Optimum Space Complexity Lower Bound Achieved by Quantum Pong Circuit

Michael Coulombe

Electrical Engineering and Computer Science Department
Massachusetts Institute of Technology
32 Vasser Street, Cambridge, MA
mcoulomb@mit.edu

Abstract

Pong™ is a classic game that has been recreated numerous times, but since it was discovered 40 years ago, the space complexity of a quantum computer implementation of Pong has remained an open question. Disproving the short-lived conjecture that it requires $O(wh)$ qubits for an $w \times h$ playing field, $QONG$ is a quantum circuit that implements Pong using $O(\log wh^3)$ qubits to store the game-state and zero scratch qubits, bringing the upper bound on the minimum number of qubits required to the information-theoretic space lower bound as well as only taking polylog time per step. $QONG$ was also implemented in Javascript to play.

1. Introduction

A classical circuit for Pong™was originally discovered by Al Alcorn at Syzygy/Atari in 1972 [4, 5]. Alcorn was keenly aware that space complexity of the Pong circuit was a fundamental problem, as his original circuit had "no memory other than flip flops" and "not a single line of software code was involved in the construction of Pong" [4]. In the past 40 years, the classical Pong circuit has been extensively redesigned and optimized on a variety of platforms, alongside but separated from the development and maturation of the field of quantum computing, despite the discovery of the universality of quantum simulation [3].

In pursuit of the grand unified theory of fundamental physics, complexity theory, and video game design, $QONG$ is the first known quantum circuit for Pong, a circuit which uses the information-theoretic minimum-size game-state, and a reversible algorithm that requires no additional scratch-space qubits.

2. Previous Work

Many have attempted to simulate Pong using quantum mechanical principles, though most researchers have focused on playing Ping-Pong with entangled particles as a communication protocol [1].

Copyright © ACM [to be supplied]. . . $15.00
DOI: http://dx.doi.org/10.1145/(to come)

The most promising result so far was in 2013, when a quantum wave simulation with paddles was developed by Daniel Zerbino [7], which starts with a Gaussian amplitude for the ball that propagates chaotically in every direction until a shared "measurement" button is pressed to collapse the ball into a Gaussian again. While entertaining, Zerbino's "Order 4 Runge-Kutta resolution of the Schroedinger equation applied to a WaveFunction" is not simulating a quantum circuit.

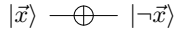
3. The Game

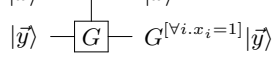
$QONG$ implements Pong as it was originally discovered. Given a discrete field of width w and height h (where $n = \lceil \log_2 w \rceil$ and $m = \lceil \log_2 h \rceil$), there are two paddles of unit width and length ℓ at distance g from the left and right sides and a unit ball that starts in the center. In each step, the ball moves one unit in each direction based on its 2-bit velocity vector, bouncing off of the paddles and the top and bottom walls by reversing its velocity in that direction. The paddles move based on each player's joystick input (neutral, up, or down) within the bounds of the top and bottom walls. The game is lost by the first player to have the ball hit the wall behind their paddle.

Extensions to the game include obstacles which apply $x - z$ or $y - z$ rotations to the ball's velocity vector qubits, as well as having an AI opponent controlling one joystick for single-player mode.

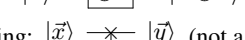
4. The Quantum Model

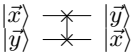
The building blocks for the quantum circuits are as follows:

- Bit-Flip Gate: 

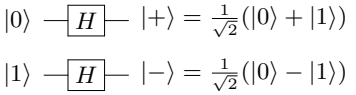
- Control Combinator: 

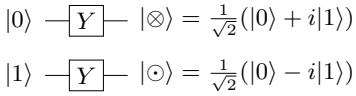
To simplify the drawing of circuits, this notation is used:

- XOR by constant: 

- Physical Wire Crossing:  (not a gate)

Obstacles use these $\frac{\pi}{2}$-rotation gates to create superpositions:

- Hadamard Gate: 

- Y Gate: 

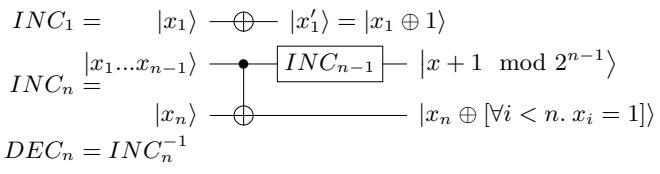


Figure 1. Increment Circuit.

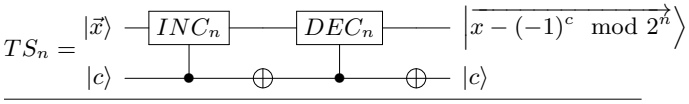


Figure 2. Controlled Increment or Decrement Circuit

5. Circuits

$QONG$ was built in a modular fashion with three foundational innovations: subcircuits INC_n, $LESS_n$, and $J_{m,h}$.

Figure 1 shows INC_n, a circuit which takes in a n-qubit binary number and increments its value, going to zero on overflow. For one qubit, INC_1 reduces simply to a bit flip. For $n > 1$, first the carry is propagated by flipping $|x_n\rangle$ if all less-significant bits are all $|1\rangle$, then recursively they are incremented by INC_{n-1}. Decrementing is done by DEC_n, which is just the inverse of INC_n (performing the gates in reverse order, due to the simplicity of the components).

Given increment and decrement, other essential circuits can be built. Figure 2 shows TS_n which will either increment or decrement $|\vec{x}\rangle$ based on the value of $|c\rangle$. Figure 5 shows that, together with $BNC_{n,h}$, which flips a velocity direction qubit $|d\rangle$ if its input position $|\vec{y}\rangle$ is at the top or bottom wall, $VSTEP_{n,h}$ fully implements the vertical step update for the ball.

Horizontal motion is much more complex than vertical motion due to the presence of moving paddles rather than full walls. $LESS_n$ tests whether $x < p$, given as two n-qubit numbers and output by flipping output qubit $|b\rangle$. Detailed in Figure 6, the base case $LESS_1$ is only true if $x = 0$ and $y = 1$, which is easily testable. For $n > 1$, $LESS_n$ uses the fact that $x < p$ iff either $x_n < p_n$ (like before) or $x_n = p_n$ and the least significant bits of x are less than y's. EQ_2 is used to check $x_n = p_n$ then that result is used to conditionally do the recursive comparison $LESS_{n-1}$ before finally testing $x_n < p_n$. The result is $b \oplus [x < p]$ because the $LESS_{n-1}$ result is used only if the most significant bits are equal, and the $LESS_1$ result is true only if the most significant bits are NOT equal, thus disjointly covering both cases.

The circuit to detect paddles is $HBNC_{n,g,\ell}$, detailed in Figure 7. This circuit is parametrized by c, the horizontal position of the contact side of the paddle, and ℓ, the length of the paddles. Given the position of the ball $|\vec{x}\rangle$ and $|\vec{y}\rangle$, the position of the top of a paddle $|\vec{\alpha}\rangle$, and the horizontal velocity bit $|r\rangle$, $HBNC_{n,g,\ell}$ flips $|r\rangle$ if $x = c$ and $y \in [\alpha, \alpha + \ell)$. By flipping qubits of $|\vec{x}\rangle$ by $\neg c$, they will be set to all $|1\rangle$ if $x = c$, thus it can be used as a conditional on the rest of the circuit. When $x = c$, $LESS_n$ is used to test $y < \alpha$ and $\alpha + \ell - 1 < y$, which non-simultaneously may flip $|r\rangle$, so by flipping the result $|r\rangle$ is flipped only when both checks fail thus when $y \in [\alpha, \alpha + \ell)$.

To finish the horizontal motion of the ball is $HSTEP_{n,w,g,\ell}$, in Figure 8. Rather than the contact position c, this circuit is parametrized by g, the gap between the paddle and the wall behind it, as well as the paddle length ℓ. $HSTEP$ applies $HBNC$ to the left and right paddles at vertical positions $|\vec{\alpha}\rangle$ and $\left|\vec{\beta}\right\rangle$, respectively, then finally used TS_n to apply the new velocity $|r'\rangle$ to the ball's position.

This would be the complete picture if the paddles were stationary, but the final innovation, $J_{m,h}$, is what allows player input to move the paddles. A joystick has three positions: neutral (no motion), up (-1), and down $(+1)$, represented by two qubits a_d, a_u as input to the circuit as shown in Figure 9. A difficulty arises due to reversibility, specifically when the paddle is at the top or bottom wall and the joystick input compels the paddle to push against it. To allow for the output of $J_{m,h}$ to be read to know whether the paddle did not move because of lack of input or a wall was in the way, and conveniently for compatibility with TS_m, the input joystick qubits are given output values that describe the situation. On legal inputs, it outputs $a_p = 1$ when the paddle moved in direction given by a_d', and it outputs $a_p = 0$ when it did not move and $a_d' = 1$ says it was stopped by a wall.

$$J_{m,h}(\alpha, a_d, a_u) \mapsto (\alpha - a_p(-1)^{a_d'}, a_d', a_p)$$

$$J_{m,h}(\alpha, 0, 0) = (\alpha, 0, 0) \quad //\text{no input}$$

$$J_{m,h}(0, 0, 1) = (0, 1, 0) \quad //\text{pushing against top wall}$$

$$J_{m,h}(h - \ell, 1, 0) = (h - \ell, 1, 0) \quad //\text{pushing against bottom wall}$$

$$J_{m,h}(\alpha, 0, 1) = (\alpha - 1, 0, 1) \quad //\text{move up}$$

$$J_{m,h}(\alpha, 1, 0) = (\alpha + 1, 1, 1) \quad //\text{move down}$$

$$J_{m,h}(\alpha, 1, 1) = \text{illegal input}$$

Before these pieces can be combined into a full $QONG$ circuit, one should spice up the game with obstacles which create superpositions, the workhorse of quantum computing. Figure 4 describes OBS_F, a schema for a circuit implementing a field of "obstacles" described by $F \in ([w] \times [h] \times \{H, Y\} \times [2])^*$. When the (x, y) position of the ball match the (u, v) position of the obstacle, an H or Y gate is applied to either the horizontal or vertical velocity qubit. This results in a beam-splitting effect in the superpositions of the ball, which doubles the fun!

Finally, Figure 10 shows the full $QONG_{n,m,h,w,g,\ell,F}$ circuit. In each step, first the obstacles are checked by OBS to beam-split the ball. Next, the ball is moved horizontally by $HSTEP$ then vertically by $VSTEP$. Lastly, the joysticks' inputs are applied in J_m to move the paddles. Extensive playtesting showed that this order of operations resulting in the most natural-feeling play. For example, $HSTEP$ is before $VSTEP$ because if the ball is at the edge of the paddle before the step and both are moving in the same direction, then the ball will bounce off rather than dodge the edge.

Two-player Pong is great fun, but even a superposition of being player 1 and being player 2 does not let one play when alone. Fortunately, perfect classical Pong play was discovered to be computable in P [2]; it is unknown who first proved this, but it is now a staple of competitive play and implemented in $PAI_{m,\ell}$ and used to make $QONGAI_{n,m,h,w,g,\ell,F}$. Detailed in Figure 11, the PAI circuit controls the joystick in lieu of a human player by trying to move the paddle such that the ball's y position is aligned with the center of the paddle vertically. $QONGAI$ integrates PAI by sticking it right before J acting on the α paddle, though it is conceivable that it could also be used on the β paddle.

6. Complexity

The space complexity of a problem is the number of bits of memory required to solve the problem. Unlike a classical circuit, a quantum circuit does not have wires between gates which can store intermediate values separate from the input storage; instead, the input qubits are the sole memory used throughout the computation and used to store the output. Due to this restriction combined with the requirement of reversibility, many quantum circuits have scratch-space qubits which come with the input and are output as "garbage" which must be "collected" or irreversibly thrown away [6].

$QONG_{n,m,h,w,g,\ell,F}$ achieves optimal space complexity both in terms of the encoding of the input and the number of scratch qubits. Given an $w \times h$ field, the game-state necessarily contains the (x, y) position of the ball, $\lceil \log_2 w \rceil + \lceil \log_2 h \rceil$ bits, the vertical (α, β) positions of the paddles, $2\lceil \log_2 h \rceil$ bits, the velocity vector (r, d), 2 bits, and both player's joystick inputs, $2\lceil \log_2 3 \rceil$ bits, thus a total of $O(\log wh^3)$ bits of information. The $QONG$ circuit takes in exactly and only this required amount of qubits as input

The time complexity of a quantum circuit is the maximum depth (number of atomic gates). Decomposing, it is clear that: INC_n takes $O(n + T(INC_{n-1})) = O(n^2)$ time (DEC_n too), TS_n is $O(n^2)$, OBS_F is $O(|F|)$, $BNC_{n,h}$ is $O(5)$ so $VSTEP_{n,h}$ is $O(n^2)$, $LESS_n$ is $O(7 + T(LESS_{n-1})) = O(n)$, $HBNC_{n,c,l}$ is $O(1 + 2n + 2\ell n^2) = O(\ell n^2)$, $HSTEP_{n,w,g,\ell}$ is $O(\ell n^2)$, $J_{m,h}$ is $O(7 + m^2) = O(m^2)$, so $QONG_{n,m,h,w,g,\ell,F}$ is just $O(|F| + \ell n^2 + m^2)$, which is just $\log^2$ in the field dimensions. To implement the AI opponent, $PAI_{m,\ell}$ only adds $O(\ell m^2)$ time. The further time analysis and optimization of $QONG$ is a worldwide ongoing research effort.

7. Simulation Results

As monumental as $QONG$ is for the advancement of science, it truly shines as a video game. Unfortunately, given the lack of practical quantum computers which can support roughly $\log_2 wh^3$ qubits for fun values of w and h, I resorted to a classical simulation to aid the game design.

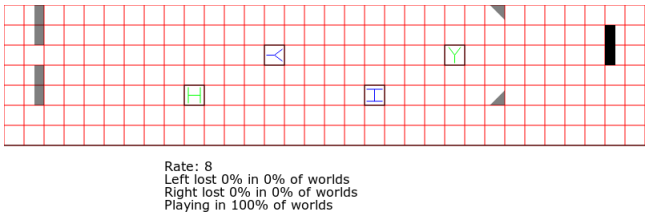


Figure 3. Simulating $QONGAI_{5,3,7,32,1,2,F}$ using field obstacles $F = [(9, 4, H, 1), (13, 2, Y, 0), (18, 4, H, 0), (22, 2, Y, 1)]$.

Figure 3 shows a screenshot of my JavaScript™ simulation, written using the trendy Playground.js framework and supporting single- and multi-player modes. The ball (represented by arrows) was just split into a superposition by the green $|d\rangle \mapsto Y|d\rangle$ obstacle, and the PAI is controlling the paddle in the left, following the ball in each superposition. To keep score, the simulation uses post-selection to remove worlds (terms in the superposition) in which one player loses and it counts both the total percentage of the removed worlds in which each player has lost and the fraction of the remaining percentage that each loss counted for. The simulation ends when some player has lost the game in every world.

Simulation playtests show that $QONG$ is a fun, well-balanced multiplayer game, and that $QONGAI$ is like batting in baseball against a wall made out of pitching machines.

In the spirit of double-blind academic peer-review, the game is graciously hosted under MIT License for play online by a SIGTBD-local source not affiliated with the author: kirsybuu.github.io/qong/

8. Future Work

To celebrate the 45th anniversary of the original Pong™ release, I plan to work with Atari to release $QONG$ as the first true quantum arcade machine. Like it was first marketed [4], the cabinets will be $26'' \times 50'' \times 24''$, at least 150 lbs (including the CRT), and just

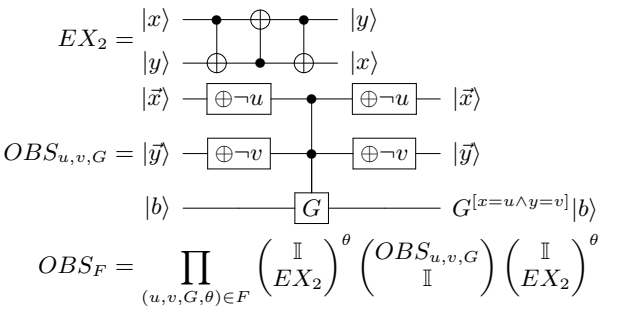


Figure 4. Location-Specific Gates ("Obstacles")

25¢ per play. Pricing is currently being evaluated by partners in the flourishing quantum engineering and arcade industries.

9. Acknowledgements

I would like to thank Prashant Vasudevan for helping with $LESS_n$ and then being too busy to do any actual work, as well as him and William "Mighty Willtor" Leiserson for play-testing.

References

[1] A. Chamoli and C. M. Bhandari. Secure direct communication based on ping-pong protocol. *Quantum Information Processing*, 8(4):347–356, 2009. doi: 10.1007/s11128-009-0112-2.

[2] R. A. Harris and J. B. Gorasia. Pong: An introduction to implementing computer game strategies, 2008. URL http://www.jgorasia.com/Files/Spring08/ICB/Gorasia_Harris.pdf.

[3] S. Lloyd. Universal quantum simulators. *Science*, 273(5278):1073–1078, 1996. ISSN 00368075, 10959203. URL http://www.jstor.org/stable/2899535.

[4] H. Lowood. Videogames in computer space: The complex history of pong. *IEEE Annals of the History of Computing*, 31(3):5–19, July 2009. ISSN 1058-6180. doi: 10.1109/MAHC.2009.53.

[5] B. N. Video image positioning control system for amusement device, Feb. 19 1974. URL https://www.google.com/patents/US3793483. US Patent 3,793,483.

[6] M. Pica Ciamarra. Quantum reversibility and a new model of quantum automaton. *eprint arXiv:quant-ph/0102104*, Feb. 2001.

[7] D. Zerbino. Pong with quantum mechanics. URL https://github.com/dzerbino/QuantumPong.

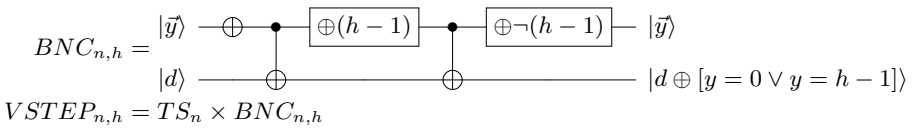


Figure 5. Bouncy Wall Velocity Update and Vertical Motion

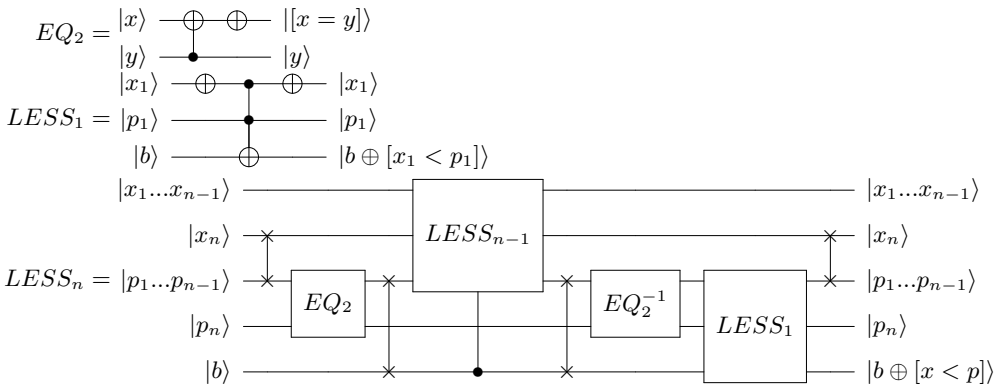


Figure 6. Integer Less-Than Circuit

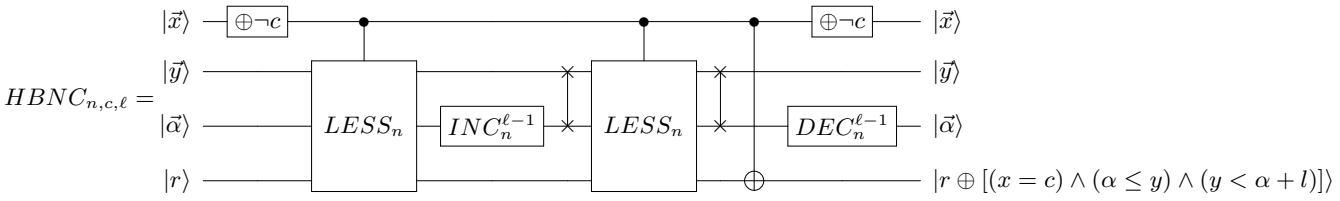


Figure 7. Paddle Bounce Circuit.

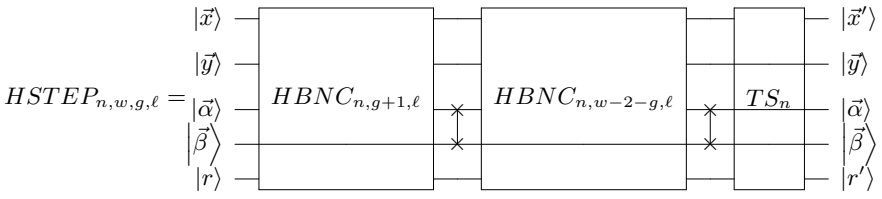


Figure 8. Horizontal Velocity Update and Motion Circuit

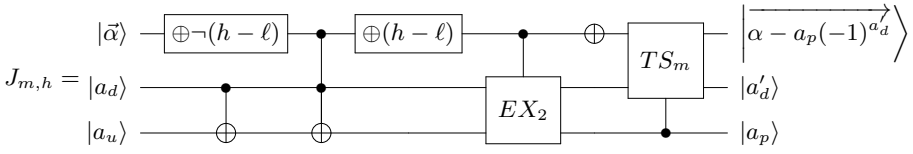


Figure 9. Joypad Paddle Control Circuit

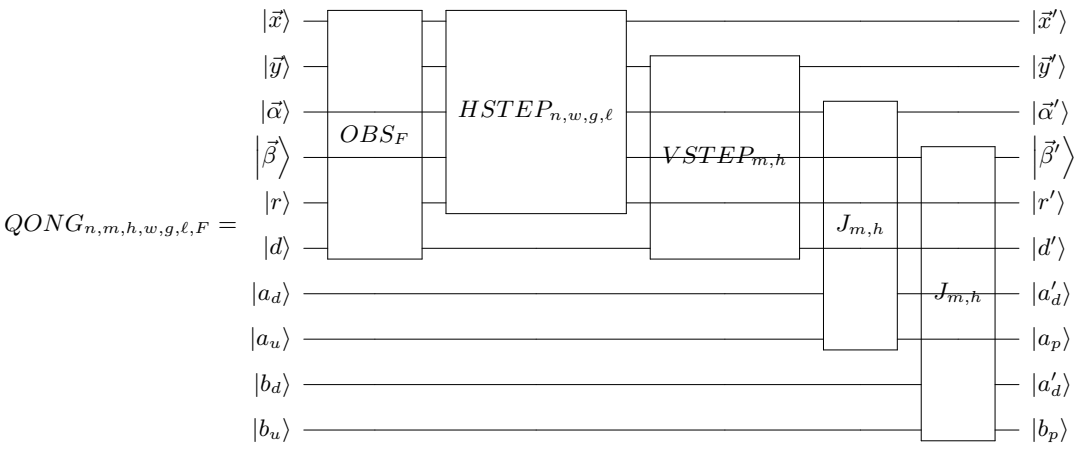


Figure 10. One-Step, Two-Player Quantum Pong Circuit

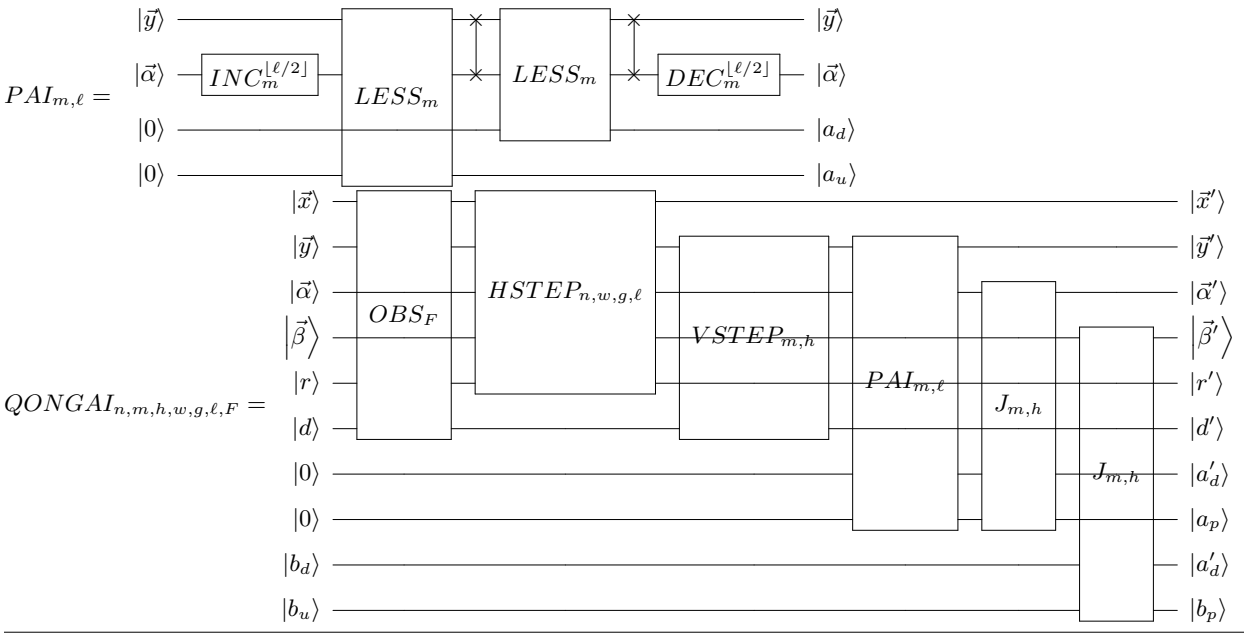


Figure 11. One-Step, One-Player Quantum Pong Circuit