of quantum circuits. Phys. Scr. **100**(2), 025107 (2025)
30. Banerjee, A., Pathak, A.: An algorithm for minimization of quantum cost. Applied Mathematics & Information Sciences **6**(1), 157–165 (2012)
31. Zhang, S., Huang, K., Li, L.: Depth-optimized quantum circuit synthesis for diagonal unitary operators with asymptotically optimal gate count. Phys. Rev. A **109**(4), 042601 (2024)
32. Li, X., Yang, G., Torres, C.M., Jr., Zheng, D., Wang, K.L.: A class of efficient quantum incrementer gates for quantum circuit synthesis. Int. J. Mod. Phys. B **28**(01), 1350191 (2014)
33. Barenco, A., Bennett, C.H., Cleve, R., DiVincenzo, D.P., Margolus, N., Shor, P., Sleator, T., Smolin, J.A., Weinfurter, H.: Elementary gates for quantum computation. Phys. Rev. A **52**, 3457–3467 (1995)
34. Wille, R., Chattopadhyay, A., Drechsler, R.: From reversible logic to quantum circuits: Logic design for an emerging technology. In: 2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation, pp. 268–274 (2016). IEEE
35. Hong, Y.: On computing the distribution function for the poisson binomial distribution. Computational Statistics & Data Analysis **59**, 41–51 (2013)
36. Biscarri, W., Zhao, S.D., Brunner, R.J.: A simple and fast method for computing the poisson binomial distribution function. Computational Statistics & Data Analysis **122**, 92–100 (2018)
37. Ouellet, Y., Quimper, C.-G.: Ao (n\log 2̂ n) checker and o (n 2̂\log n) filtering algorithm for the energetic reasoning. In: CPAIOR, pp. 477–494 (2018)
38. Tilly, J., Chen, H., Cao, S., Picozzi, D., Setia, K., Li, Y., Grant, E., Wossnig, L., Rungger, I., Booth, G.H., et al.: The variational quantum eigensolver: a review of methods and best practices. Phys. Rep. **986**, 1–128 (2022)
39. Machicao, J., Marco, A.G., Bruno, O.M.: Chaotic encryption method based on life-like cellular automata. Expert Syst. Appl. **39**(16), 12626–12635 (2012)
40. McCluskey, E.J.: Minimization of boolean functions. The Bell System Technical Journal **35**(6), 1417–1444 (1956)
41. Brooks, M.: Beyond quantum supremacy: the hunt for useful quantum computers. Nature **574**(7776), 19–22 (2019)
42. Escanez-Exposito, D., del Castillo, D.: Quantum Game of Life repository. https://github.com/jdanielescanez/qgol. [Accessed 30-07-2025] (2025)

43. Nielsen, M.A., Chuang, I.L.: Quantum computation and quantum information. Cambridge University Press (2010)