Universidad de Nariño

Facultad de Ciencias Exactas y Naturales

Departamento de Física

EVALUACIÓN DE LA VIABILIDAD TÉCNICA Y OPTIMIZACIÓN DEL ALGORITMO DE SHOR EN PROCESADORES NISQ: UN ESTUDIO COMPARATIVO DE FIDELIDAD Y TRANSPILACIÓN EN IBM QUANTUM

TRABAJO DE GRADO

Para optar al título profesional de:

Físico

Christopher Andrey Villota Freire

San Juan de Pasto, Colombia

junio de 2026

Universidad de Nariño

Facultad de Ciencias Exactas y Naturales

Departamento de Física

EVALUACIÓN DE LA VIABILIDAD TÉCNICA Y OPTIMIZACIÓN DEL ALGORITMO DE SHOR EN PROCESADORES NISQ: UN ESTUDIO COMPARATIVO DE FIDELIDAD Y TRANSPILACIÓN EN IBM QUANTUM

Christopher Andrey Villota Freire

TRABAJO DE GRADO

Director:

Ph.D Jorge Hernán López Melo

San Juan de Pasto, Colombia

junio de 2026

Nota de aceptación

_____

_____

_____

_____

_____

_____

Jorge Hernán López Melo

_____

Director

Paola Concha Obando

_____

Jurado

Oscar Cadena Ibarra

_____

Jurado

San Juan de Pasto, junio de 2026

Agradecimientos

Al culminar esta etapa, miro hacia atrás y comprendo que ningún logro es enteramente propio: detrás de cada paso hay manos que sostienen, voces que alientan y presencias que acompañan en silencio. Por ello, estas líneas son apenas un humilde intento de devolver, en forma de palabras, lo mucho que he recibido a lo largo de este camino.

A mis padres, Claudia y Carlos, mi gratitud más sincera y eterna. Gracias por creer en mí incluso cuando yo mismo dudaba, por entregarme lo mejor de ustedes sin reservas y por enseñarme, con el ejemplo, que el esfuerzo honesto y la perseverancia son los cimientos sobre los que se construye una vida con sentido. Cada renuncia suya se ha transformado en una conquista mía, y este trabajo es una pequeña forma de honrar todo lo que han sembrado en mí.

A todos los profesores que dejaron una huella en mi formación, mi reconocimiento sincero. Gracias por compartir, con generosidad y vocación, no solo el conocimiento de su disciplina, sino también la pasión con que la habitan. Cada clase, cada explicación paciente y cada pregunta provocadora contribuyó a moldear la mirada con la que hoy me asomo a la física y al mundo.

De manera muy especial, agradezco a mi director de trabajo de grado, el profesor Jorge Hernán López Melo, por su acompañamiento riguroso, su confianza y su disposición permanente a lo largo de este proceso. Gracias por todo lo aprendido dentro y fuera del aula, por las conversaciones que trascendieron lo estrictamente académico y por mostrarme, con su ejemplo, que enseñar es también un acto de generosidad y compromiso. Su guía paciente fue fundamental para que este trabajo encontrara su rumbo, y las enseñanzas que llevo conmigo van mucho más allá de lo que estas páginas pueden contener.

Por último, pero no por ello menos importante, agradezco a la Universidad de Nariño, casa de saber que me abrió sus compuertas y se convirtió en mi segundo hogar durante estos años. En sus aulas, pasillos y laboratorios aprendí no solo física, sino también el valor del pensamiento crítico, del trabajo colectivo y del compromiso con la región. Espero que este modesto aporte sea una pequeña retribución a la oportunidad invaluable que me brindó de formarme como físico y como persona.

A todos, mi gratitud profunda y sincera.

Resumen

Este trabajo de grado evalúa la viabilidad técnica y el desempeño del algoritmo de Shor para la factorización del entero $N = 15$ en procesadores cuánticos NISQ de IBM, comparando los resultados obtenidos en tres entornos de ejecución progresivamente más realistas: simulación ideal, simulación con modelo de ruido calibrado (*Fake Backend* basado en `ibm_torino`) y ejecución en el procesador real `ibm_torino` (familia Heron r1, 133 qubits superconductores). La implementación se realizó en Python utilizando el SDK Qiskit. Se aplicaron tres estrategias de optimización y supresión de errores: el algoritmo de ruteo SABRE para adaptar el circuito a la topología *Heavy-Hex*, el Desacoplamiento Dinámico (DD) para reenfocar la decoherencia y el Pauli Twirling (PT) para convertir errores coherentes en ruido estocástico. El desempeño se cuantificó mediante la fidelidad de Hellinger y la probabilidad de éxito experimental (PST). Los resultados demuestran experimentalmente la ejecución exitosa de una versión compilada del algoritmo de Shor para $N = 15$ en hardware real, recuperando los factores 3×5 a partir de la fase medida, con una señal máxima de $PST = 82{,}8\,\%$ y una tasa de éxito cercana al $90\,\%$ en la extracción de los factores no triviales sobre las bases coprimas válidas. En el conjunto de configuraciones evaluadas, el nivel de optimización del transpilador fue la variable de control con *mayor impacto observado* sobre la señal experimental, con una ganancia de hasta dos órdenes de magnitud en PST entre `opt=0` y `opt=2` sobre la base $a = 4$. Por contraste, las técnicas de supresión de errores mostraron un efecto contraproducente en el régimen estudiado: el Desacoplamiento Dinámico degradó la señal en más de 60 puntos porcentuales. Se documenta, sobre las dos bases con medición dependiente, disponibles en ambos entornos, que el *Fake Backend* utilizado subestima la calidad del hardware real en el régimen de circuitos de profundidad moderada estudiado. El trabajo aporta evidencia experimental y una metodología reproducible para futuros estudios en computación cuántica en la Universidad de Nariño.

Palabras clave: computación cuántica, algoritmo de Shor, era NISQ, IBM Quantum, transpilación, SABRE, Desacoplamiento Dinámico, Pauli Twirling, fidelidad de Hellinger.

Abstract

This thesis evaluates the technical feasibility and performance of Shor's algorithm for the factorization of the integer $N = 15$ on IBM NISQ quantum processors, comparing the results obtained in three progressively more realistic execution environments: ideal simulation, simulation with a calibrated noise model (*Fake Backend* based on `ibm_torino`), and execution on the real `ibm_torino` processor (Heron r1 family, 133 superconducting qubits). The implementation was carried out in Python using the Qiskit SDK. Three optimization and error suppression strategies were applied: the SABRE routing algorithm to adapt the circuit to the *Heavy-Hex* topology, Dynamical Decoupling (DD) to refocus decoherence, and Pauli Twirling (PT) to convert coherent errors into stochastic noise. Performance was quantified using the Hellinger fidelity and the experimental probability of success (PST). The results experimentally demonstrate the successful execution of a compiled version of Shor's algorithm for $N = 15$ on real hardware, recovering the 3×5 factorization from the measured phase, with a maximum signal of $PST = 82{,}8\,\%$ and a success rate close to $90\,\%$ in the extraction of non-trivial factors over the valid coprime bases. Across the set of evaluated configurations, the transpiler optimization level was the control variable with the *largest observed impact* on the experimental signal, yielding gains of up to two orders of magnitude in PST between `opt=0` and `opt=2` for base $a = 4$. In contrast, the error suppression techniques showed a counterproductive effect in the studied regime: Dynamical Decoupling degraded the signal by more than 60 percentage points. It is also documented, over the two bases for which dependent measurements are available in both environments, that the *Fake Backend* used underestimates the quality of real hardware in the moderate-depth regime studied. This work provides experimental evidence and a reproducible methodology for future research in quantum computing at the University of Nariño.

Keywords: quantum computing, Shor's algorithm, NISQ era, IBM Quantum, transpilation, SABRE, Dynamical Decoupling, Pauli Twirling, Hellinger fidelity.

Contenido

1. Introducción **1**

2. Planteamiento del problema **3**

3. Objetivos **5**
3.1. Objetivo general 5
3.2. Objetivos específicos 5

4. Marco referencial **6**
4.1. Marco de antecedentes 6
4.2. Fundamentos de la computación cuántica 9
4.2.1. Qubit, notación y matrices de Pauli 9
4.2.2. Evolución, medición y sistemas compuestos 9
4.2.3. Superposición y entrelazamiento 10
4.2.4. Modelo de circuitos y compuertas 10
4.2.5. Ruido NISQ: T_1, T_2, canales de error y errores de lectura 13
4.3. El algoritmo de Shor 14
4.3.1. Transformada cuántica de Fourier (QFT) 16
4.3.2. Estimación de fase cuántica (QPE) y cotas de error 17
4.3.3. Búsqueda de orden y exponenciación modular 18
4.3.4. Reducción clásica: de la factorización a la búsqueda de orden 21
4.3.5. Ejemplo guiado: factorización de $N = 15$ 23
4.4. El contexto NISQ y la arquitectura de IBM Quantum 25
4.4.1. La era NISQ: qubits físicos frente a qubits lógicos 25
4.4.2. Arquitectura de IBM: qubits superconductores transmon 26
4.4.3. Topología Heavy-Hex y el cuello de botella de la conectividad 27
4.5. Transpilación y algoritmo SABRE 28
4.5.1. El proceso de transpilación y niveles de optimización 28
4.5.2. Algoritmo SABRE 30
4.6. Estrategias de supresión en compilaciones complejas 32
4.6.1. Desacoplamiento Dinámico (Dynamical Decoupling) 32
4.6.2. Pauli Twirling (PT) 33
4.7. Métricas de Evaluación 34

4.7.1. Fidelidad de Hellinger 34
4.7.2. Probabilidad de Éxito Experimental (PST) 35

5. Diseño metodológico e implementación experimental 37
5.1. Delimitación del aporte respecto al repositorio base 37
5.2. Pre-procesamiento clásico y selección de parámetros 37
5.2.1. Selección de la base a y caracterización del grupo multiplicativo . . . 38
5.2.2. Dimensionamiento de los registros cuánticos 38
5.3. Construcción del circuito ideal 39
5.3.1. Síntesis del oráculo de exponenciación modular 39
5.4. Protocolo experimental en tres fases 41
5.5. Transpilación 44
5.6. Técnicas de supresión de errores 45
5.7. Post-procesamiento clásico y métricas de evaluación 46

6. Resultados 48
6.1. Resultados de la simulación ideal 48
6.1.1. Caracterización de los bloques fundamentales 48
6.1.2. Simulación ideal pura: línea base sin restricción topológica 50
6.1.3. Transpilación a la topología de FakeTorino 52
6.1.4. Impacto de la estrategia de mapeo: trivial vs. SABRE 53
6.1.5. Post-procesamiento clásico: extracción por fracciones continuas . . . 55
6.2. Resultados de la simulación con ruido: FakeTorino 56
6.2.1. Choque con el ruido: comparación frente a la simulación ideal 56
6.2.2. Efecto de las técnicas de supresión de errores 59
6.2.3. Post-procesamiento bajo ruido: la robustez del filtro clásico 60
6.3. Ejecución experimental en hardware real: IBM Torino 61
6.3.1. Descripción de la ejecución experimental 61
6.3.2. Resultados sin supresión de errores 63
6.3.3. Efecto de la base a en la señal de hardware 64
6.3.4. Impacto de las técnicas de supresión de errores 67
6.3.5. Análisis y discusión 70

7. Conclusiones 73
7.1. Resultados principales 73
7.2. Limitaciones del trabajo 75
7.3. Perspectivas futuras 76
7.4. Reflexión final 77

A. Preliminares matemáticos y formalismo cuántico 79
A.1. Canales cuánticos y representación de Kraus 79
A.2. Canal despolarizante post-compuerta 79
A.3. Exponenciación modular 80

A.4. Extracción del orden mediante fracciones continuas 81
A.5. Formalismo del Desacoplamiento Dinámico 82
A.6. Formalismo del Pauli Twirling 84

B. Arquitectura del software e implementación 86
B.1. Migración técnica del repositorio base a Qiskit 2.x 86
B.2. Estructura modular del proyecto 88
B.3. Configuración detallada del transpilador y SABRE avanzado 89
B.4. Configuración detallada de las técnicas de supresión 91
B.5. Scripts de ejecución por fase 92
B.6. Módulos de post-procesamiento clásico 93
B.7. Configuración del entorno y versiones de software 94

C. Tablas complementarias 95
C.1. Verificación de la Transformada Cuántica de Fourier 95
C.2. Bloques de exponenciación modular controlada 96
C.3. Tablas de resultados en simulación ideal 98
C.4. Tablas de resultados en simulación con Ruido 103
C.5. Tablas de resultados de ejecución en hardware IBM 104

Índice de tablas

4.1. Tabla de verdad para la compuerta CNOT, mostrando la inversión del qubit objetivo condicionado al qubit de control. 12

4.2. Niveles de optimización en el transpilador de Qiskit [53]. 29

5.1. Caracterización del grupo $\mathbb{Z}_{15}^*$: bases, órdenes y factores obtenidos. 38

5.2. Configuración de registros cuánticos para $N = 15$. 39

6.1. Cadena de post-procesamiento clásico para los resultados con $y \neq 0$ obtenidos en simulación ideal pura ($N = 15$, $t = 9$, $M = 4096$ *shots*). 55

6.2. Configuración del backend y parámetros experimentales. 61

6.3. Resumen de *jobs* ejecutados en `ibm_torino`. 62

6.4. Resultados por base a en hardware real (`ibm_torino`, `opt` $= 3$, `approx` $= 0{,}7$, PT activado en todos los *jobs*). 65

6.5. Distribución de probabilidad para $a = 4$, $r = 2$ en `ibm_torino` (4096 *shots*). Los estados $|000000000\rangle$ y $|100000000\rangle$ corresponden a las fases teóricas $\varphi = 0$ y $\varphi = 1/2$, respectivamente. 66

6.6. Comparación de técnicas de supresión de errores en hardware real (`ibm_torino`, $a = 4$, `opt` $= 3$, `approx` $= 0{,}7$, 116 CZ, 4096 *shots*). Se reportan la señal PST, la fidelidad $\mathcal{F}_H$, el número de *outcomes* únicos y los factores extraídos. 67

6.7. Efecto del *Pauli Twirling* para $a = 4$ en `ibm_torino` (116 CZ, 4096 *shots*). 68

6.8. Contraste del efecto de la supresión entre el simulador ruidoso (FakeTorino, Fase 2) y el hardware real (`ibm_torino`, Fase 3), para $a = 4$, `opt` $= 3$. La celda *Solo PT* en HW se obtiene por inferencia indirecta de V1. *Nota:* los valores de FakeTorino reportados aquí provienen de la corrida del estudio de supresión (Tabla C.11); por eso, para la fila *sin supresión*, se reproduce el par $(PST_{\text{FT}}, \mathcal{F}_H^{\text{FT}}) = (3{,}52\,\%,\ 0{,}0039)$ del estudio de supresión, en lugar del par $(4{,}30\,\%,\ 0{,}0010)$ que reporta la Tabla C.9 (corrida independiente con la misma configuración nominal). 70

6.9. Degradación de la señal PST (%) desde la simulación ideal hasta el hardware real, `opt` $= 3$, `approx` $= 0{,}7$. La columna FakeTorino corresponde al *baseline* sin supresión de la Fase 2 y solo está disponible para $a \in \{4, 7\}$. La columna HW agrupa V1 ($a = 4, 7$, PT activado) y A3 ($a \in \{2, 8, 11, 13\}$, PT activado). 71

B.1. Delimitación de aportes y resumen de la migración a Qiskit 2.x: repositorio base frente a contribución propia. 87

B.2. Parámetros del transpilador empleados en el trabajo. 90

B.3. Hiperparámetros avanzados de SABRE. 90

B.4. Parámetros de `DynamicalDecouplingOptions` empleados. 91

B.5. Parámetros de `TwirlingOptions` empleados. 91

B.6. *Scripts* de ejecución por fase del protocolo experimental. 92

B.7. Organización de los 13 *jobs* ejecutados en `ibm_torino`. 93

B.8. Entorno de software utilizado en los experimentos. 94

C.1. Métricas de profundidad y verificación unitaria de la QFT para $n = 9$ qubits. La columna $\|U^\dagger U - I\|_F$ cuantifica la desviación de la unitariedad; valores del orden de 10^{-14} confirman precisión numérica de punto flotante. 95

C.2. Profundidad de dos qubits (Depth 2Q) y conteo de compuertas de dos qubits (Gates 2Q) para el circuito completo de exponenciación modular controlada, por base. La profundidad total es la suma de los bloques no triviales $C\text{-}U_{a^{2^k}}$ con a^{2^k} mód $N \neq 1$. 96

C.3. Métricas del circuito completo `RegisterQC` (13 Qubits) para $N = 15$ antes (pre-transpilación) y después (post-transpilación a la base $CX + U3$, sin restricción topológica). La *Depth Total (pre)* corresponde a las 12 capas secuenciales del circuito de alto nivel: una compuerta X de inicialización, nueve pares $H - C - U_{a^{2k}}$ aplicados en secuencia sobre cada qubit de control, una QFT^{-1} y una medición final. La post-transpilación revela el costo real en compuertas de dos qubits. Entiéndase (D.) como Depth (profundidad). 98

C.4. Métricas del circuito `RegisterQC` ejecutado en simulación ideal pura (*all-to-all*) para $N = 15$, $t = 9$ y 4096 shots (transpilado con `AerSimulator`), evaluadas bajo los cuatro niveles de optimización del transpilador de Qiskit. Se reportan la profundidad total (D), la profundidad restringida a compuertas de dos qubits (D_{2Q}), el conteo de compuertas de dos qubits (G_{2Q}), el conteo total de compuertas (G_{tot}) y la probabilidad de señal (PST). El estado Δ indica que, pese a la correcta identificación del orden r, los factores clásicos son triviales. 99

C.5. Métricas del circuito `RegisterQC` transpilado a la topología Heavy-Hex de `FakeTorino` (Heron r1) y ejecutado en simulación ideal (sin modelo de decoherencia ni errores de compuerta) para $N = 15$, $t = 9$ y 4096 shots. Se reportan la profundidad y el conteo de compuertas de dos qubits (D_{2Q}, G_{2Q}), la probabilidad de señal (PST), la probabilidad de ruido (masa fuera de los picos teóricos del QPE) y los factores extraídos. El estado Δ marca los casos con factores triviales por $a \equiv -1 \pmod{N}$. 100

C.6. Comparativa del *overhead* de ruteo Heavy-Hex: profundidad y conteo de compuertas de dos qubits entre la simulación pura (*all-to-all*) y la transpilación a `FakeTorino` bajo los cuatro niveles de optimización del transpilador de Qiskit. Los ratios $D_{2Q}^{\text{Torino}}/D_{2Q}^{\text{Puro}}$ y $G_{2Q}^{\text{Torino}}/G_{2Q}^{\text{Puro}}$ cuantifican el factor de amplificación introducido por la topología. 101

C.7. Métricas de transpilación para el circuito `RegisterQC` ($N = 15$) bajo los mapeos trivial y SABRE, transpilado sobre la topología Heavy-Hex de `FakeTorino` con 4096 *shots* y semilla 457. El conteo de SWAPs explícitas reportado por Qiskit es nulo en todas las configuraciones debido a que el pase de traducción al conjunto de compuertas nativas (`BasisTranslator`) descompone cada SWAP en tres compuertas de dos qubits, que quedan contabilizadas en Gates_{2Q}. 102

C.8. Reducción porcentual lograda por SABRE respecto al mapeo trivial en profundidad de dos qub125its (D_{2Q}) y conteo de compuertas de dos qubits (G_{2Q}), para cada base a y nivel de optimización. Se observa una dependencia no monótona con el nivel de optimización: un mínimo local en Opt=1 y máximos en Opt=2. 103

C.9. Métricas de la simulación con ruido (FakeTorino, sin supresión) para $N = 15$. Se reportan la profundidad de compuertas de dos qubits (D2Q), el conteo de compuertas de dos qubits (G2Q), la probabilidad de señal teórica (PST), la fidelidad de Hellinger ($\mathcal{F}_H$) y los factores extraídos. 103

C.10. Análisis detallado de fidelidad con ruido (`opt` $= 3$, FakeTorino). Se reportan la fidelidad de Hellinger ruidosa e ideal, la degradación $\Delta\mathcal{F}_H = \mathcal{F}_H^{\text{ideal}} - \mathcal{F}_H^{\text{ruido}}$. 103

C.11. Comparación de técnicas de supresión (`opt` $= 3$, FakeTorino, 512 *shots*). Se reportan PST, $\mathcal{F}_H$ y los factores extraídos para cada variante. *Nota:* la fila de *a*=4 sin supresión ($PST = 3{,}52\,\%$, $\mathcal{F}_H = 0{,}0039$) corresponde a la corrida del estudio de mitigación; la línea base equivalente reportada en la Tabla C.9 ($PST = 4{,}30\,\%$, $\mathcal{F}_H = 0{,}0010$) corresponde a una corrida independiente con la misma configuración nominal pero distinta semilla de *sampling*. Las diferencias del orden de un punto porcentual son compatibles con la variabilidad esperada en distribuciones cercanas a la uniformidad sobre $M = 512$ *shots*. 104

C.12. Impacto del nivel de optimización del transpilador en hardware real (`ibm_torino`, $a = 4$, `approx` $= 0{,}7$, 4096 *shots*, Pauli Twirling activado). Se reportan la profundidad 2Q, el conteo de compuertas CZ, el número de SWAPs estimados, la señal PST y la fidelidad $\mathcal{F}_H$. 104

Índice de figuras

4.1. Nombres, símbolos y matrices unitarias para las compuertas comunes de un solo qubit [24]. 11
4.2. Representación de la equivalencia entre compuertas CNOT y SWAP, operaciones fundamentales para el enrutamiento de circuitos mediante el algoritmo SABRE. Fuente: Gemini, Google, (2026). 12
4.3. Representación de la operación controlada-U [24]. 12
4.4. Esquema general del algoritmo de Shor (Fuente: creada con Claude, Anthropic, (2026).) 15
4.5. Circuito eficiente para la transformada cuántica de Fourier. No se muestran las compuertas de intercambio al final del circuito, las cuales invierten el orden de los cúbits, ni los factores de normalización de $1/\sqrt{2}$ en la salida [24]. . . . 16
4.6. Primera etapa del procedimiento de estimación de fase. Se han omitido los factores de normalización de $1/\sqrt{2}$, a la derecha [24]. 17
4.7. Esquema del procedimiento general de estimación de fase. Los t qubits superiores (la barra '/' denota un haz de cables) [24]. 18
4.8. Circuito cuántico para el algoritmo de búsqueda de orden. Este circuito también se puede utilizar para la factorización [24]. 19
4.9. Imágenes de un cúbit transmon sintonizable en frecuencia. (a) El transmon completo consta de una capacitancia grande (en forma de "+") en paralelo con un SQUID conectado a tierra. En (b) y (c) se muestra un acercamiento del SQUID y una unión Josephson, respectivamente. [50] 26
4.10. Topología Heavy-Hex del procesador cuántico IBM *ibm_fez* (panel derecho) de la familia Heron r2 con 156 qubits. A la izquierda los datos de calibración mas importantes [19]. No se muestra el procesador *ibm_torino* usado en el trabajo ya que fue retirado de los recursos de IBM. 27
4.11. Ejemplo ilustrativo de transpilación de un circuito cuántico simple en Qiskit para un procesador IBM con topología Heavy-Hex [53]. A la izquierda se muestra el circuito lógico original. A la derecha aparece el circuito transpilado, donde se han insertado compuertas nativas del hardware y se han mapeado los qubits lógicos q_0 y q_1 a los qubits físicos 58 y 57, respectivamente. Nótese cómo incluso un circuito sencillo requiere varias compuertas de dos qubits (*ECR* en Eagle) y rotaciones adicionales para adaptarse a la conectividad restringida del procesador. 30

4.12. Ejemplo de búsqueda heurística basada en SWAP [8]. 31

4.13. Actualización inicial del mapeo mediante la técnica de recorrido inverso [8]. . 31

4.14. Diagrama esquemático de secuencias DD. (a) Dos secuencias CPMG concatenadas. (b) Dos secuencias XY4 concatenadas. Todas las secuencias se muestran para un período de inactividad de 4τ. CPMG usa solo compuertas X (rectángulo azul), mientras que XY4 usa compuertas X e Y (rectángulos verdes) [62]. 33

5.1. Representación esquemática del algoritmo de Shor para la búsqueda de periodo. El registro superior ($n = 9$ qubits) se inicializa en una superposición uniforme mediante compuertas Hadamard (H). El registro inferior ($m = 4$ qubits) se inicializa en el estado computacional $|1\rangle$ mediante una compuerta X. La cascada de oráculos representa la exponenciación modular controlada $c\text{-}U_{a^{2^j} \pmod N}$, donde cada base $b = a^{2^j} \pmod N$ ha sido pre-computada clásicamente. El proceso finaliza con una Transformada de Fourier Cuántica Inversa ($QFT^\dagger$) y la medición del registro de control. 40

5.2. Diagrama de flujo del protocolo experimental por fases. (Fuente: creada con Claude, Anthropic, (2026).) 43

6.1. Diagrama esquemático del circuito `RegisterQC` para las bases $a = 4$ (superior) y $a = 7$ (inferior) con $N = 15$. La duplicación visible de bloques entrelazantes en $a = 7$ refleja directamente que su orden $r = 4$ activa dos potencias no triviales de U_a. 49

6.2. Histogramas de medición del registro de conteo en simulación ideal del algoritmo de Shor ($N = 15$, $t = 9$, 4096 *shots*) para $a = 4$ (a) y $a = 7$ (b), la distribución de las otras dos bases restantes es análoga a $a = 4$. Las líneas verticales punteadas marcan los picos teóricos y_s predichos por el QPE; las barras coloreadas muestran los conteos observados. 50

6.3. Resultados de métricas de circuitos para la simulación ideal del algoritmo de Shor ($N = 15$) asumiendo una topología de conectividad completa (*all-to-all*), evaluado en los niveles de optimización de transpilación del 0 al 3 para las bases coprimas $a \in \{4, 7, 11, 14\}$. **(a)** Señal ideal pura obtenida, demostrando que en ausencia de decoherencia y errores operacionales, la probabilidad teórica se mantiene en un 100 % invariable frente a la optimización. **(b)** Profundidad del circuito en función de las compuertas de dos qubits (2Q). **(c)** Conteo total de compuertas (G_{tot}) presentes en el circuito transpilado. Se destaca que la base $a = 7$ requiere un costo de recursos lógicos significativamente mayor en comparación con las bases $a = 4, 11$ y 14, las cuales presentan un comportamiento superpuesto y una ligera mitigación en la cantidad de compuertas a medida que se incrementa el nivel de optimización. 51

6.4. Evaluación del *overhead* de ruteo al transpilar el algoritmo para $N = 15$ hacia la topología Heavy-Hex de `FakeTorino`, en función de los cuatro niveles de optimización de Qiskit para las bases $a \in \{4, 7, 11, 14\}$. **(a)** Conteo absoluto de compuertas de dos qubits (G_{2Q}) resultantes tras la adaptación a la topología física. **(b)** Profundidad absoluta del circuito restringida a compuertas de dos qubits (D_{2Q}). **(c)** Factor de amplificación (ratio) del número de compuertas en comparación con la simulación ideal pura (*all-to-all*). **(d)** Factor de amplificación de la profundidad en comparación con el caso ideal. Se evidencia que los niveles de optimización superiores (2 y 3) reducen sustancialmente el costo introducido por la conectividad limitada del hardware. 52

6.5. Comparación de las métricas de transpilación para el circuito con $N = 15$. Se evalúa el rendimiento del mapeo trivial frente al enrutamiento SABRE bajo distintos niveles de optimización (Opt) y valores base (a). Los paneles muestran: (a) la profundidad total del circuito transpilado, (b) la profundidad exclusiva de compuertas de dos qubits (D_{2Q}), (c) el conteo total de compuertas de dos qubits (G_{2Q}), y (d) la Fidelidad de Hellinger (F_H), la cual se mantiene estable e ideal en la mayoría de los casos. 54

6.6. Reducción porcentual lograda por el algoritmo de enrutamiento SABRE respecto al mapeo trivial para $N = 15$. (a) Porcentaje de reducción en la profundidad de compuertas de dos qubits (D_{2Q}). (b) Porcentaje de reducción en el conteo de compuertas de dos qubits (G_{2Q}). Se destaca una dependencia no monótona respecto al nivel de optimización de Qiskit, observándose un mínimo local de mejora en Opt = 1 y el máximo desempeño general en Opt = 2. 55

6.8. Resultados de métricas de rendimiento para la simulación con ruido (FakeTorino) del algoritmo de Shor ($N = 15$) utilizando las bases $a = 4$ (rojo) y $a = 7$ (azul), bajo los niveles de optimización de transpilación 0 y 3. **(a)** Profundidad del circuito en términos de compuertas de dos qubits (D2Q). **(b)** Conteo total de compuertas de dos qubits (G2Q) requeridas en los circuitos transpilados. **(c)** Probabilidad de Señal Teórica (PST), donde las barras sólidas indican los porcentajes obtenidos bajo el modelo de ruido y las áreas tenues punteadas representan el límite ideal del 100 %. **(d)** Fidelidad de Hellinger ($\mathcal{F}_H$) calculada entre las distribuciones de estados obtenidas y las teóricas; el marco tenue ilustra la fidelidad ($= 1{,}0$) correspondiente a un entorno sin ruido. Se observa de manera general que, aunque el nivel de optimización 3 reduce las métricas de profundidad y conteo de compuertas, esto no se traduce necesariamente en una mejora de la PST o la $\mathcal{F}_H$ para todas las bases bajo la influencia del ruido. 57

6.7. Histogramas del algoritmo de Shor ($N = 15$, $t = 9$, $M = 512$ *shots*) con `FakeTorino`. Se presentan cuatro configuraciones: (a) base $a = 4$, opt=0, $r = 2$; (b) base $a = 4$, opt=3, $r = 2$; (c) base $a = 7$, opt=0, $r = 4$; (d) base $a = 7$, opt=3, $r = 4$. Las líneas verticales discontinuas marcan los picos teóricos $y_s = s\,2^m/r$ y la horizontal punteada la altura teórica ideal $1/r$. Los recuadros internos reproducen cada distribución en escala logarítmica para hacer visible el *piso de ruido*. 58

6.9. Evaluación comparativa de técnicas de supresión de errores en simulación ruidosa. El panel (a) muestra la probabilidad de éxito (PST) y el panel (b) detalla la fidelidad (F_H) para $a = 4$ y $a = 7$. En el eje horizontal se contrastan cuatro escenarios experimentales: ejecución estándar (Sin supresión), DD, PT, y la combinación de ambos métodos (DD + PT). 60

6.10. Impacto del parámetro `approximation_degree` sobre la profundidad de compuertas de dos qubits (D_{2Q}) tras la transpilación a la topología Heavy-Hex de `ibm_torino`, para las bases a ejecutadas en hardware. La configuración exacta (`approx` $= 1{,}0$) produce profundidades entre 430 y 910 capas CZ, mientras que la configuración aproximada (`approx` $= 0{,}7$) las reduce a entre 12 y 100 capas, una compresión de un orden de magnitud. Esta reducción es la que hace viable la ejecución dentro del tiempo de coherencia del procesador. 62

6.11. Distribuciones de probabilidad observadas en `ibm_torino` (verde) frente a la distribución teórica ideal (azul) para las cuatro bases del estudio A3. En cada panel se reportan únicamente los *bitstrings* asociados a las fases teóricas $\varphi = s/r$ del período correspondiente. 65

6.12. Distribuciones de probabilidad medidas en `ibm_torino` para $a = 4$ en el estudio inicial V1. El eje horizontal muestra el resultado de la medición normalizado al intervalo $[0, 1]$. 66

A.1. Ejemplo de DD de una secuencia simple X-X en un circuito sencillo [64]. . . 83

A.2. Principio algebraico del Pauli Twirling. Primera fila: una compuerta de Pauli Λ_P puede absorberse en los operadores envolventes, simplificando el circuito. Segunda fila: una compuerta ideal U envuelta entre pares de Pauli P–P' equivale a U con pares actualizados, preservando la operación lógica. Tercera fila: cuando la compuerta presenta un error coherente $\tilde{U} = U \cdot E$, el canal resultante tras el twirling es equivalente a $\Lambda_P \cdot U$, convirtiendo el error coherente en un factor de Pauli aleatorizado [64]. 85

A.3. Pauli Twirling para las compuertas nativas CNOT (filas superiores) y ECR (filas inferiores) en procesadores IBM Quantum. Cada conjunto de circuitos dentro de una fila es lógicamente equivalente, es decir, produce el mismo efecto ideal sobre los qubits, al satisfacer la condición $P_{\text{después}} \cdot U \cdot P_{\text{antes}} = e^{i\varphi}U$. Los pares de Pauli mostrados (X, Y, Z en los qubits de control y objetivo) constituyen el conjunto finito de combinaciones válidas del que Qiskit Runtime selecciona aleatoriamente en cada instancia del circuito [64]. 85

C.1. Caracterización de los bloques fundamentales del algoritmo de Shor para $N = 15$. Se muestra la profundidad y el conteo de compuertas para la QFT ($n = 9$ qubits) y la exponenciación modular controlada $C\text{-}U_{a^{2^k}}$ para cada base $a \in \{4, 7, 11, 14\}$. La verificación numérica confirma la unitariedad de todos los operadores ($\|U^\dagger U - I\|_F < 10^{-10}$) y la periodicidad $(U_a)^r = I$, validando la corrección de la implementación antes de la ejecución del circuito completo. 97

Glosario

ALAP (*As Late As Possible*): criterio de planificación que retrasa cada operación lo máximo posible dentro del circuito, dejando los huecos de inactividad al inicio para insertar pulsos de protección.

AQFT: versión recortada de la Transformada Cuántica de Fourier en la que se descartan las rotaciones de ángulo más pequeño; reduce la profundidad del circuito a cambio de una pequeña pérdida de precisión.

Backend: dispositivo donde finalmente corre el circuito, ya sea una máquina física en un laboratorio de IBM o un programa que imita su comportamiento.

Basis gates (compuertas nativas): repertorio reducido de operaciones que el procesador sabe ejecutar directamente; toda compuerta del algoritmo se traduce a este conjunto antes de correr.

Bitstring: cadena de ceros y unos que resulta de medir varios qubits a la vez. Cada repetición del experimento produce una de estas cadenas.

Coprimo: se dice de dos enteros que solo comparten el 1 como divisor común; condición que debe cumplir la base a frente a N en el algoritmo de Shor.

Crosstalk: interferencia indeseada entre qubits vecinos: al accionar uno, los de al lado se contaminan ligeramente, alterando la operación pretendida.

Eagle / Heron / Flamingo: sucesivas familias de procesadores superconductores de IBM. Eagle tiene 127 qubits, Heron r1 (el utilizado en este trabajo) 133, y Flamingo es la generación anunciada como siguiente paso.

ECR (*Echoed Cross-Resonance*): compuerta nativa de dos qubits en la familia Eagle de IBM (basada en acoplamiento de cross-resonance); lógicamente equivale a la CNOT salvo por rotaciones locales adicionales. En la familia Heron r1 utilizada en este trabajo, IBM migró a CZ como compuerta nativa de dos qubits.

Fake Backend: simulador local que imita un procesador real cargando una ”fotografía”de

sus parámetros de calibración; permite ensayar el algoritmo sin consumir tiempo de cómputo en la nube.

Heurística: regla práctica que busca una buena solución sin garantizar la óptima, útil cuando explorar todas las posibilidades resultaría inviable en tiempo de cómputo.

Idle time (tiempo de inactividad): intervalo durante el cual el qubit no participa en ninguna compuerta pero sigue expuesto al entorno y, por tanto, a la pérdida de coherencia.

Job / Job ID: envío individual de un circuito al servicio en la nube de IBM. Cada uno recibe un identificador único para recuperar después sus resultados.

Layout: asignación inicial que decide qué qubit lógico del algoritmo se coloca sobre qué qubit físico del chip antes de empezar a ejecutar.

Little-endian: convención de orden en la que el bit menos significativo se escribe primero. Qiskit la adopta, lo que obliga a invertir cadenas al traducir resultados a enteros.

Open Plan: modalidad gratuita de acceso a IBM Quantum, con un cupo mensual limitado de tiempo de procesador real.

Outcome: cada resultado distinto observado al medir el registro. Cuantos más outcomes únicos aparezcan, más dispersa está la probabilidad y peor es la señal del algoritmo.

Pipeline: secuencia ordenada de pasos en la que la salida de cada etapa alimenta a la siguiente; en transpilación describe las fases sucesivas de adaptación del circuito al hardware.

POVM (*Positive Operator-Valued Measure*): formalismo general de medida cuántica, más amplio que las mediciones proyectivas, capaz de describir cualquier procedimiento experimental realista de extracción de información.

Qiskit: entorno de desarrollo en Python creado por IBM para construir, simular y enviar circuitos cuánticos a sus procesadores.

QPU (*Quantum Processing Unit*): chip físico que ejecuta las operaciones cuánticas; análogo de la CPU en computación clásica.

Routing (enrutamiento): etapa de la transpilación que decide cómo mover la información entre qubits no vecinos, insertando SWAPs cuando hace falta.

RSA: criptosistema ampliamente desplegado en Internet cuya seguridad depende de la dificultad clásica de factorizar números muy grandes, problema que el algoritmo de Shor amenaza con resolver rápidamente.

SamplerV2: primitiva del entorno Qiskit que envía circuitos al hardware y devuelve la distribución estadística de los resultados de medición.

SDK (*Software Development Kit*): colección de bibliotecas y herramientas con las que se programa para una plataforma específica; aquí se refiere al kit de Qiskit para hardware de IBM.

Semilla (*seed*): número que inicializa un proceso pseudoaleatorio; fijarla garantiza que repetir el experimento con los mismos datos arroje siempre el mismo resultado.

Shot: cada disparo individual del circuito en el dispositivo. Promediar muchos shots permite reconstruir la distribución estadística de las salidas.

Snapshot: fotografía instantánea de los parámetros de calibración del procesador (tiempos, errores, conectividad) que se carga en los simuladores para imitar su comportamiento real.

Trampas de iones: alternativa tecnológica a los qubits superconductores, en la que átomos cargados se confinan con campos electromagnéticos y se controlan con láseres.

Twirling: técnica genérica de aleatorización que envuelve una operación entre otras escogidas al azar, con el fin de simetrizar el ruido y convertir errores complejos en otros más sencillos de tratar estadísticamente.

Ventaja cuántica: situación en la que un computador cuántico resuelve un problema sustancialmente más rápido que cualquier algoritmo clásico conocido; el caso de Shor frente a la factorización es uno de los ejemplos más espectaculares.

Capítulo 1

Introducción

En 1994, Peter Shor demostró que un computador cuántico ideal podía factorizar enteros en tiempo polinomial [1], abriendo una brecha exponencial frente a los mejores algoritmos clásicos conocidos. Más allá de su impacto teórico sobre la criptografía de clave pública, ese resultado convirtió a la factorización en un banco de pruebas privilegiado para evaluar el progreso del hardware cuántico: cada nueva generación de procesadores se mide, entre otros indicadores, por su capacidad de ejecutar instancias no triviales de este algoritmo. Tres décadas después, la promesa de una ventaja cuántica útil sigue estando condicionada por las características físicas de los dispositivos disponibles [2, 3], lo que vuelve indispensable estudiar el algoritmo no solo en el plano teórico, sino también, y sobre todo, en el contexto del hardware real al que hoy es posible acceder.

El presente trabajo de grado se inscribe en esa dirección y nace de la confluencia de dos circunstancias. Por un lado, la posibilidad efectiva de ejecutar circuitos en procesadores cuánticos superconductores de última generación a través de la plataforma IBM Quantum y su entorno de desarrollo Qiskit, una infraestructura accesible mediante una conexión a internet desde cualquier lugar del mundo. Por otro lado, la escasez de trabajos experimentales en computación cuántica reportados desde la Universidad de Nariño y la región [4], que motiva el desarrollo de una metodología reproducible capaz de servir como referencia para futuras investigaciones en la institución. Conviene precisar desde ahora que la implementación aquí evaluada corresponde a la variante *compilada* del algoritmo de Shor, estándar en la literatura NISQ para casos pedagógicos [5–7]: la exponenciación modular se sintetiza como un operador unitario precomputado clásicamente para cada base a, decisión que se discute en detalle en §5.3 y en las limitaciones del trabajo (§7).

Como caso de estudio se eligió la factorización de $N = 15$, la instancia pedagógica canónica del algoritmo. Aunque pequeña, esta instancia exige circuitos de profundidad considerable y constituye una prueba exigente para los procesadores actuales. Para mitigar los efectos del ruido se evaluaron tres estrategias complementarias: el algoritmo de ruteo SABRE, que reduce heurísticamente la inserción de compuertas SWAP impuestas por la conectividad restringida de la topología *Heavy-Hex* [8]; el Desacoplamiento Dinámico (DD), que reenfoca

la decoherencia de baja frecuencia mediante secuencias de pulsos en los intervalos de inactividad de los qubits [9]; y el Pauli Twirling (PT), que convierte errores coherentes en ruido estocástico promediable sobre múltiples realizaciones del circuito [10].

El diseño experimental se articuló en tres fases progresivamente más realistas. En la primera se verificó la corrección del circuito mediante simulaciones ideales, estableciendo una línea base libre de ruido y caracterizando el sobrecosto introducido por la topología del procesador. En la segunda, los circuitos optimizados se ejecutaron sobre un simulador con modelo de ruido calibrado (*Fake Backend*) basado en el procesador `ibm_torino`. En la tercera, los circuitos validados se enviaron al procesador real `ibm_torino` (familia Heron r1, 133 qubits superconductores) a través del IBM Quantum Open Plan. Los resultados permitieron demostrar experimentalmente la factorización de $N = 15 = 3 \times 5$ sobre hardware real y, de manera no anticipada, identificar un hallazgo metodológico: en el régimen estudiado, la elección del nivel de optimización del transpilador resulta determinante para la viabilidad del algoritmo, mientras que las técnicas de supresión de errores no siempre resultan beneficiosas y pueden incluso degradar la señal cuando se aplican de forma indiscriminada.

El presente documento se organiza en siete capítulos. Tras esta introducción, el **Capítulo 2** formula el planteamiento del problema y la pregunta de investigación que guía el estudio, y el **Capítulo 3** enuncia los objetivos generales y específicos del trabajo. El **Capítulo 4** desarrolla el marco referencial, que incluye los antecedentes de implementaciones previas del algoritmo de Shor en diversas plataformas y un marco teórico que construye el formalismo necesario para comprender el algoritmo, el contexto NISQ, la arquitectura de IBM Quantum, los procesos de transpilación y supresión de errores y las métricas de evaluación empleadas. El **Capítulo 5** expone el diseño metodológico y la implementación experimental en sus tres fases. El **Capítulo 6** presenta los resultados organizados en las mismas tres fases: simulación ideal, simulación con ruido y ejecución en hardware real. Finalmente, el **Capítulo 7** sintetiza las conclusiones del trabajo, identifica sus limitaciones y propone perspectivas futuras para la continuación de esta línea de investigación en la Universidad de Nariño.

Capítulo 2

Planteamiento del problema

Desde hace varias décadas, la comunidad científica ha anticipado una revolución computacional basada en las propiedades de la mecánica cuántica [11, 12]. Los computadores cuánticos explotan fenómenos como la superposición y el entrelazamiento [13, 14] para abordar problemas intratables para la computación clásica, tales como la simulación de nuevos materiales [15], el diseño de fármacos [16] y, notablemente, la factorización de enteros grandes, lo cual representa una amenaza teórica para los sistemas criptográficos de clave pública actuales, como RSA [17, 18]. Sin embargo, la materialización de estas promesas enfrenta un obstáculo físico fundamental: la necesidad de un hardware tolerante a fallos (FTQC). Aunque existen anuncios de procesadores con más de 1000 qubits físicos [3], la industria aún se encuentra en la era de los Dispositivos Cuánticos de Escala Intermedia Ruidosos (NISQ, por sus siglas en inglés) [3, 19]. En este contexto, los qubits son susceptibles a la decoherencia y las compuertas lógicas poseen tasas de error que se acumulan rápidamente con la profundidad del circuito, lo que limita severamente la ejecución de algoritmos complejos como el de Shor.

Esta limitación física resulta particularmente crítica en el algoritmo de Shor, cuya ventaja exponencial teórica exige una profundidad de circuito considerable debido a la exponenciación modular y la Transformada Cuántica de Fourier (QFT) [20]. La ejecución directa de estas rutinas tiende a exceder los tiempos de coherencia (T_1, T_2) de los dispositivos actuales, produciendo salidas dominadas por ruido en lugar del espectro esperado [5, 20]. En el plano experimental, esto obliga a traducir el algoritmo teórico a circuitos físicos eficientes mediante estrategias de compilación, transpilación, mitigación y supresión de errores que respeten la topología del procesador y sus restricciones de ruido [21]. De este modo, la brecha entre la formulación teórica y su ejecución concreta en hardware NISQ no constituye dos problemas independientes, sino la manifestación unificada de una misma causa física.

En el ámbito local, la Universidad de Nariño no dispone de infraestructura cuántica propia, pero el acceso remoto a la nube de IBM [22] habilita la investigación experimental de estas limitaciones. No obstante, persiste un vacío en la comprensión práctica sobre cuánto se degradan estos algoritmos al someterlos a las restricciones físicas reales. El problema central no radica en la mera ejecución del algoritmo, sino en cuantificar la brecha entre la simulación

ideal y la ejecución ruidosa para casos abordables en NISQ (como $N = 15$), determinando si las técnicas actuales de transpilación y supresión de errores permiten obtener resultados distinguibles del ruido.

En este contexto de hardware real pero limitado, surge la siguiente pregunta de investigación: ¿Cuál es el desempeño y la viabilidad técnica de implementar versiones optimizadas para NISQ del algoritmo de Shor para enteros pequeños en procesadores cuánticos de IBM, comparando su fidelidad, profundidad de circuito y probabilidad de éxito frente a simulaciones ideales y modelos de ruido?

Capítulo 3

Objetivos

3.1 Objetivo general

Evaluar la viabilidad técnica y el desempeño de implementaciones educativas y optimizadas del algoritmo de Shor para enteros pequeños ($N = 15$) en procesadores cuánticos NISQ de IBM, mediante un análisis comparativo de fidelidad, profundidad de circuito y probabilidad de éxito entre simuladores con modelos de ruido y la ejecución en hardware real.

3.2 Objetivos específicos

1. Diseñar y caracterizar en Qiskit los bloques fundamentales del algoritmo (Transformada Cuántica de Fourier y Exponenciación Modular), analizando el impacto de estrategias de transpilación avanzadas (como el algoritmo de ruteo SABRE) en la reducción de la profundidad del circuito y el conteo de compuertas CNOT.

2. Implementar estrategias de optimización híbrida mediante pre-procesamiento clásico para identificar los coeficientes a que minimicen teóricamente la complejidad del circuito de exponenciación modular, haciendo viable la ejecución experimental para casos de estudio como $N = 15$ en dispositivos NISQ.

3. Ejecutar las versiones optimizadas del algoritmo utilizando "Fake Backends" (simuladores locales con ruido real) y hardware de IBM, aplicando técnicas de supresión de errores (como *Dynamical Decoupling* y *Pauli Twirling*) para determinar como influyen en la relación señal-ruido de los resultados.

4. Cuantificar la degradación de la señal cuántica comparando los resultados experimentales frente a las simulaciones ideales, utilizando la métrica de *Fidelidad de Hellinger* (Medida de similitud) y *Probabilidad de Éxito* (tasa de obtención del periodo correcto).

Capítulo 4

Marco referencial

4.1 Marco de antecedentes

Desde finales del siglo XIX y durante las primeras décadas del siglo XX, la física experimentó una transformación profunda. Los problemas asociados a la radiación de cuerpo negro, el efecto fotoeléctrico y otros fenómenos condujeron a la formulación de la mecánica cuántica, disciplina que introdujo nociones como la cuantización, la superposición y el principio de incertidumbre [23]. Este desarrollo histórico, junto con las tecnologías instrumentales que desencadenó (tales como el control preciso de estados cuánticos y técnicas de medida coherente), es la base indispensable que hace posible hoy manipular qubits y diseñar algoritmos cuánticos implementables en hardware real [23, 24].

La unidad de información en este nuevo paradigma es el qubit, que puede existir en superposición y entrelazarse con otros, propiedades que habilitan algoritmos con ventajas teóricas sobre sus contrapartes clásicas [24]. En particular, el algoritmo de Shor, propuesto en 1994, demostró que la factorización de enteros (problema central en criptografía asimétrica) puede resolverse en tiempo polinómico sobre un computador cuántico, lo que generó un interés inmediato en la verificación experimental de sus subrutinas críticas: estimación de fase, búsqueda del orden y la transformada cuántica de Fourier (QFT) [1, 24]. No obstante, la distancia entre la formulación matemática y la ejecución en hardware físico real se mantiene condicionada por limitaciones tecnológicas (tasas de error de las compuertas, decoherencia y conectividad), lo que ha orientado la investigación hacia versiones compiladas o reducidas de Shor y hacia esquemas híbridos para sortear las restricciones de los dispositivos NISQ (Noisy Intermediate-Scale Quantum) [25, 26].

Las primeras implementaciones experimentales de Shor y sus bloques constitutivos se realizaron en plataformas diversas, aportando lecciones metodológicas vigentes. La demostración en NMR (resonancia magnética nuclear) que factorizó $N = 15$ validó la factibilidad conceptual de Shor y permitió desarrollar técnicas de control y verificación trasladables a otras tecnologías [25]. En fotónica se llevaron a cabo versiones compiladas que corroboraron la generación de entrelazamiento multipartito y la ejecución de subrutinas de la QFT bajo

condiciones reales [6,27,28], mientras que trabajos con trampas de iones documentaron rutas escalables para implementar operaciones aritméticas modulares y transformadas cuánticas complejas [29]. Estas implementaciones históricas establecieron dos convenciones prácticas esenciales: (i) la necesidad de reducir la profundidad de los circuitos y minimizar operaciones multi-qubit costosas, y (ii) la conveniencia de validar experimentalmente mediante métricas robustas (fidelidad, probabilidades de salida y chequeos de entrelazamiento) y comparaciones con simuladores ideales y ruidosos [6, 25, 27, 29].

La democratización del acceso a QPUs superconductoras a través de servicios en la nube, particularmente la plataforma IBM Quantum y su ecosistema Qiskit, cambió radicalmente el panorama experimental: hoy es factible para equipos de investigadores y estudiantes enviar circuitos a procesadores remotos, ejecutar simuladores con modelos de ruido y ensayar técnicas de supresión y mitigación sin disponer del laboratorio físico [30, 31]. Esta accesibilidad ha impulsado una proliferación de estudios que implementan versiones compiladas o reducidas de Shor directamente en *backends* de IBM, documentando el flujo completo de experimentación (desde el mapeo lógico-físico hasta la aplicación de mitigaciones en lectura y extrapolación a ruido cero) y ofreciendo buenas prácticas para garantizar la reproducibilidad [30, 32, 33]. La evidencia reciente muestra que, con ajustes de compilación, supresión y mitigación apropiados, es posible obtener factorizaciones experimentales reproducibles para valores de N pequeños o moderados en QPUs de IBM, aunque la probabilidad de éxito se degrada con la profundidad y el número de compuertas de dos qubits [5, 26, 29].

Entre las contribuciones más relevantes para la implementación en IBM destaca la demostración de order-finding para $N = 21$ ejecutada sobre QPUs de IBM, la cual empleó una versión compilada de la estimación de fase y compuertas Toffoli aproximadas para reducir la complejidad física, además, el trabajo verificó la presencia de entrelazamiento entre registros, ofreciendo un precedente metodológico claro para validar implementaciones en *backends* reales con recursos limitados [5]. Trabajos más recientes han refinado esta metodología, demostrando que la viabilidad de ejecutar Shor en hardware de IBM (como *Eagle* o *Heron*) depende críticamente de técnicas de pre-procesamiento clásico para seleccionar coeficientes a que generen periodos cortos, y del uso de algoritmos de ruteo estocástico (SABRE) para adaptar el circuito a la topología *Heavy-Hex* del chip [20]. Además, este estudio resalta la importancia de utilizar *Fake Backends* (simuladores con modelos de ruido reales) como paso intermedio de validación para predecir la similitud antes de consumir recursos de hardware [20, 32].

La bibliografía metodológica y los tutoriales oficiales de Qiskit aportan, además, conocimiento operativo imprescindible: describen herramientas de compilación (incluyendo heurísticas para el mapeo de qubits y la reducción de compuertas), técnicas de supresión (*Dynamical Decoupling*, *Pauli Twirling*) o mitigación (corrección de *readout*, extrapolación a ruido cero) y prácticas de registro (anotar la versión del backend, *snapshots* y calibraciones) que permiten comparar de forma rigurosa simulador ideal, simulador con ruido y ejecución en hardware [31–33]. La convergencia de estas herramientas y resultados sugiere una hoja de

ruta experimental consistente: escoger una variante de Shor apta para NISQ, documentar el *backend* y las calibraciones, aplicar protocolos de supresión o mitigación y cuantificar métricas clave (probabilidad de éxito, fidelidad, dependencia con la profundidad) mediante comparaciones controladas entre simuladores y QPUs reales [26, 29, 32]. A pesar de estos avances, la escalabilidad hacia enteros de relevancia práctica sigue limitada por restricciones técnicas (principalmente la imperfección de compuertas de dos qubits y la falta de corrección de errores a gran escala), por lo que la investigación aplicada continúa orientada a mejorar la relación coste-beneficio de las implementaciones y a desarrollar procedimientos reproducibles en la nube [25, 26, 29].

En el ámbito nacional, Colombia ha mostrado señales de un interés institucional creciente hacia tecnologías cuánticas, con la llegada del primer computador cuántico a una universidad colombiana (basado en NMR, de dos qubits por lo que incapaz de ejecutar Shor) [34] y la inclusión de estas tecnologías en convocatorias de Minciencias [35, 36]. Sin embargo, la revisión documental sugiere que en la región de Nariño y en la Universidad de Nariño existe aún una escasez de trabajos experimentales publicados que documenten implementaciones de Shor en QPUs de IBM [4, 37, 38]. Este trabajo se fundamenta en estos antecedentes para llenar dicho vacío, adoptando las metodologías de optimización de circuito y validación en simuladores ruidosos propuestas en la literatura reciente para evaluar el desempeño real del algoritmo en el entorno académico local [4, 20]. El objetivo es aportar evidencia experimental rigurosa y guías prácticas para la comunidad de Pasto y la Universidad de Nariño, aprovechando los recursos formativos y técnicos disponibles junto a la experiencia metodológica acumulada en la literatura [30, 31].

4.2 Fundamentos de la computación cuántica

Esta sección recopila los elementos del formalismo de la computación cuántica que seran necesarios para entender la implementación experimental del algoritmo de Shor sobre el hardware NISQ de IBM Quantum. El detalle adicional sobre el modelo de ruido y la mitigación de errores de lectura se difiere al Apéndice A.

4.2.1 Qubit, notación y matrices de Pauli

La computación cuántica se formula sobre espacios de Hilbert complejos de dimensión finita [24, 39]. En la notación de Dirac, un vector de estado se denota $|\psi\rangle$ (*ket*) y su dual $\langle\psi|$ (*bra*), con producto interno $\langle\phi|\psi\rangle$.

El *bit cuántico* o *qubit* es la unidad fundamental de información cuántica. Su espacio de estados es $\mathcal{H} = \mathbb{C}^2$ y la base ortonormal estándar, denominada *base computacional*, está dada por

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}, \qquad |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}. \tag{4.1}$$

El estado más general de un qubit es una superposición coherente

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \qquad \alpha, \beta \in \mathbb{C}, \qquad |\alpha|^2 + |\beta|^2 = 1, \tag{4.2}$$

donde α y β son las *amplitudes de probabilidad*: al medir en la base computacional, los resultados $|0\rangle$ y $|1\rangle$ se obtienen con probabilidades $|\alpha|^2$ y $|\beta|^2$ respectivamente [24, 39].

Las matrices de Pauli (Figura 4.1), junto con la identidad, generan el espacio de operadores hermíticos sobre $\mathbb{C}^2$ [24] y aparecen sistemáticamente en la descripción de compuertas y canales de ruido.

Para describir el estado de un sistema cuántico de manera general, se introduce el formalismo del *operador densidad* ρ, (matriz densidad si se lo representa en una base ortonormal) [24,40]. Este enfoque unifica la descripción tanto de sistemas con certeza cuántica completa (*estados puros*), donde $\rho = |\psi\rangle\langle\psi|$, como de sistemas que presentan incertidumbre clásica (por ejemplo, debido a la interacción con el entorno). En este último caso, el sistema se encuentra en un *estado mixto*, el cual se representa mediante un ensamble estadístico de estados puros dado por $\rho = \sum_i p_i|\psi_i\rangle\langle\psi_i|$ con $\sum_i p_i = 1$ [24, 40]. Un operador densidad es valido si y solo si, se tiene la traza unitaria, $Tr(\rho) = 1$, la positividad, $\rho \geq 0$ (todos sus autovalores son no negativos). Este formalismo es indispensable para modelar la decoherencia y el ruido en procesadores NISQ (Sección 4.2.5).

4.2.2 Evolución, medición y sistemas compuestos

La evolución de un sistema cuántico cerrado se describe por una transformación unitaria $|\psi'\rangle = U|\psi\rangle$ con $U^\dagger U = I$, lo que garantiza la conservación de la norma; esta evolución unita-

ria se implementa en el modelo de circuitos como la aplicación sucesiva de *compuertas cuánticas*. Las mediciones empleadas en este trabajo son *proyectivas en la base computacional* $\{|0\rangle, |1\rangle\}$. Para sistemas de n qubits, el espacio de estados es el producto tensorial $\mathcal{H}^{\otimes n} = \mathbb{C}^{2^n}$; un estado factorizado de n qubits en un *estado producto* se escribe $|\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle$ [24].

4.2.3 Superposición y entrelazamiento

El principio de superposición es consecuencia directa de la linealidad del espacio de estados: dada una base ortonormal $\{|e_i\rangle\}_{i=1}^{n}$, toda combinación lineal normalizada constituye un estado físico válido [24, 39]. La superposición permite que un registro de n qubits codifique 2^n amplitudes simultáneamente, y es la base del paralelismo cuántico explotado por la transformada cuántica de Fourier en el algoritmo de Shor.

Conceptualmente, el entrelazamiento cuántico representa una manifestación extrema de correlaciones no clásicas, en la cual las propiedades físicas de un subsistema no pueden describirse de manera individual y completamente independiente de las del otro, independientemente de la distancia que los separe. En el marco de la teoría de la información cuántica, este fenómeno no se considera únicamente una propiedad estadística, sino un *recurso* (algo costoso de producir que permite implementar transformaciones valiosas) físico fundamental. Al igual que el tiempo de CPU o la memoria en la computación tradicional, el entrelazamiento es un recurso cuantificable que se consume de forma activa durante la ejecución de tareas lógicas; constituye el ingrediente indispensable para superar las cotas de complejidad clásica y habilitar la ventaja algorítmica en rutinas críticas como la exponenciación modular.

Para un sistema bipartito puro, un estado $|\Psi\rangle \in \mathcal{H}_A \otimes \mathcal{H}_B$ se define como *separable* si puede expresarse estrictamente de la forma $|\Psi\rangle = |\psi_A\rangle \otimes |\phi_B\rangle$, catalogándose como *entrelazado* en caso contrario [24]. No obstante, al considerar las interacciones de un procesador real con su entorno, es indispensable generalizar esta noción mediante el formalismo de matrices de densidad para dar cuenta de los *estados mixtos*. En este escenario estadístico, un estado bipartito descrito por el operador densidad ρ es separable si y solo si puede descomponerse como una combinación convexa de estados producto [41]:

$$\rho = \sum_i p_i \rho_A^{(i)} \otimes \rho_B^{(i)} \tag{4.3}$$

donde $p_i \geq 0$ representa la distribución de probabilidad clásica tal que $\sum_i p_i = 1$. Si la matriz ρ no admite dicha descomposición, se concluye que el sistema se encuentra en un estado mixto entrelazado.

4.2.4 Modelo de circuitos y compuertas

Las operaciones lógicas del algoritmo se implementan mediante *compuertas cuánticas*, operadores unitarios sobre uno o varios qubits [24]. Las compuertas de un solo qubit empleadas

con mayor frecuencia se resumen en la Figura 4.1; entre ellas destaca la *compuerta Hadamard*,

$$H|0\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \qquad H|1\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle), \tag{4.4}$$

que genera superposiciones equitativas a partir de la base computacional y constituye el primer paso del registro de conteo en el algoritmo de Shor.

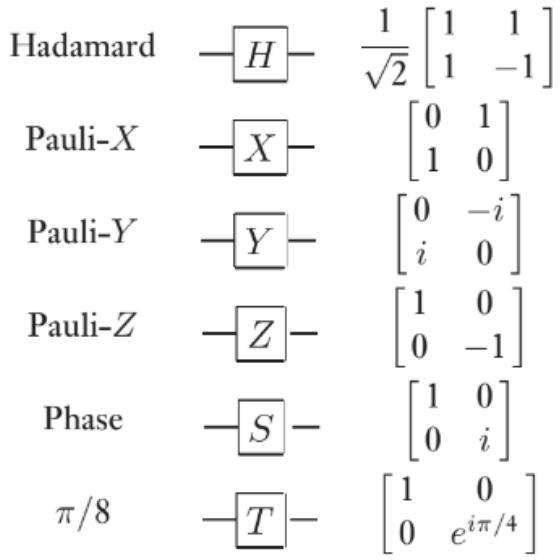

Figura 4.1: Nombres, símbolos y matrices unitarias para las compuertas comunes de un solo qubit [24].

Las *compuertas de rotación de fase* (siendo la S y T de la Figura 4.1 casos particulares para $k = 2$ y $k = 3$ respectivamente), centrales para la construcción de la transformada cuántica de Fourier (QFT), están dadas por

$$R_k = \begin{bmatrix}1 & 0\\ 0 & e^{2\pi i/2^k}\end{bmatrix}. \tag{4.5}$$

La generación de entrelazamiento y la aritmética modular del algoritmo de Shor requieren *compuertas de dos qubits*. La más simple es la CNOT (*Controlled-NOT*), que actúa como

$$\text{CNOT} = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix}, \qquad \text{CNOT}|a,b\rangle = |a, a \oplus b\rangle, \tag{4.6}$$

con $\oplus$ denotando la suma módulo 2 [24, 39]. Para clarificar el efecto de esta compuerta sobre los estados de la base computacional, la Tabla 4.1 presenta su tabla de verdad.

Además de la CNOT, la ejecución de algoritmos en hardware físico demanda el uso de otras compuertas de dos qubits. La compuerta *SWAP* intercambia el estado de dos qubits ($|a,b\rangle \rightarrow$

Entrada $\|a, b\rangle$		**Salida** $\|a, a \oplus b\rangle$	
Control (a)	Objetivo (b)	Control	Objetivo
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Tabla 4.1: Tabla de verdad para la compuerta CNOT, mostrando la inversión del qubit objetivo condicionado al qubit de control.

$|b, a\rangle$) y permite enrutar dinámicamente los circuitos lógicos para que se ajusten a la topología y conectividad física del procesador (tendrá gran relevancia en el algoritmo SABRE, véase la Figura 4.2).

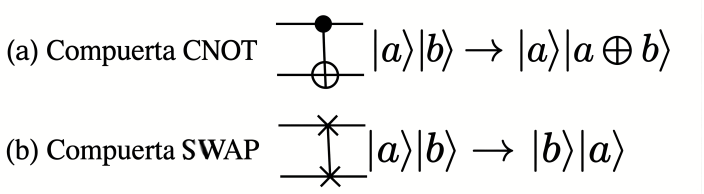

Figura 4.2: Representación de la equivalencia entre compuertas CNOT y SWAP, operaciones fundamentales para el enrutamiento de circuitos mediante el algoritmo SABRE. Fuente: Gemini, Google, (2026).

La CNOT es un caso particular de *compuerta controlada-U*: dada una operación unitaria U sobre un qubit, la operación controlada-U aplica U al qubit objetivo si y solo si el qubit de control está en $|1\rangle$ (Figura 4.3).

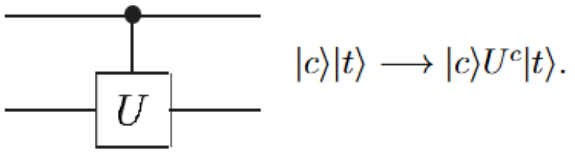

Figura 4.3: Representación de la operación controlada-U [24].

A diferencia del ejemplo simple de un solo qubit objetivo ilustrado en la Figura 4.3, la exponenciación modular $CU_{a,N}$ —el núcleo computacional del algoritmo de Shor— posee un espacio de actuación sustancialmente mayor. Específicamente, $CU_{a,N}$ es un operador unitario que actúa sobre un registro completo de múltiples qubits que representan de manera conjunta el estado modular. En este espacio ampliado, un qubit del registro de conteo actúa como control para condicionar la aplicación de la multiplicación modular $x \pmod{N}$ sobre todo el registro de trabajo, lográndose en la práctica mediante una orquestación sistemática

de múltiples compuertas controladas.

Por otro lado, la *compuerta CZ* (*Controlled-Z*) aplica una fase de -1 al sistema si y solo si ambos qubits se encuentran en el estado $|1\rangle$. Esta última adquiere especial relevancia práctica al ser una compuerta nativa de dos qubits en procesadores de IBM Quantum, como el `ibm_torino` (arquitectura Heron r1).

4.2.5 Ruido NISQ: T_1, T_2, canales de error y errores de lectura

Las subsecciones precedentes asumen evolución unitaria ideal. En el hardware real, los qubits están inevitablemente acoplados a su entorno electromagnético y térmico, fenómeno que produce la *decoherencia* y degrada la información cuántica [2, 24]. En el régimen NISQ, la decoherencia, los errores de compuerta y los errores de lectura constituyen la principal limitación a la profundidad ejecutable de circuitos como el del algoritmo de Shor.

A nivel fenomenológico, el estado del qubit se describe estadísticamente mediante la matriz de densidad ρ. En esta representación, los elementos de la diagonal principal reciben el nombre de *poblaciones* (representando la probabilidad clásica de encontrar al sistema en un estado base particular), mientras que los elementos fuera de la diagonal se denominan *coherencias* (los cuales cuantifican las superposiciones cuánticas puras). Bajo esta base, la pérdida de información de un qubit se caracteriza por dos escalas temporales [2, 24, 42]:

- El *tiempo de relajación* T_1, asociado al decaimiento exponencial de la población del estado excitado $|1\rangle$ hacia el estado fundamental $|0\rangle$, gobernado por $\rho_{11}(t) = \rho_{11}(0)\, e^{-t/T_1}$.
- El *tiempo de decoherencia transversal* T_2, asociado al decaimiento temporal de las coherencias fuera de la diagonal del operador densidad, $\rho_{01}(t) = \rho_{01}(0)\, e^{-t/T_2}$.

Ambos tiempos están intrínsecamente relacionados mediante la tasa total de pérdida de coherencia. Físicamente, el mecanismo de relajación poblacional (caracterizado por $1/T_1$) contribuye de manera directa a la degradación de la coherencia transversal. A esto se le suma el efecto del desfase puro (*pure dephasing*), dictado por el tiempo característico T_ϕ. Esta relación de tasas se expresa como:

$$\frac{1}{T_2} = \frac{1}{2T_1} + \frac{1}{T_\phi}. \tag{4.7}$$

Dado que la tasa de desfase puro obedece a una dinámica estrictamente positiva o nula ($1/T_\phi \geq 0$), se deduce y justifica matemáticamente la cota fundamental:

$$T_2 \leq 2T_1. \tag{4.8}$$

El error introducido por una compuerta imperfecta se modela habitualmente componiendo la operación unitaria ideal con un *canal despolarizante*, que en su forma estándar actúa como

$$\mathcal{E}_{\text{depol}}(\rho) = (1-p)\,\rho + \frac{p}{d}\,I, \tag{4.9}$$

donde d es la dimensión del espacio de Hilbert sobre el cual opera el canal. Aunque para un qubit aislado se tiene $d = 2$, los errores dominantes que limitan la profundidad de los circuitos reportados por IBM provienen de sus compuertas entrelazantes de dos qubits (tales como la CNOT o la *Cross-Resonance* de la CZ). Por consiguiente, el canal despolarizante efectivo que captura la principal fuente de error del hardware asume una dimensión $d = 4$. El parámetro p está directamente relacionado con la tasa de error de compuerta reportada por IBM en sus calibraciones diarias [3, 24]. Un mayor número de compuertas en el circuito se traduce en un canal despolarizante efectivo más severo y, por tanto, en una pérdida de fidelidad medible al final del algoritmo.

Finalmente, la medición proyectiva implementada en el hardware no es ideal: se modela mediante una *matriz de confusión* (o de asignación) [43]

$$\mathcal{A} = \begin{bmatrix} P(0|0) & P(0|1) \\ P(1|0) & P(1|1) \end{bmatrix}, \tag{4.10}$$

donde $P(m|s)$ es la probabilidad de registrar el resultado m cuando el estado real del qubit es $|s\rangle$. Por el principio de conservación de la probabilidad, todo evento de medición debe arrojar un resultado válido, lo cual impone una estricta condición de completitud sobre las columnas de la matriz $\mathcal{A}$:

$$P(0|0) + P(1|0) = 1 \quad \text{y} \quad P(0|1) + P(1|1) = 1. \tag{4.11}$$

La asimetría habitual entre $P(1|0)$ y $P(0|1)$ refleja la asimetría energética entre $|0\rangle$ y $|1\rangle$ en los qubits superconductores [2, 32]. La inversión de $\mathcal{A}$ permite mitigar el error de lectura a posteriori sobre las distribuciones de probabilidad medidas; el procedimiento concreto, junto con la representación de Kraus de los canales y la composición del canal despolarizante con la operación unitaria, se desarrolla en el Apéndice A.

4.3 El algoritmo de Shor

El algoritmo de Shor [1] resuelve, en un computador cuántico ideal y en tiempo polinomial, el problema de factorizar un entero compuesto impar N. Su importancia trasciende el interés algorítmico: la seguridad del criptosistema RSA, ampliamente desplegado en la infraestructura de Internet, descansa sobre la presunta dificultad clásica de este mismo problema. Los mejores algoritmos clásicos conocidos (en particular el *General Number Field Sieve*) requieren un tiempo subexponencial en el número de bits $L = \lceil \log_2 N \rceil$, mientras que el algoritmo de Shor lo logra en $\mathcal{O}(L^3)$ operaciones cuánticas, una aceleración superpolinomial.

La idea central es indirecta. En lugar de atacar la factorización de forma directa, Shor reduce el problema a una tarea relacionada: la *búsqueda de orden* [24]. Dado un entero a coprimo con N, se define el orden r de x módulo N como el menor entero positivo que satisface

$$x^r \equiv 1 \pmod{N}.$$

Una vez conocido r, una manipulación clásica elemental permite (con probabilidad bien acotada inferiormente) obtener un factor no trivial de N a partir del cálculo del máximo común divisor entre $x^{r/2} \pm 1$ y N. La dificultad se traslada entonces a determinar r. Clásicamente esta tarea es exponencial en L, pero cuánticamente admite una solución eficiente porque r es el período de la función $f(z) = x^z$ mód N, y los algoritmos cuánticos pueden extraer periodicidades de funciones evaluadas en superposición mediante la transformada cuántica de Fourier. La Figura 4.4 sintetiza la estructura del algoritmo en tres bloques, cuyos ingredientes se desarrollarán formalmente en las subsecciones siguientes.

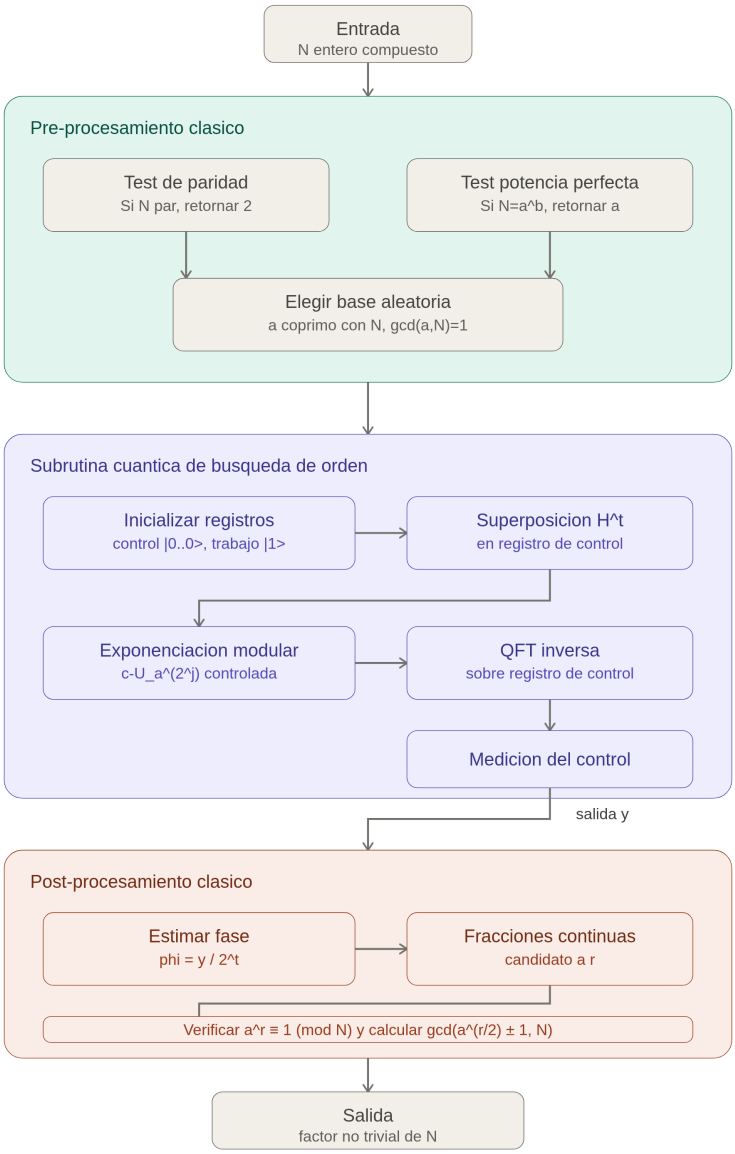

Figura 4.4: Esquema general del algoritmo de Shor (Fuente: creada con Claude, Anthropic, (2026).)

4.3.1 Transformada cuántica de Fourier (QFT)

La transformada cuántica de Fourier (QFT) es el operador lineal que efectúa el cambio de la base computacional a la base de fase (o base de Fourier). Sobre un espacio de Hilbert de dimensión $M = 2^n$, la QFT se define por su acción sobre los estados ortonormales $\{|j\rangle : j = 0, \ldots, M-1\}$ [24]:

$$\text{QFT}|j\rangle = \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} e^{2\pi i\, jk/M} \, |k\rangle. \tag{4.12}$$

A partir de esta definición se obtiene, mediante la representación binaria $j = \sum_{m=1}^{n} j_m 2^{n-m}$ y $k = \sum_{m=1}^{n} k_m 2^{n-m}$ ($j_m, k_m \in \{0, 1\}$) y la definición de la fracción binaria $0.j_l \ldots j_n = \sum_{m=l}^{n} j_m 2^{l-m-1}$, la forma factorizada [24]

$$\text{QFT}|j_1 j_2 \ldots j_n\rangle = \frac{1}{2^{n/2}} \big(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\big) \otimes \big(|0\rangle + e^{2\pi i 0.j_{n-1} j_n}|1\rangle\big) \otimes \cdots \otimes \big(|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n}|1\rangle\big). \tag{4.13}$$

Esta representación revela que el estado de entrada se mapea a fases relativas locales en un estado completamente separable (Figura 4.5), lo que dictamina la factibilidad de sintetizar el operador QFT en hardware mediante compuertas de Hadamard (H) locales y rotaciones de fase controladas $R_k = \text{diag}(1, e^{2\pi i/2^k})$, con compuertas SWAP al final para invertir el orden de los qubits y alinear la salida con la convención estándar del algoritmo [24].

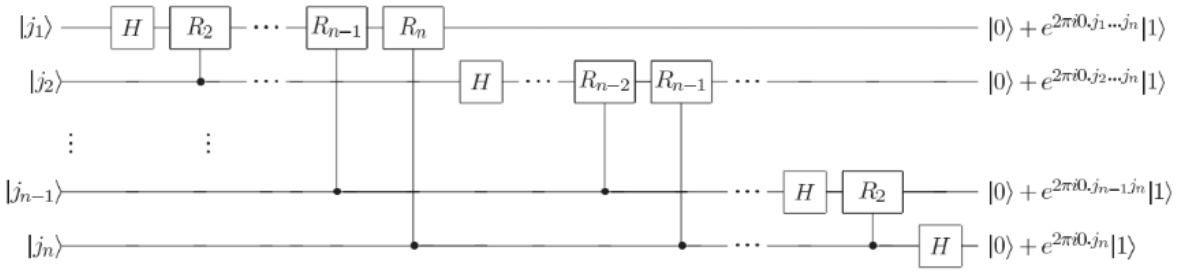

Figura 4.5: Circuito eficiente para la transformada cuántica de Fourier. No se muestran las compuertas de intercambio al final del circuito, las cuales invierten el orden de los cúbits, ni los factores de normalización de $1/\sqrt{2}$ en la salida [24].

Limitaciones de la QFT en el régimen NISQ.

Asintóticamente, la QFT exacta demanda $\mathcal{O}(n^2)$ compuertas. Sin embargo, en la arquitectura NISQ, las compuertas R_k para valores grandes de k inducen rotaciones con ángulos $\theta = 2\pi/2^k$ exponencialmente pequeños. Cuando la magnitud de estas rotaciones cae por debajo de la tasa de error intrínseca de las compuertas de dos qubits, estas operaciones no solo son redundantes, sino perjudiciales, ya que inyectan más ruido por decoherencia térmica que corrección de fase. En el presente trabajo, sin embargo, no se aplicó AQFT mediante truncamiento explícito de las rotaciones R_k; en su lugar, se delegó la aproximación al transpilador de Qiskit cuya función exacta se detalla en el Apéndice B.1. Por tanto, la AQFT se presenta aquí como referencia metodológica de la literatura, no como técnica adoptada por el flujo experimental.

4.3.2 Estimación de fase cuántica (QPE) y cotas de error

La QPE resuelve el problema de extraer el autovalor $e^{2\pi i\varphi}$ asociado a un autovector $|u\rangle$ de un operador unitario U. El circuito emplea dos registros: un registro de conteo de t qubits inicializado en $|0\rangle^{\otimes t}$ y un registro de trabajo que contiene el estado $|u\rangle$ (Figura 4.6).

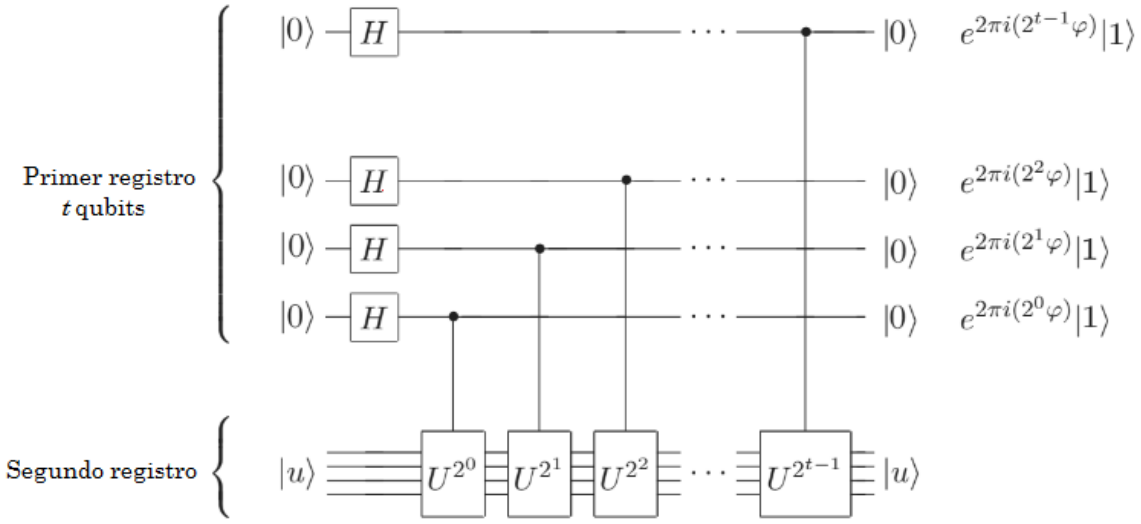

Figura 4.6: Primera etapa del procedimiento de estimación de fase. Se han omitido los factores de normalización de $1/\sqrt{2}$, a la derecha [24].

Tras la aplicación paralela de compuertas Hadamard al registro de conteo y la posterior aplicación de las operaciones controladas $c\text{-}U^{2^j}$, el estado del sistema evoluciona a

$$|\psi_1\rangle = \frac{1}{2^{t/2}} \sum_{k=0}^{2^t-1} e^{2\pi i\varphi k}|k\rangle \otimes |u\rangle. \tag{4.14}$$

Este estado tiene exactamente la forma de una QFT aplicada al estado $|y\rangle$ (compárese con la Ec. 4.12): es decir, $|\psi_1\rangle_{\text{conteo}} = \text{QFT}|y\rangle$ cuando $\varphi = y/2^t$. Por consiguiente, la aplicación de la $\text{QFT}^\dagger$ recupera determinísticamente el estado $|y\rangle$. Esta es la razón fundamental por la cual la estimación de fase funciona: la $\text{QFT}^\dagger$ deshace la codificación de la fase en las amplitudes del registro de conteo y permite leer directamente el valor y que codifica $\varphi = y/2^t$. El circuito completo de la estimación de fase cuántica se muestra en la Figura 4.7.

Si φ puede expresarse exactamente en t bits ($\varphi = y/2^t$), la ecuación anterior garantiza que la medición produce $|y\rangle$ con certeza. Si φ no posee una expansión binaria finita exacta en t bits, el proceso es probabilístico. La estimación de fase requiere acotar matemáticamente el error: para aproximar φ a n bits de precisión con una probabilidad de éxito mínima de $1-\epsilon$, el número de qubits de conteo t requeridos obedece la cota [24]

$$t = n + \left\lceil \log_2\left(2 + \frac{1}{2\epsilon}\right)\right\rceil. \tag{4.15}$$

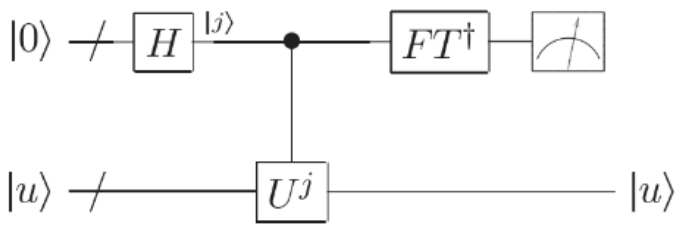

Figura 4.7: Esquema del procedimiento general de estimación de fase. Los t qubits superiores (la barra '/' denota un haz de cables) [24].

En el hardware actual, el número estricto de qubits disponibles y su tiempo de coherencia transversal T_2 impiden satisfacer esta cota para precisiones altas, lo que obliga a emplear técnicas de aproximación y optimización [20].

El procedimiento de la estimación de fase cuántica se resume formalmente en el Algoritmo 1.

Algoritmo 1 Estimación de fase cuántica (QPE) [24]

Entradas: (1) Una caja negra que realiza la operación controlada c-U^j para entero j, (2) un autoestado $|u\rangle$ de U con autovalor $e^{2\pi i\varphi_u}$, y (3) $t = n + \lceil \log(2 + \frac{1}{2\epsilon}) \rceil$ qubits inicializados en $|0\rangle$.

Salidas: Una aproximación de n bits $\widetilde{\varphi}_u$ a φ_u.

Tiempo de ejecución: $\mathcal{O}(t^2)$ operaciones y una llamada a la caja negra c-U^j. Éxito con probabilidad $\geq 1 - \epsilon$.

Procedimiento:

1: *Estado inicial:* $|0\rangle^{\otimes t}|u\rangle$

2: *Crear superposición:* Aplicar $H^{\otimes t}$ al registro de conteo:

$$\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|u\rangle$$

3: *Aplicar caja negra:* Aplicar las operaciones controladas c-U^{2^j}:

$$\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle U^j|u\rangle = \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} e^{2\pi i j\varphi_u}|j\rangle|u\rangle$$

4: *Aplicar QFT inversa:* Aplicar QFT† al registro de conteo:

$\rightarrow |\widetilde{\varphi}_u\rangle|u\rangle$

5: *Medir:* Medir el registro de conteo para obtener $\widetilde{\varphi}_u$.

4.3.3 Búsqueda de orden y exponenciación modular

Para enteros positivos x y N, $x < N$, coprimos, el orden r de x módulo N es el menor entero positivo tal que $x^r \equiv 1 \pmod{N}$. El problema de búsqueda de orden consiste en determinar

r para un par (x, N) dado. Es un problema difícil en un computador clásico, en el sentido de que no se conoce un algoritmo que lo resuelva usando recursos polinomiales en $L = \lceil \log_2 N \rceil$.

El algoritmo cuántico de búsqueda de orden es el algoritmo de estimación de fase aplicado al operador unitario de multiplicación modular sobre el registro de trabajo [24]:

$$U_{x,N}|y\rangle \equiv |\, xy \pmod{N}\,\rangle, \qquad y \in \{0, 1, \ldots, N-1\}. \tag{4.16}$$

Los estados definidos por

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i s k/r} |x^k \bmod N\rangle, \qquad s \in \{0, 1, \ldots, r-1\}, \tag{4.17}$$

son autovectores de $U_{x,N}$ con autovalores $\lambda_s = e^{2\pi i s/r}$ [24].

La preparación directa de los $|u_s\rangle$ requeriría conocer r a priori, precisamente la cantidad que se desea determinar. Esta dificultad se evita mediante una propiedad estructural: el estado computacional $|1\rangle$ (entendido como la codificación binaria de L bits del número 1), que es el estado inicial natural del registro de trabajo, admite la descomposición

$$|1\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle,$$

es decir, $|1\rangle$ es una superposición uniforme de todos los autovectores $|u_s\rangle$ [24]. Como consecuencia, al preparar el registro de trabajo en $|1\rangle$ y ejecutar la QPE con $t = 2L + 1 + \lceil \log(2 + 1/2\epsilon) \rceil$ qubits en el registro de conteo, el algoritmo realiza la estimación de fase simultáneamente sobre todos los autovectores $|u_s\rangle$. Tras la medición se obtiene una estimación de la fase $\varphi \approx s/r$ con precisión de $2L+1$ bits, para un s aleatorio uniformemente distribuido y con probabilidad de al menos $(1-\epsilon)/r$ [24]. El esquema completo se representa en la Figura 4.8.

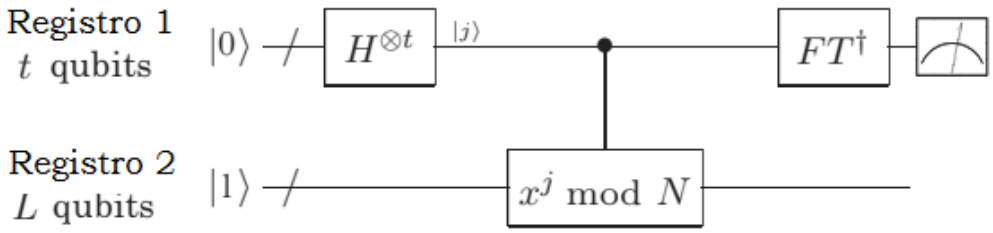

Figura 4.8: Circuito cuántico para el algoritmo de búsqueda de orden. Este circuito también se puede utilizar para la factorización [24].

El cuello de botella: exponenciación modular

La viabilidad técnica del algoritmo de Shor descansa en la implementación eficiente de la secuencia de operaciones controladas $c\text{-}U_{x,N}^{2^j}$ que actúan durante la estimación de fase. Su efecto combinado equivale a multiplicar el contenido del segundo registro por la exponencial modular $x^z \pmod N$, donde z es el contenido del primer registro. El desarrollo detallado de esta estrategia, incluyendo los casos en que la búsqueda de orden falla, se presenta en el Apéndice A.3.

Origen de la ventaja cuántica

La exponenciación modular por sí sola no es más rápida en un computador cuántico que en uno clásico: ambos la realizan en $\mathcal{O}(L^3)$ operaciones. La ventaja cuántica radica en cómo se utiliza esta operación.

- Clásicamente, hallar r requiere evaluar $x, x^2, x^3, \ldots$ de forma secuencial hasta encontrar $x^r \equiv 1 \pmod N$, lo que en el peor caso requiere $\mathcal{O}(r)$ evaluaciones, exponencial en L.

- Cuánticamente, gracias a la superposición, el circuito evalúa $x^z \pmod N$ para todos los valores de z simultáneamente en una sola ejecución; la $\text{QFT}^\dagger$ extrae la periodicidad r de las fases globales del estado resultante sin enumerar caso por caso. Esta es la esencia de la aceleración exponencial del algoritmo de Shor.

Fracciones continuas

La extracción de r a partir de la fase medida se garantiza mediante la teoría clásica de fracciones continuas:

Teorema 1 (Convergencia de fracciones continuas)**.** *Supóngase que s/r es un número racional tal que*

$$\left|\frac{s}{r} - x\right| \leq \frac{1}{2r^2}. \tag{4.18}$$

Entonces s/r es un convergente de la expansión en fracción continua de x.

Aplicado a la medición $x = y/2^t$, este teorema implica que si el número de qubits de conteo cumple $t = 2L + 1$, la condición de convergencia se satisface y r puede hallarse en tiempo $\mathcal{O}(L^3)$. La descripción detallada del procedimiento de extracción (expansión en fracción continua, recursión de los convergentes (Ecs. A.5 y A.6) y verificación clásica del candidato) se presenta en el Apéndice A.4 y se retoma en el contexto del trabajo experimental en Sección 5.7.

El procedimiento completo de la búsqueda cuántica de orden se formaliza en el Algoritmo 2.

Algoritmo 2 Búsqueda cuántica de orden [24]

Entradas: (1) Una caja negra $U_{x,N}$ que realiza la transformación $|j\rangle|k\rangle \to |j\rangle|x^j k \text{ mód } N\rangle$ para x coprimo con el número de L bits N, (2) $t = 2L + 1 + \lceil \log(2 + \frac{1}{2\epsilon}) \rceil$ qubits inicializados en $|0\rangle$, y (3) L qubits inicializados en el estado $|1\rangle$.

Salidas: El menor entero $r > 0$ tal que $x^r \equiv 1 \pmod{N}$.

Tiempo de ejecución: $\mathcal{O}(L^3)$ operaciones. Éxito con probabilidad $\mathcal{O}(1)$.

Procedimiento:

1: *Estado inicial:* $|0\rangle^{\otimes t}|1\rangle$ — *estado inicial*

2: *Crear superposición:*

$$\to \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|1\rangle$$ — *crear superposición*

3: *Aplicar* $U_{x,N}$*:*

$$\to \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|x^j \text{ mód } N\rangle$$ — *aplicar* $U_{x,N}$

$$\approx \frac{1}{\sqrt{r\,2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} e^{2\pi i s j/r} |j\rangle|u_s\rangle$$ — *expandir en autovectores*

4: *Aplicar QFT inversa al primer registro:*

$$\to \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\widetilde{s/r}\rangle|u_s\rangle$$ — *aplicar* $\text{QFT}^\dagger$

5: *Medir el primer registro:*

$$\to \widetilde{s/r}$$ — *medir primer registro*

6: *Aplicar algoritmo de fracciones continuas* a $\widetilde{s/r}$ para extraer r.

$$\to r$$ — *fracciones continuas*

Implicaciones para la era NISQ

En implementaciones pedagógicas orientadas a la evaluación en plataformas NISQ como IBM Quantum, el caso $N = 15$ es la cota menor práctica actual. Incluso para este caso, las implementaciones estándar de la exponenciación modular son transpiladas a miles de compuertas CNOT, aniquilando la fidelidad del estado mucho antes de la ejecución de la $\text{QFT}^\dagger$. Por ende, en este trabajo se emplean arquitecturas altamente optimizadas con conocimiento previo del estado de la subrutina (compilación dependiente de a) [20], minimizando el número de compuertas de dos qubits para habilitar el estudio experimental de fidelidad distinguible del ruido térmico.

4.3.4 Reducción clásica: de la factorización a la búsqueda de orden

Una vez establecidos los bloques fundamentales que intervienen en el algoritmo de Shor, queda por establecer la conexión entre la factorización prima de un entero compuesto y

dicha maquinaria. El problema de factorización es equivalente al problema de búsqueda de orden, en el sentido de que un algoritmo polinomial para la búsqueda de orden se transforma directamente en un algoritmo polinomial para la factorización [24].

La reducción procede en dos pasos. Primero, se demuestra que es posible calcular un factor de N a partir de cualquier solución no trivial $x \neq \pm 1 \pmod{N}$ de la ecuación $x^2 \equiv 1 \pmod{N}$. Segundo, se demuestra que un y elegido al azar y coprimo con N tiene, con probabilidad acotada inferiormente, un orden r par que satisface $y^{r/2} \neq \pm 1 \pmod{N}$, de modo que $x \equiv y^{r/2} \pmod{N}$ es una solución no trivial de la ecuación anterior. Estos resultados se formalizan en los siguientes teoremas [24].

Teorema 2 (Reducción de factorización a búsqueda de orden)**.** *Supóngase que N es un número compuesto de L bits, y x es una solución no trivial a la ecuación $x^2 = 1 \pmod{N}$ en el rango $1 \leq x \leq N$, es decir, ni $x = 1 \pmod{N}$ ni $x = N - 1 \equiv -1 \pmod{N}$. Entonces al menos uno de*

$$\gcd(x - 1, N) \quad y \quad \gcd(x + 1, N), \tag{4.19}$$

es un factor no trivial de N que puede ser calculado utilizando $\mathcal{O}(L^3)$ operaciones.

La demostración se apoya en la identidad $(y^{r/2} - 1)(y^{r/2} + 1) \equiv 0 \pmod{N}$, que implica que N divide al producto. Si ninguna de las dos diferencias es múltiplo de N (es decir, $y^{r/2} \not\equiv \pm 1 \pmod{N}$), entonces cada una debe compartir un factor primo con N. El cálculo del máximo común divisor (gcd) se realiza eficientemente con el algoritmo de Euclides en tiempo polinomial.

Teorema 3 (Probabilidad de la factorización)**.** *Supóngase que $N = p_1^{\alpha_1} \dots p_m^{\alpha_m}$ es la factorización prima de un entero positivo compuesto impar. Sea x un entero elegido uniformemente al azar, sujeto a los requisitos de que $1 \leq x \leq N - 1$ y x es coprimo con N. Sea r el orden de x módulo N. Entonces*

$$p(r \text{ es par y } x^{r/2} \neq -1 \pmod{N}) \geq 1 - \frac{1}{2^m}. \tag{4.20}$$

Los Teoremas 2 y 3 se combinan para producir un algoritmo que, con alta probabilidad, devuelve un factor no trivial de cualquier número compuesto N. Todos los pasos pueden realizarse de manera eficiente en una computadora clásica salvo (por lo que se sabe hoy en día) la subrutina de búsqueda de orden, que clásicamente exige recursos exponenciales. Repitiendo el procedimiento se obtiene una factorización prima completa de N [24].

El procedimiento completo del algoritmo de Shor se formaliza en el Algoritmo 3.

Algoritmo 3 Reducción de la factorización a la búsqueda de orden [24]

Entradas: Un número compuesto N

Salidas: Un factor no trivial de N

Tiempo de ejecución: $\mathcal{O}((\log N)^3)$ operaciones. Éxito con probabilidad $\mathcal{O}(1)$.

1: **if** N es par **then**
2: **return** el factor 2
3: **end if**
4: Determinar si $N = a^b$ para enteros $a \geq 1$, $b \geq 2$. Si es así, **return** el factor a.
5: Elegir aleatoriamente x en el rango $1 \leq x \leq N - 1$.
6: **if** $\gcd(x, N) > 1$ **then**
7: **return** el factor $\gcd(x, N)$
8: **end if**
9: Usar la **subrutina cuántica de búsqueda de orden** (Algoritmo 2) para hallar el orden r de x módulo N.
10: **if** r es par **y** $x^{r/2} \not\equiv -1 \pmod{N}$ **then**
11: Calcular $\gcd(x^{r/2} - 1, N)$ y $\gcd(x^{r/2} + 1, N)$.
12: **return** el factor no trivial encontrado.
13: **else**
14: El algoritmo falla. Repetir desde el paso 9.
15: **end if**

Los pasos 1-3 y 4 del algoritmo devuelven un factor o, de lo contrario, aseguran que N es un entero impar con más de un factor primo. Estos pasos se realizan en $\mathcal{O}(1)$ y $\mathcal{O}(L^3)$ operaciones, respectivamente. Los pasos 5-8 devuelven un factor o producen un elemento x de $\{0, 1, 2, \ldots, N-1\}$ elegido al azar. El paso 9 llama a la subrutina de búsqueda de orden, calculando el orden r de x módulo N. El paso 10 completa el algoritmo: el Teorema 3 garantiza que con probabilidad de al menos un medio r es par y $x^{r/2} \neq -1 \pmod{N}$, y entonces el Teorema 2 garantiza que $\gcd(x^{r/2} - 1, N)$ o $\gcd(x^{r/2} + 1, N)$ es un factor no trivial de N.

4.3.5 Ejemplo guiado: factorización de $N = 15$

Para aterrizar el formalismo previo en el caso de estudio del trabajo, se recorre a continuación el Algoritmo 3 aplicado a $N = 15$, identificando explícitamente los valores numéricos en cada paso.

1. *Comprobar si N es par:* como $N = 15$ es impar, el algoritmo no termina aquí.

2. *Comprobar si N es una potencia perfecta:* se examinan las posibles representaciones $N = a^b$ con $a \geq 2$ y $b \geq 2$. Para $b = 2$, $\sqrt{15} \approx 3{,}87 \notin \mathbb{Z}$; para $b = 3$, $\sqrt[3]{15} \approx 2{,}47 \notin \mathbb{Z}$; valores mayores de b requerirían $a < 2$. Por lo tanto, $N = 15$ no es potencia perfecta.

3. *Elegir una base x coprima con N:* el grupo multiplicativo $\mathbb{Z}_{15}^*$ está formado por los enteros del intervalo $[1, 14]$ coprimos con 15:

$$\mathbb{Z}_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}.$$

   La base $x = 1$ es trivial (orden $r = 1$, no aprovechable); las demás constituyen candidatos válidos. En este trabajo se evalúan experimentalmente las bases representativas $x \in \{4, 7, 11, 14\}$. Para fijar el ejemplo, se toma $x = 7$.

4. *Verificar coprimalidad:* como $\gcd(7, 15) = 1$, no se obtiene un factor por esta vía y se procede a la subrutina cuántica.

5. *Subrutina cuántica de búsqueda de orden:* el registro de trabajo se inicializa en $|1\rangle$, el registro de conteo en $|0\rangle^{\otimes t}$ con $t = 2L + 1 = 9$ (puesto que $L = \lceil \log_2 15 \rceil = 4$), y se ejecuta el Algoritmo 2 con el operador $U_{7,15}$. La subrutina devuelve, tras el post-procesamiento por fracciones continuas, el orden $r = 4$. Esto se verifica calculando explícitamente las potencias de 7 módulo 15:

$$\begin{aligned}
7^1 &\equiv 7 \pmod{15},\\
7^2 &= 49 \equiv 4 \pmod{15},\\
7^3 &= 7 \cdot 4 = 28 \equiv 13 \pmod{15},\\
7^4 &= 7 \cdot 13 = 91 = 6 \cdot 15 + 1 \equiv 1 \pmod{15}.
\end{aligned}$$

   El menor entero positivo que satisface $7^r \equiv 1 \pmod{15}$ es, en efecto, $r = 4$.

6. *Comprobar las condiciones del Teorema 2:* el orden $r = 4$ es par. Se evalúa entonces $7^{r/2} = 7^2 \equiv 4 \pmod{15}$. Como $4 \not\equiv -1 \equiv 14 \pmod{15}$, se cumple la condición $x^{r/2} \not\equiv -1 \pmod{N}$ y la rama exitosa del algoritmo se activa.

7. *Calcular los factores:* se evalúan los máximos comunes divisores

$$\begin{aligned}
\gcd(7^{r/2} - 1,\ 15) &= \gcd(48, 15) = 3,\\
\gcd(7^{r/2} + 1,\ 15) &= \gcd(50, 15) = 5.
\end{aligned}$$

   Ambos resultan ser factores no triviales de N, lo que conduce a la factorización

$$15 = 3 \times 5.$$

Este recorrido ilustra el flujo abstracto del algoritmo. La traducción de la subrutina cuántica del Paso 5 a un circuito ejecutable en hardware IBM Quantum, incluyendo la compilación dependiente de la base a, la elección del número efectivo de qubits y la estructura de la QFT, se desarrolla en el Capítulo 5.

4.4 El contexto NISQ y la arquitectura de IBM Quantum

Antes de abordar las estrategias de transpilación y supresión de errores, es imprescindible comprender el entorno de hardware en el que se ejecutarán los circuitos del algoritmo de Shor. Esta sección presenta un panorama del régimen computacional actual, la plataforma de hardware empleada en este trabajo y las restricciones físicas que determinan la viabilidad de los experimentos.

4.4.1 La era NISQ: qubits físicos frente a qubits lógicos

El término *Noisy Intermediate-Scale Quantum* (NISQ) fue introducido por Preskill [2] para designar la generación actual de procesadores cuánticos, caracterizada por dos rasgos definitorios: (i) un número intermedio de qubits (de decenas a unos pocos miles) y (ii) la *ausencia de corrección cuántica de errores* (QEC) a escala completa. En este régimen, cada qubit que opera en el chip es un *qubit físico* expuesto directamente al ruido del entorno, en contraste con el *qubit lógico* de la computación cuántica tolerante a fallos (*Fault-Tolerant Quantum Computing*, FTQC), donde la información se codifica redundantemente en cientos o miles de qubits físicos mediante códigos correctores de errores [26].

Las limitaciones centrales de los dispositivos NISQ pueden resumirse en tres ejes que operan simultáneamente [2, 44]:

1. *Decoherencia y tiempos de coherencia finitos:* como se formalizó en la Sección 4.2.5, los qubits superconductores poseen tiempos de relajación T_1 y de desfase T_2 del orden de decenas a cientos de microsegundos [2, 3, 45]. Esto impone un límite superior estricto a la profundidad del circuito: una vez que el tiempo de ejecución total del circuito transpilado se aproxima a T_2, la señal cuántica se degrada irreversiblemente [2].

2. *Errores intrínsecos de las compuertas:* las compuertas de un qubit alcanzan fidelidades $\gtrsim 99{,}9\,\%$ en los procesadores actuales de IBM [3, 46], mientras que las compuertas de dos qubits nativas (ECR en la familia Eagle previa y CZ en Heron r1, esta última implementada mediante un acoplador *tunable*) [47] presentan tasas de error típicas del orden de $\epsilon_{2q} \sim 10^{-3}$ [3, 32, 45]. El error global del circuito se acumula multiplicativamente: para un circuito con N_g compuertas de dos qubits, la fidelidad del estado de salida decae aproximadamente como $(1 - \epsilon_{2q})^{N_g}$, lo que hace que circuitos con unos pocos cientos de compuertas de dos qubits produzcan distribuciones de medición esencialmente indistinguibles del ruido [48]. Este comportamiento multiplicativo ha sido confirmado tanto en contextos de síntesis de circuitos [48] como en modelos analíticos de acumulación de errores [49].

3. *Errores de lectura (readout):* la medición de cada qubit también introduce errores asimétricos modelados por la matriz de confusión (Ec. 4.10), con tasas típicas de 1–5 % por qubit [32].

4.4.2 Arquitectura de IBM: qubits superconductores transmon

Los procesadores de IBM Quantum se basan en qubits superconductores de tipo *transmon* (*transmission-line shunted plasma oscillation qubit*), que constituyen la plataforma de hardware más extendida en la industria cuántica actual [3].

El principio físico fundamental es la cuantización de la carga y el flujo magnético en un circuito superconductor que incorpora una *unión Josephson*: un elemento no lineal cuya energía depende de la diferencia de fase superconductora a través de la unión. Esta no linealidad permite que los dos niveles de energía más bajos del circuito ($|0\rangle$ y $|1\rangle$) estén separados por una frecuencia de transición ω_{01} distinta de las transiciones superiores, de modo que el sistema se comporta efectivamente como un sistema de dos niveles (qubit), a diferencia de un oscilador armónico lineal donde todos los niveles están igualmente espaciados [3, 24].

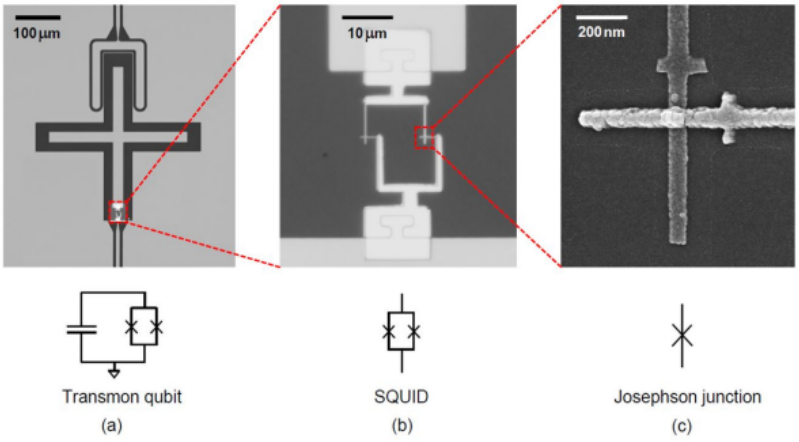

Figura 4.9: Imágenes de un cúbit transmon sintonizable en frecuencia. (a) El transmon completo consta de una capacitancia grande (en forma de ”+”) en paralelo con un SQUID conectado a tierra. En (b) y (c) se muestra un acercamiento del SQUID y una unión Josephson, respectivamente. [50]

Las operaciones (compuertas) sobre los qubits transmon se realizan mediante *pulsos de microondas* calibrados a la frecuencia de resonancia de cada qubit ($\omega_{01} \sim 4$–5 GHz) [3]. Una rotación arbitraria de un qubit se logra modulando la amplitud, la fase y la duración del pulso. El conjunto de compuertas nativas (*basis gates*) varía con la familia del procesador. En la familia Heron r1 (utilizada en este trabajo) se migró a la compuerta CZ como nativa de dos qubits, mientras que las compuertas nativas de un qubit incluyen $RZ(\lambda)$, SX y X, además de instrucciones de control como *ID*, *DELAY* y *MEASURE* [51].

4.4.3 Topología Heavy-Hex y el cuello de botella de la conectividad

En un procesador cuántico ideal con conectividad *all-to-all*, cualquier compuerta de dos qubits podría aplicarse directamente entre dos qubits cualesquiera del chip. Sin embargo, los procesadores superconductores de IBM emplean una topología de conectividad restringida denominada *Heavy-Hex* [3, 51]. En esta arquitectura, los qubits se disponen en los vértices y aristas de una red hexagonal pesada (*heavy hexagonal lattice*), donde cada qubit está acoplado a un máximo de dos o tres vecinos según su posición en la red. Los procesadores de la familia Eagle (127 qubits), Heron (133 qubits) y los más recientes de la hoja de ruta de IBM utilizan variantes de esta topología [3].

La consecuencia operativa más importante de esta conectividad restringida es la necesidad de insertar *compuertas SWAP* durante la transpilación. Cuando el circuito lógico requiere una compuerta de dos qubits entre qubits que no son vecinos físicos en la topología, el transpilador debe mover la información cuántica a través de la red insertando operaciones SWAP. En la familia Heron r1, donde la compuerta entrelazante nativa es CZ, una operación SWAP se descompone en tres compuertas CZ intercaladas con rotaciones locales de un qubit, lo que *triplica aproximadamente* el número de compuertas entrelazantes ruidosas por cada SWAP insertado [20, 51, 52]. Este fenómeno constituye el *principal cuello de botella físico* para la ejecución de algoritmos profundos como Shor: un circuito lógico con N_{2Q} compuertas de dos qubits puede inflarse a aproximadamente $3N_{\text{SWAP}} + N_{2Q}$ compuertas de dos qubits tras la transpilación, donde N_{SWAP} depende de la distancia topológica entre los qubits involucrados y de la calidad del algoritmo de ruteo empleado.

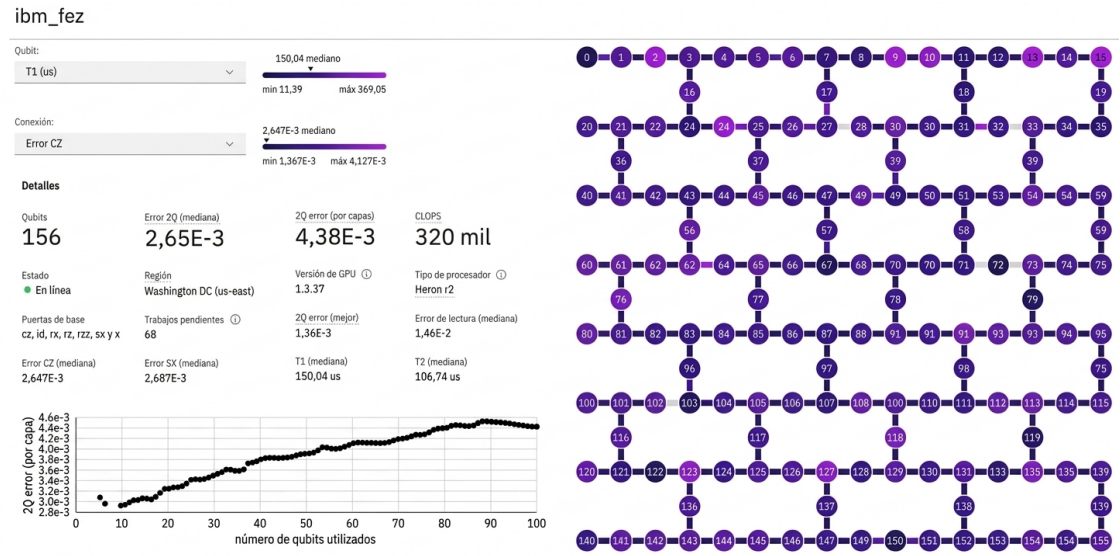

Figura 4.10: Topología Heavy-Hex del procesador cuántico IBM *ibm_fez* (panel derecho) de la familia Heron r2 con 156 qubits. A la izquierda los datos de calibración mas importantes [19]. No se muestra el procesador *ibm_torino* usado en el trabajo ya que fue retirado de los recursos de IBM.

4.5 Transpilación y algoritmo SABRE

La ejecución exitosa de algoritmos cuánticos profundos, como el algoritmo de Shor, en arquitecturas NISQ requiere una adaptación exhaustiva del circuito lógico a las restricciones físicas del hardware y la aplicación de técnicas simultáneas de supresión y mitigación de ruido. El marco de desarrollo Qiskit proporciona las abstracciones de transpilación, supresión y mitigación que posibilitan la transición de la teoría a la experimentación real [31, 51]. A continuación, se detalla el formalismo de estas estrategias que fueron implementadas sobre los procesadores IBM.

4.5.1 El proceso de transpilación y niveles de optimización

La ejecución de algoritmos cuánticos en hardware real de IBM Quantum, requiere la transformación de una representación de alto nivel y abstracta del circuito a un conjunto de instrucciones físicas que respete las limitaciones del *backend* objetivo. Este proceso es llevado a cabo por el transpilador cuántico (transpiler), una herramienta fundamental que actúa como compilador, adaptando la descripción lógica del algoritmo a la arquitectura física subyacente [53]. Dicha adaptación es indispensable para sortear restricciones como la conectividad limitada entre qubits (topología del chip), el conjunto nativo de compuertas (*basis gates*) y las características de calibración específicas de cada dispositivo [54].

El proceso de transpilación en Qiskit se estructura en un *pipeline* compuesto por seis etapas secuenciales, cada una con un propósito específico para transformar el circuito de entrada en uno optimizado y compatible con el hardware [55]. Estas etapas son:

1. *Inicialización (init):* configura el transpilador con los parámetros iniciales y las propiedades del *backend* objetivo.

2. *Asignación lógica a física (layout):* establece un mapeo inicial entre los qubits virtuales (lógicos) del circuito y los qubits físicos del dispositivo. Este mapeo busca, en sus etapas iniciales, minimizar la distancia para futuras operaciones.

3. *Enrutamiento (routing):* dado que las compuertas de dos qubits solo pueden ejecutarse en pares físicamente conectados, esta etapa inserta compuertas SWAP para enrutar los qubits lógicos a posiciones donde las compuertas no locales puedan ser aplicadas.

4. *Traducción (translation):* convierte todas las compuertas del circuito a un conjunto de compuertas base (*basis gates*) que son físicamente realizables por el *backend*, como las mencionadas (`rz`, `sx`, `x`, `cz`).

5. *Optimización (optimization):* aplica una serie de transformaciones para reducir el número de compuertas, la profundidad del circuito y otras métricas, sin alterar la función unitaria del algoritmo.

6. *Programación (scheduling):* asigna los tiempos de ejecución de cada instrucción en el hardware, considerando efectos como la calibración dinámica (e.g., compuertas RZ virtuales) y los tiempos de relajación de los qubits.

Para controlar el balance entre el tiempo de compilación y la calidad de la optimización, el transpilador ofrece cuatro niveles de optimización, configurados mediante el parámetro `optimization_level`, cuyo valor puede ser 0, 1, 2 o 3 [56]. Cada nivel activa un conjunto diferente de rutinas de optimización, impactando significativamente en el número final de compuertas, la profundidad del circuito y el tiempo de compilación. En la Tabla 4.2 se presenta un resumen comparativo de estos niveles.

Tabla 4.2: Niveles de optimización en el transpilador de Qiskit [53].

Nivel	**Transformaciones activadas**	**Número-compuertas**	**Profundidad**	**Tiempo-compilación**
0	Solo etapas de traducción y enrutamiento mínimos. No se aplican optimizaciones de reducción.	Alto	Alta	Mínimo
1	Optimizaciones ligeras: cancelación de compuertas adyacentes (e.g., `Not` seguida de `Not`), y plegado de compuertas.	Reducción moderada	Reducción moderada	Bajo
2	Optimizaciones medias: incluye las del nivel 1, más consolidación de bloques unitarios (`ConsolidateBlocks`) y cancelación conmutativa (`CommutativeCancellation`). Re-síntesis de bloques para usar compuertas nativas más eficientes.	Reducción significativa	Reducción significativa	Moderado
3	Optimizaciones exhaustivas: incluye todas las del nivel 2, junto con técnicas de re-enrutamiento y optimización dependientes del ruido (*noise-adaptive*) como `SABRE` con parámetros más agresivos.	Reducción máxima	Reducción máxima	Alto

Dentro de las etapas de asignación (*layout*) y enrutamiento (*routing*), el algoritmo SABRE juega un papel crucial, especialmente en los niveles de optimización más altos. Su eficacia, como se verá en la siguiente sección, radica en la capacidad de realizar un *layout* inicial adaptativo y un proceso de enrutamiento que considera las calibraciones en tiempo real del dispositivo, buscando un equilibrio entre la inserción de compuertas SWAP y la fidelidad de

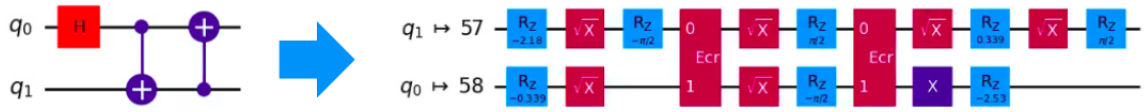

Figura 4.11: Ejemplo ilustrativo de transpilación de un circuito cuántico simple en Qiskit para un procesador IBM con topología Heavy-Hex [53]. A la izquierda se muestra el circuito lógico original. A la derecha aparece el circuito transpilado, donde se han insertado compuertas nativas del hardware y se han mapeado los qubits lógicos q_0 y q_1 a los qubits físicos 58 y 57, respectivamente. Nótese cómo incluso un circuito sencillo requiere varias compuertas de dos qubits (*ECR* en Eagle) y rotaciones adicionales para adaptarse a la conectividad restringida del procesador.

la ejecución [57].

En resumen, la transpilación es un proceso multi-etapa que adapta un algoritmo cuántico abstracto a las realidades físicas de un dispositivo NISQ. Los niveles de optimización de Qiskit proporcionan un mecanismo de control fino sobre el balance entre el esfuerzo computacional de compilación y la calidad del circuito resultante.

4.5.2 Algoritmo SABRE

En los procesadores superconductores de IBM, una compuerta de dos qubits sólo es físicamente realizable entre qubits unidos por una arista del grafo de acoplamiento del chip [58]. Esta restricción geométrica fractura la correspondencia entre el circuito lógico y el circuito físico: dos qubits que el algoritmo trata como vecinos en el espacio de Hilbert pueden estar separados por varios sitios sobre la red *Heavy-Hex*. Cuando esto ocurre, el transpilador inserta compuertas SWAP para acercar los estados, y cada SWAP se descompone a su vez en varias compuertas entrelazantes nativas, amplificando simultáneamente la profundidad del circuito y la probabilidad acumulada de error [8].

El algoritmo SABRE (*SWAP-based BidiRectional heuristic search*) [8] aborda este problema renunciando a una optimización global del mapeo y eligiendo, en cada paso, el SWAP más conveniente mediante una heurística local. Para ello representa el hardware como un grafo de conectividad y precalcula la distancia topológica entre cada par de qubits físicos, una métrica que estima el número mínimo de intercambios necesarios para que dos estados puedan interactuar coherentemente. A continuación, organiza las compuertas entrelazantes del circuito según sus dependencias y aísla un *frente de interacciones inmediatas*: el conjunto de operaciones que están listas lógicamente pero cuya ejecución física depende del mapeo vigente. Si alguna compuerta del frente ya conecta sitios vecinos, se ejecuta de inmediato; en caso contrario, SABRE genera SWAP candidatos únicamente en torno a los qubits implicados, evitando explorar reconfiguraciones subóptimas en regiones distantes del procesador [8].

La heurística que selecciona el SWAP combina tres criterios físicamente intuitivos: resolver las interacciones urgentes del frente, anticipar las inmediatamente posteriores para no comprometerlas con un atajo apresurado, y distribuir espacialmente la actividad entrelazante para favorecer SWAP paralelizables que no incrementen la profundidad total del circuito [8].

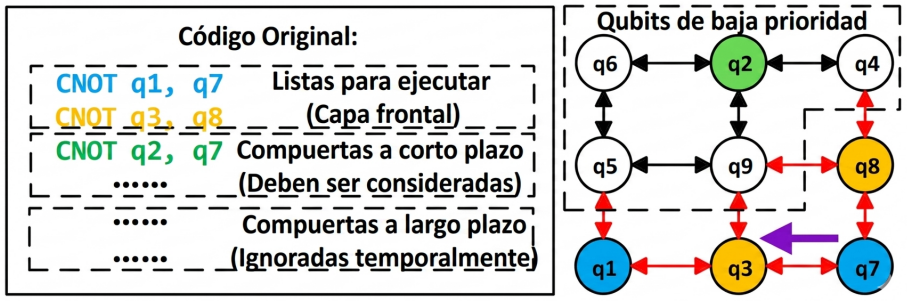

Figura 4.12: Ejemplo de búsqueda heurística basada en SWAP [8].

Un rasgo particularmente potente de SABRE es que no se limita al transporte local de estados, sino que también optimiza la colocación inicial de los qubits sobre el cristal. Para ello explota la reversibilidad de la mecánica cuántica: ejecuta un recorrido hacia adelante, registra el mapeo final, transpila el circuito invertido tomando ese resultado como condición de contorno y, finalmente, refina con esa información el mapeo inicial del circuito original. Esta bidireccionalidad es esencial en circuitos como el de Shor, donde optimizar únicamente las primeras compuertas resulta insuficiente: la distribución física inicial debe anticipar las interacciones más exigentes que aparecerán a lo largo de la exponenciación modular [8].

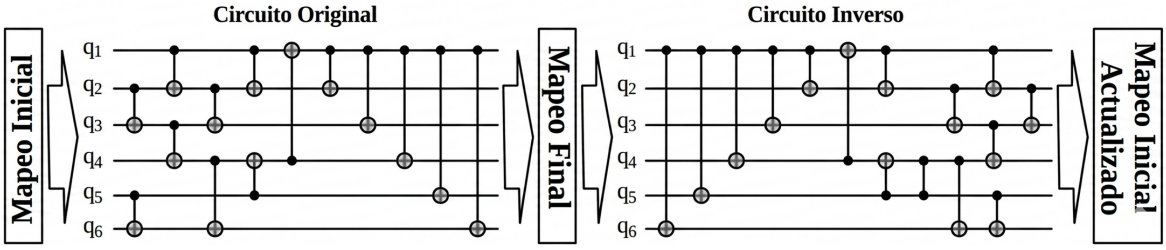

Figura 4.13: Actualización inicial del mapeo mediante la técnica de recorrido inverso [8].

En síntesis, SABRE reduce la sobrecarga de transporte cuántico restringiendo la búsqueda a los SWAP físicamente relevantes y eligiendo entre ellos con una métrica de distancia que aproxima el coste experimental real: menos intercambios implican menos compuertas entrelazantes y, por tanto, menor acumulación de decoherencia [8]. Esta filosofía resulta especialmente apropiada para la implementación del algoritmo de Shor sobre hardware NISQ, donde el cuello de botella no es computacional sino físico: cada compuerta adicional debilita la señal útil del estado cuántico antes de la medición, de modo que un algoritmo de enrutamiento

que minimiza profundidad y operaciones entrelazantes incrementa directamente la viabilidad experimental del circuito [58].

4.6 Estrategias de supresión en compilaciones complejas

En los dispositivos NISQ, incluso los circuitos relativamente cortos del algoritmo de Shor acumulan errores con rapidez: las compuertas de dos qubits (ECR o CZ) introducen errores coherentes a cada aplicación, y los intervalos en los que un qubit permanece inactivo lo exponen a la decoherencia del entorno. Sin acceso a corrección de errores completa, se recurre a técnicas de *supresión de errores* que actúan durante la compilación y la ejecución del circuito con bajo costo adicional. En este trabajo se emplean dos técnicas complementarias: el *Desacoplamiento Dinámico* (DD) y el *Pauli Twirling* (PT). Ambas son compatibles con el flujo de transpilación (incluyendo SABRE) y se activan automáticamente desde Qiskit como configuraciones del `SamplerV2`.

4.6.1 Desacoplamiento Dinámico (Dynamical Decoupling)

Durante los tiempos de espera entre compuertas (*idle times*), un qubit superconductor permanece expuesto a interacciones no controladas con su entorno: fluctuaciones térmicas, *crosstalk* estático con qubits vecinos y acoplamiento con modos del resonador [3, 9]. Estas interacciones, agrupadas en un acoplamiento sistema–baño H_{SB}, hacen que la fase relativa del qubit evolucione de forma impredecible (*dephasing*), degradando la coherencia que el algoritmo necesita para producir interferencia constructiva [59]. El Desacoplamiento Dinámico no elimina físicamente este acoplamiento: lo *promedia a cero* mediante la aplicación rápida de pulsos de control sobre el qubit.

La intuición proviene del *eco de espín* de Hahn (1950) en resonancia magnética nuclear [60]: si durante un intervalo τ el qubit acumula una fase errónea $+\phi$ debido al ruido, un pulso π alrededor del eje x (compuerta X) invierte el signo de la evolución del ruido, de modo que durante el siguiente intervalo τ se acumula $-\phi$ y ambas contribuciones se cancelan a primer orden en τ. La generalización moderna consiste en aplicar secuencias periódicas de pulsos durante el intervalo de inactividad, de manera que en el límite de pulsos rápidos el Hamiltoniano efectivo de error se promedia a un término que actúa exclusivamente sobre el baño y deja al qubit desacoplado del entorno a primer orden [9, 59].

Sobre esta base, una de las secuencias mas relevante es XY4, propuesta originalmente por Maudsley en NMR [61], que alterna pulsos π alrededor de los ejes x e y:

$$\text{XY4} \equiv Y - \tau - X - \tau - Y - \tau - X - \tau. \tag{4.21}$$

XY4 destaca sobre alternativas más simples (como la secuencia X-X, equivalente a CPMG) porque responde a una propiedad física concreta: al rotar simultáneamente alrededor de dos ejes ortogonales del espacio de Pauli, XY4 simetriza errores tanto de defasamiento (σ_z) como

de *bit-flip* (σ_x) e interacciones cruzadas (σ_y), mientras que X-X solo cancela los términos que anticonmutan con X y deja sin atender los acoplamientos σ_x estáticos [9, 62]. XY4 es, por tanto, una secuencia universal de primer orden cuya residual escala como $\mathcal{O}(\tau^2)$.

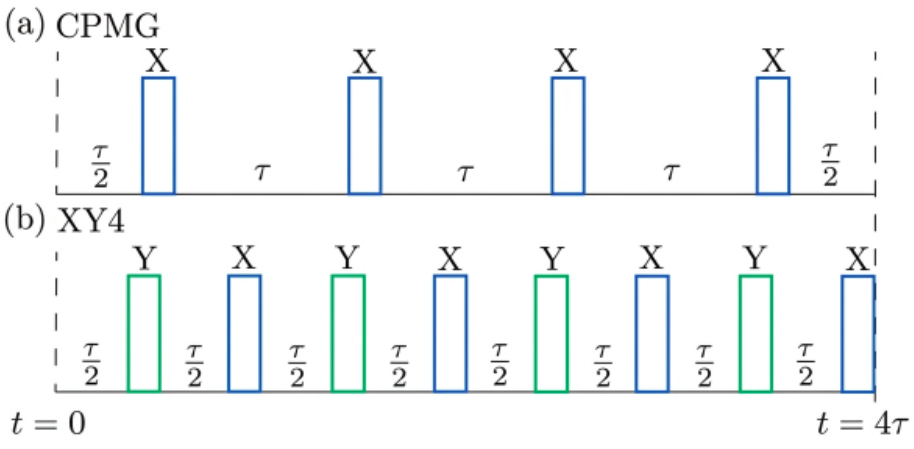

Figura 4.14: Diagrama esquemático de secuencias DD. (a) Dos secuencias CPMG concatenadas. (b) Dos secuencias XY4 concatenadas. Todas las secuencias se muestran para un período de inactividad de 4τ. CPMG usa solo compuertas X (rectángulo azul), mientras que XY4 usa compuertas X e Y (rectángulos verdes) [62].

4.6.2 Pauli Twirling (PT)

A diferencia del DD, que actúa sobre los intervalos de inactividad, el Pauli Twirling aborda los errores que ocurren *durante* la ejecución de las propias compuertas de dos qubits. Una compuerta real implementa una unitaria $\tilde{\mathcal{U}} = \mathcal{U} \cdot \mathcal{E}$ ligeramente desviada de la ideal $\mathcal{U}$, donde $\mathcal{E}$ representa una pequeña rotación con dirección y magnitud fijas. Al tratarse de un error *coherente* (repetible y no aleatorio) las desviaciones se acumulan de forma constructiva, es decir, cuadráticamente con el número N_g de compuertas, mientras que los errores estocásticos lo hacen linealmente [63, 64]. En circuitos profundos como los del algoritmo de Shor, este crecimiento cuadrático constituye el modo de fallo dominante.

El Pauli Twirling, también denominado *compilación aleatorizada* [63], transforma los errores coherentes en errores estocásticos. La operación consiste en envolver cada compuerta de dos qubits $\mathcal{U}$ entre un par de operadores de Pauli $(P_{\text{antes}}, P_{\text{después}})$ con $P \in \{I, X, Y, Z\}^{\otimes 2}$, elegidos aleatoriamente de un conjunto finito de combinaciones válidas. Tras promediar sobre múltiples instancias aleatorizadas del circuito, las componentes fuera de la diagonal del canal de error en la base de Pauli se cancelan y el ruido residual queda descrito por un canal estocástico de Pauli, cuya acumulación es lineal en N_g [10, 64].

La condición física que hace posible esta operación sin alterar el algoritmo es que el par de Pauli debe preservar la operación lógica de la compuerta original. Para una compuerta nativa de dos qubits como CZ (la utilizada en Heron r1), los pares válidos satisfacen

$$P_{\text{después}} \cdot \mathcal{U} \cdot P_{\text{antes}} = e^{i\phi}\,\mathcal{U}, \tag{4.22}$$

para alguna fase global ϕ, irrelevante para las probabilidades de medición [64]. Cada compuerta de dos qubits admite, por tanto, un conjunto finito de pares válidos del que el motor de ejecución selecciona aleatoriamente en cada instancia.

DD y PT actúan sobre fuentes de error diferentes: el primero protege a los qubits durante los intervalos de inactividad, el segundo aleatoriza los errores de las compuertas activas. El análisis detallado se reporta en los Apéndices A.5 y A.6.

4.7 Métricas de Evaluación

Para evaluar qué tan bien funciona un algoritmo cuántico ejecutado en hardware real, se necesitan indicadores numéricos que permitan comparar el resultado obtenido con el resultado esperado. En un mundo ideal, esta comparación se haría reconstruyendo completamente el estado cuántico final del sistema mediante Tomografía de Estado Cuántico (QST) [24]. Sin embargo, para un sistema de m qubits, QST requiere la medición de $4^m - 1$ observables independientes. En el caso del algoritmo de Shor para $N = 15$ con $m = 9$ qubits de control, esto implica más de $2{,}6 \times 10^5$ mediciones distintas, una demanda que excede tanto los tiempos de coherencia de los dispositivos NISQ como los límites de acceso del IBM Quantum Open Plan [19].

Ante esta limitación, se adoptan métricas que trabajan directamente con las distribuciones de probabilidad obtenidas al medir en la base computacional, sin necesidad de reconstruir la matriz de densidad completa. En este trabajo se emplean dos métricas complementarias: la *fidelidad de Hellinger*, que cuantifica la similitud estadística entre la distribución ideal y la experimental, y la *probabilidad de éxito experimental* (PST), que mide la fracción de mediciones que caen en los resultados correctos del algoritmo.

4.7.1 Fidelidad de Hellinger

La idea intuitiva detrás de la fidelidad es simple: dados dos estados cuánticos, ¿qué tan parecidos son? Si un circuito ideal produce el estado ρ y el hardware real produce el estado σ, la fidelidad cuántica mide qué tanto se "solapan" ambos estados. Formalmente, la fidelidad de Uhlmann–Jozsa se define como [24, 65]:

$$F(\rho, \sigma) = \left(\mathrm{Tr} \sqrt{\sqrt{\rho}\,\sigma\,\sqrt{\rho}} \right)^2 . \tag{4.23}$$

Esta cantidad satisface $0 \leq F \leq 1$, donde $F = 1$ si y solo si los estados son idénticos, y $F = 0$ cuando son completamente ortogonales (distinguibles con certeza).

Cuando el estado de referencia es puro (como ocurre al comparar con una simulación ideal) la expresión se simplifica notablemente. Si $\rho = |\psi\rangle\langle\psi|$, la Ecuación 4.23 se reduce a [24, 65]:

$$F(|\psi\rangle\langle\psi|, \sigma) = \langle\psi|\sigma|\psi\rangle. \tag{4.24}$$

Es decir, la fidelidad se convierte simplemente en la probabilidad de que, al medir el estado experimental σ, se obtenga el estado ideal $|\psi\rangle$. Esta interpretación conecta directamente la fidelidad con una cantidad físicamente medible. El problema práctico es que calcular la Ecuación 4.24 requiere conocer la matriz de densidad σ, lo cual exige la tomografía completa que ya se descartó. La solución consiste en trabajar con las distribuciones de probabilidad que sí están disponibles: las frecuencias relativas de cada cadena binaria obtenidas tras muchas mediciones del circuito.

Sea $P = \{p_x\}$ la distribución de probabilidad ideal (obtenida del simulador sin ruido) y $Q = \{q_x\}$ la distribución experimental, donde cada x es una cadena de m bits. La *fidelidad clásica* o *fidelidad de Hellinger* se define como [24, 32]:

$$\mathcal{F}_H(P, Q) = \left(\sum_{x \in \{0,1\}^m} \sqrt{p_x \cdot q_x} \right)^2 . \tag{4.25}$$

Si bien la cantidad dentro del paréntesis, $\sum_x \sqrt{p_x q_x}$, no es una métrica en el sentido matemático, funciona perfectamente como medida de similitud acotada. El rango de $\mathcal{F}_H$ es $[0, 1]$. Un valor $\mathcal{F}_H \to 1$ indica que las distribuciones ideal y experimental son prácticamente indistinguibles; un valor $\mathcal{F}_H \to 0$ señala que el ruido ha destruido completamente la estructura de probabilidades del algoritmo.

La relación entre $\mathcal{F}_H$ y la fidelidad cuántica F está dada por un resultado establecido en la teoría de la información cuántica. Fuchs demostró que la fidelidad clásica obtenida a partir de cualquier medición (formalizada como un POVM) siempre proporciona una cota superior de la fidelidad cuántica [39, 65]:

$$F(\rho, \sigma) \leq \mathcal{F}_H(P, Q). \tag{4.26}$$

Esta desigualdad tiene una consecuencia práctica importante: si la fidelidad de Hellinger calculada a partir de las mediciones experimentales resulta baja, la fidelidad cuántica real es necesariamente aún más baja. En otras palabras, $\mathcal{F}_H$ ofrece una estimación *optimista* de la similitud entre los estados ideal y experimental.

4.7.2 Probabilidad de Éxito Experimental (PST)

Mientras que la fidelidad de Hellinger ofrece una visión global de la similitud entre distribuciones, la PST responde a una pregunta más directa y operativa: ¿qué fracción de las

mediciones produce un resultado que conduce a la factorización correcta?

El algoritmo de Shor funciona porque la estimación de fase cuántica (QPE) concentra la probabilidad de medición en un conjunto discreto de posiciones $\mathcal{Y}_{\text{ideal}}$ dentro del registro de conteo. Estas posiciones corresponden a las fases $\varphi_s = s/r$ de los eigenestados del operador de multiplicación modular U_x, y en el histograma aparecen como picos bien definidos en las posiciones [24]:

$$y_s = \left\lfloor \frac{s \cdot 2^m}{r} \right\rceil \quad (\text{mód } 2^m), \quad s \in \{0, 1, \ldots, r-1\}. \tag{4.27}$$

La PST se define como la suma de las probabilidades experimentales medidas exclusivamente en estas posiciones [20]:

$$PST = \sum_{y \in \mathcal{Y}_{\text{ideal}}} q_y. \tag{4.28}$$

En una simulación ideal sin ruido, toda la probabilidad se concentra en los picos del QPE, de modo que $PST = 1$. En el régimen ruidoso, la decoherencia y los errores de compuerta redistribuyen la probabilidad hacia cadenas binarias que no corresponden a ninguna fase válida, reduciendo la PST.

La PST complementa a $\mathcal{F}_H$ de forma natural: mientras la fidelidad mide la similitud global entre distribuciones, la PST cuantifica directamente la utilidad algorítmica de los resultados experimentales. Es posible obtener una $\mathcal{F}_H$ moderada pero una PST aceptable si el ruido redistribuye la probabilidad de forma relativamente uniforme entre los estados no deseados, manteniendo los picos correctos como los más probables. En la práctica, ambas métricas se reportan conjuntamente para ofrecer una evaluación completa del rendimiento del circuito en cada entorno de ejecución.

Capítulo 5

Diseño metodológico e implementación experimental

Este capítulo documenta el diseño experimental empleado para evaluar el algoritmo de Shor con $N = 15$. La exposición se organiza en torno a las decisiones que estructuran el estudio: la delimitación del aporte original respecto al repositorio base, la selección clásica de parámetros, la construcción del circuito ideal, el protocolo experimental en tres fases concebido para aislar fuentes de error sucesivas, su transpilación, las técnicas de supresión evaluadas y, finalmente, el post-procesamiento clásico y las métricas adoptadas. Los detalles propios de la implementación, se trasladan al Apéndice B.

Nota. A partir de aquí, la variable x (base en la búsqueda de orden) del algoritmo de Shor se reemplaza por a, dado que en la práctica esta última es el parámetro de entrada efectivo del algoritmo.

5.1 Delimitación del aporte respecto al repositorio base

El repositorio proporcionado por [20] aportó la arquitectura general de clases y un flujo de ejecución gobernado por archivos de configuración. Sin embargo, dicha base fue concebida sobre versiones anteriores del SDK de Qiskit y contenía dependencias, nomenclaturas y elecciones metodológicas incompatibles o subóptimas respecto al ecosistema actual de IBM Quantum. El detalle técnico de cada migración (clases sustituidas, módulos reorganizados, conversión *bitstring*-a-entero y verificaciones implementadas) se documenta en el Apéndice B.1.

5.2 Pre-procesamiento clásico y selección de parámetros

El algoritmo de Shor posee una naturaleza fundamentalmente híbrida: antes de invocar la subrutina cuántica, se ejecuta una serie de verificaciones clásicas que pueden resolver la factorización sin consumir recursos cuánticos. La envoltura clásica implementa esta lógica en un

flujo secuencial, coherente con el procedimiento formal descrito en el Algoritmo 3 del marco teórico. Solo cuando ninguna de estas verificaciones produce resultados, se procede a la fase cuántica.

5.2.1 Selección de la base a y caracterización del grupo multiplicativo

Para el caso de estudio $N = 15$, el grupo multiplicativo $\mathbb{Z}_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$ contiene ocho elementos coprimos con 15. Las bases candidatas no triviales son siete ($a = 1$ es trivial). Una rutina de pre-cálculo clásico determina, para cada base candidata, su orden multiplicativo r (evaluando sucesivamente a^k mód N para $k = 1, 2, \ldots, N-1$ hasta encontrar el primer k tal que $a^k \equiv 1 \pmod{N}$) y verifica las condiciones de no-degeneración del Algoritmo 3. Los resultados de esta caracterización, fundamentales para el diseño experimental, se resumen en la Tabla 5.1.

Tabla 5.1: Caracterización del grupo $\mathbb{Z}_{15}^*$: bases, órdenes y factores obtenidos.

Base a	**Orden** r	r **par**	$a^{r/2}$ mód 15	gcd$(a^{r/2}-1, 15)$	gcd$(a^{r/2}+1, 15)$
2	4	Sí	4	3	5
4	2	Sí	4	3	5
7	4	Sí	4	3	5
8	4	Sí	4	3	5
11	2	Sí	11	3	5
13	4	Sí	4	3	5
14	2	Sí	14	*trivial*	*trivial*

La rutina automatiza este análisis: filtra las bases cuyo periodo r es par, verifica la condición $a^{r/2} \not\equiv -1 \pmod{N}$ y descarta las que producen factores triviales (1 o N). La condición de *degeneración* descrita se refiere exclusivamente al post-procesamiento clásico de fracciones continuas, no al circuito cuántico propiamente dicho, lo que justifica la inclusión diagnóstica de $a = 14$ en la Fase 3 (Sección 5.4).

Desde la perspectiva de la complejidad del circuito cuántico, la selección de a tiene consecuencias directas. Las bases con período $r = 2$ (como $a = 4$) requieren que la estimación de fase distinga entre solo dos fases ($s/r \in \{0/2, 1/2\}$), lo que genera circuitos de menor profundidad efectiva. En contraste, las bases con $r = 4$ (como $a = 7$ o $a = 13$) exigen resolver cuatro fases y producen operadores de exponenciación modular más complejos. El diseño experimental explota esta diferencia evaluando sistemáticamente ambos grupos para cuantificar el impacto de la profundidad del circuito en la degradación por ruido.

5.2.2 Dimensionamiento de los registros cuánticos

El tamaño de los registros cuánticos se calcula dinámicamente a partir de N. Esta configuración es la utilizada en todos los experimentos reportados en el Capítulo 6.

Tabla 5.2: Configuración de registros cuánticos para $N = 15$.

Registro	**Fórmula**	**Valor**
Trabajo (qubits)	$L = \lceil \log_2 N \rceil = \lceil \log_2 15 \rceil$	4 qubits
Conteo / control	$t = 2L + 1$	9 qubits
Total lógico	$L + t$	**13** qubits
Registro clásico	mediciones del control	9 bits

5.3 Construcción del circuito ideal

La síntesis del circuito de Shor se organiza como una secuencia de tres bloques funcionales que reproducen el Algoritmo 1 de estimación de fase cuántica: la creación de la superposición inicial, la aplicación del oráculo de exponenciación modular controlada y la extracción de fase mediante la QFT inversa, tal como se muestra en la Figura 5.1.

5.3.1 Síntesis del oráculo de exponenciación modular

El núcleo computacional del circuito reside en la secuencia de operaciones controladas $c\text{-}U_a^{2^j}$ para $j = 0, 1, \ldots, t-1$. La construcción de cada operador unitario $U_a^{2^j}$ sigue un procedimiento en dos pasos:

1. *Pre-cómputo clásico.* Para cada qubit de control j, se calcula clásicamente la constante $b = a^{2^j} \bmod N$ mediante exponenciación modular por cuadrados repetidos. Esta operación, ejecutada en tiempo clásico $\mathcal{O}(L^2)$, evita la necesidad de implementar cuánticamente la secuencia completa de cuadrados repetidos descrita en la Ec. A.4, lo cual habría requerido miles de compuertas adicionales.

2. *Construcción de la matriz de permutación.* Con la constante b ya conocida, se construye una matriz de permutación P_b de dimensión $2^L \times 2^L$ que codifica el mapeo $|x\rangle \to |bx \bmod N\rangle$ para todos los estados $x \in \{0, 1, \ldots, N-1\}$. Los estados con $x \geq N$, que no tienen significado aritmético en el contexto del algoritmo, se mapean a sí mismos, es decir, $P_b[x][x] = 1$ para $x \geq N$, garantizando que la matriz sea una permutación unitaria completa. La matriz resultante se encapsula como una compuerta unitaria que posteriormente se convierte en una compuerta controlada.

Este enfoque de síntesis (también denominado *compiled Shor's algorithm* en la literatura [5, 6]) tiene una ventaja decisiva para la implementación NISQ: el transpilador recibe una única compuerta unitaria controlada de $L + 1$ qubits en lugar de un circuito aritmético profundo. Aunque el transpilador deberá descomponer esta unitaria en compuertas nativas del hardware, el conteo final de compuertas de dos qubits es significativamente menor que el de una implementación aritmética genérica [7]. Esta estrategia, ha sido objeto de crítica por

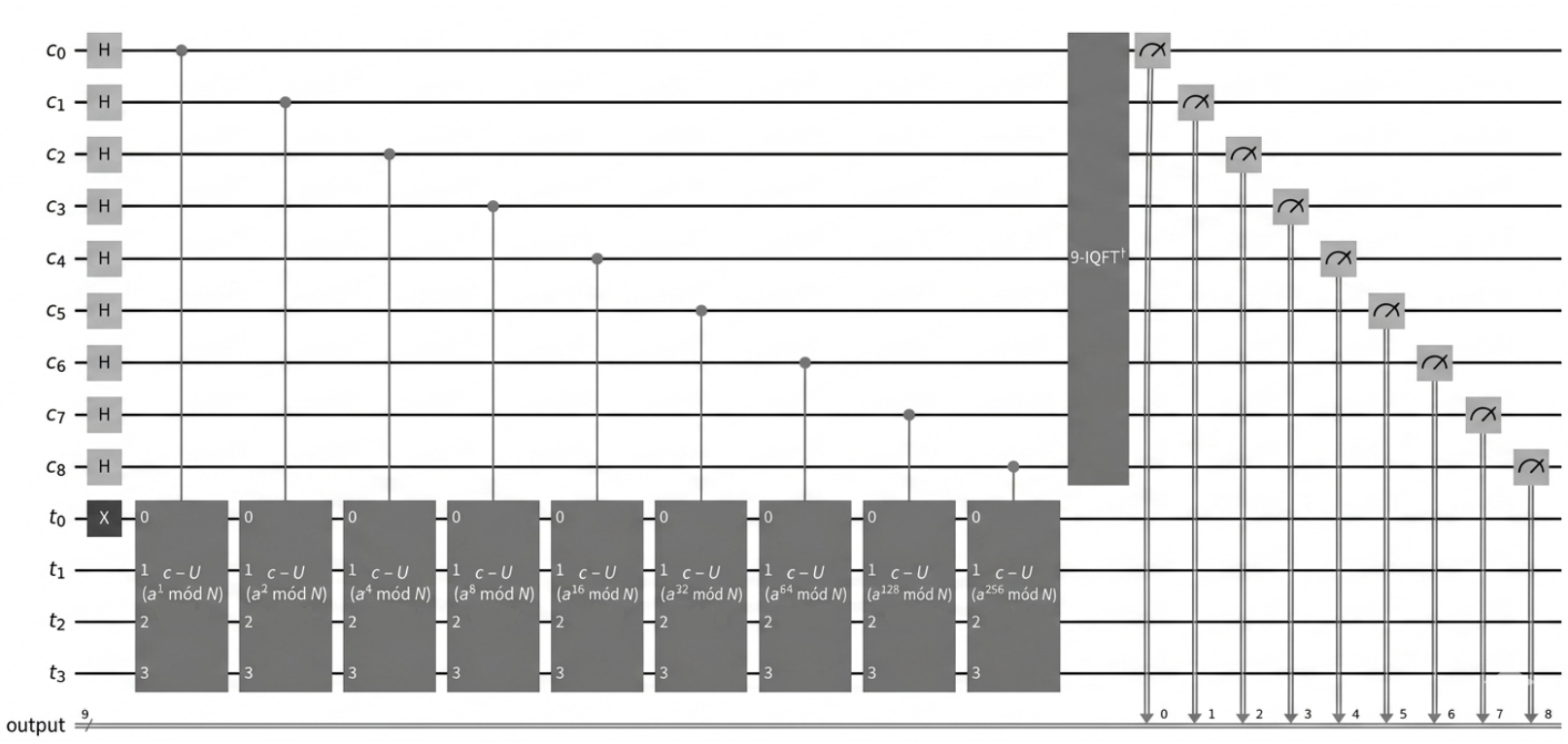

Figura 5.1: Representación esquemática del algoritmo de Shor para la búsqueda de periodo. El registro superior ($n = 9$ qubits) se inicializa en una superposición uniforme mediante compuertas Hadamard (H). El registro inferior ($m = 4$ qubits) se inicializa en el estado computacional $|1\rangle$ mediante una compuerta X. La cascada de oráculos representa la exponenciación modular controlada $c\text{-}U_{a^{2^j} \pmod N}$, donde cada base $b = a^{2^j} \pmod N$ ha sido pre-computada clásicamente. El proceso finaliza con una Transformada de Fourier Cuántica Inversa ($QFT^\dagger$) y la medición del registro de control.

Smolin et al. [7] al señalar que el conocimiento previo de la base a y de su orden r permite simplificaciones que un Shor general no podría aprovechar. El presente trabajo adopta esta vía siguiendo la práctica establecida para implementaciones NISQ pedagógicas, y la discusión de sus implicaciones sobre el alcance interpretativo de los resultados se retoma en las limitaciones del trabajo (§7).

5.4 Protocolo experimental en tres fases

El protocolo experimental se diseñó como una secuencia de tres entornos de ejecución progresivamente más realistas, de modo que la diferencia entre los resultados de dos fases consecutivas pueda atribuirse a una única fuente de degradación añadida.

Fase 1: simulación ideal

El objetivo de esta fase es doble: verificar la corrección lógica del circuito reproduciendo las distribuciones predichas por la teoría de QPE y establecer la línea base cuantitativa contra la cual se medirá la degradación en las fases posteriores. Como variante complementaria se ejecuta el mismo circuito sobre el simulador inicializado a partir del *snapshot* del procesador real con el modelo de ruido explícitamente desactivado: esta variante fuerza la topología Heavy-Hex sin aplicar decoherencia y aísla así el *overhead* topológico del impacto del ruido.

Fase 2: simulación con ruido (*Fake Backends*)

El *backend* se instancia a partir del *snapshot* de calibración del procesador `ibm_torino`, lo que carga automáticamente: los errores incoherentes (despolarización y relajación térmica) parametrizados por las tasas de error y los tiempos $T_1^{(i)}$, $T_2^{(i)}$ individuales de cada qubit; las matrices de confusión asimétricas de los errores de lectura; y el mapa de acoplamiento Heavy-Hex con las duraciones nominales de cada compuerta nativa. La limitación física fundamental de esta fase, que condiciona la lectura de sus resultados, es que el *snapshot* captura exclusivamente ruido incoherente: no modela los errores coherentes (sobre-rotaciones sistemáticas) ni el *crosstalk* dinámico entre qubits adyacentes que sí están presentes en el procesador físico. En consecuencia, los resultados de la Fase 2 deben interpretarse como una cota de la degradación esperable en hardware, y no como un predictor exacto de la Fase 3.

Fase 3: ejecución en hardware real

Los circuitos se envían al procesador `ibm_torino` (Heron r1, 133 qubits superconductores transmon en topología Heavy-Hex) a través del servicio en la nube de IBM Quantum bajo el plan de acceso abierto. La selección de este *backend* obedece a dos criterios. Primero, por consistencia metodológica: el *snapshot* empleado en la Fase 2 es directamente el de `ibm_torino`, de modo que cualquier diferencia entre las Fases 2 y 3 puede atribuirse al ruido coherente y al *crosstalk* no modelados, y no a discrepancias de arquitectura, topología

o conjunto de compuertas nativas. Segundo, por disponibilidad: durante el período experimental, los demás procesadores Heron presentaban colas con decenas de *jobs* en espera, lo que habría comprometido la agilidad necesaria para iterar sobre múltiples configuraciones. Las características nominales del *backend* se reportan en el Capítulo 6 y el desglose técnico (identificadores de *job*, configuración exacta de `SamplerV2` y `SamplerOptions`, organización de los estudios V1, A2, A3 y A6/A7) se documenta en el Apéndice B.5.

En las tres fases, los resultados de cada ejecución se almacenan en formato JSON (incluyendo los identificadores de *job* en el caso del hardware) para garantizar la trazabilidad y permitir el post-procesamiento asíncrono. El flujo de trabajo experimental se esquematiza en la Figura 5.2.

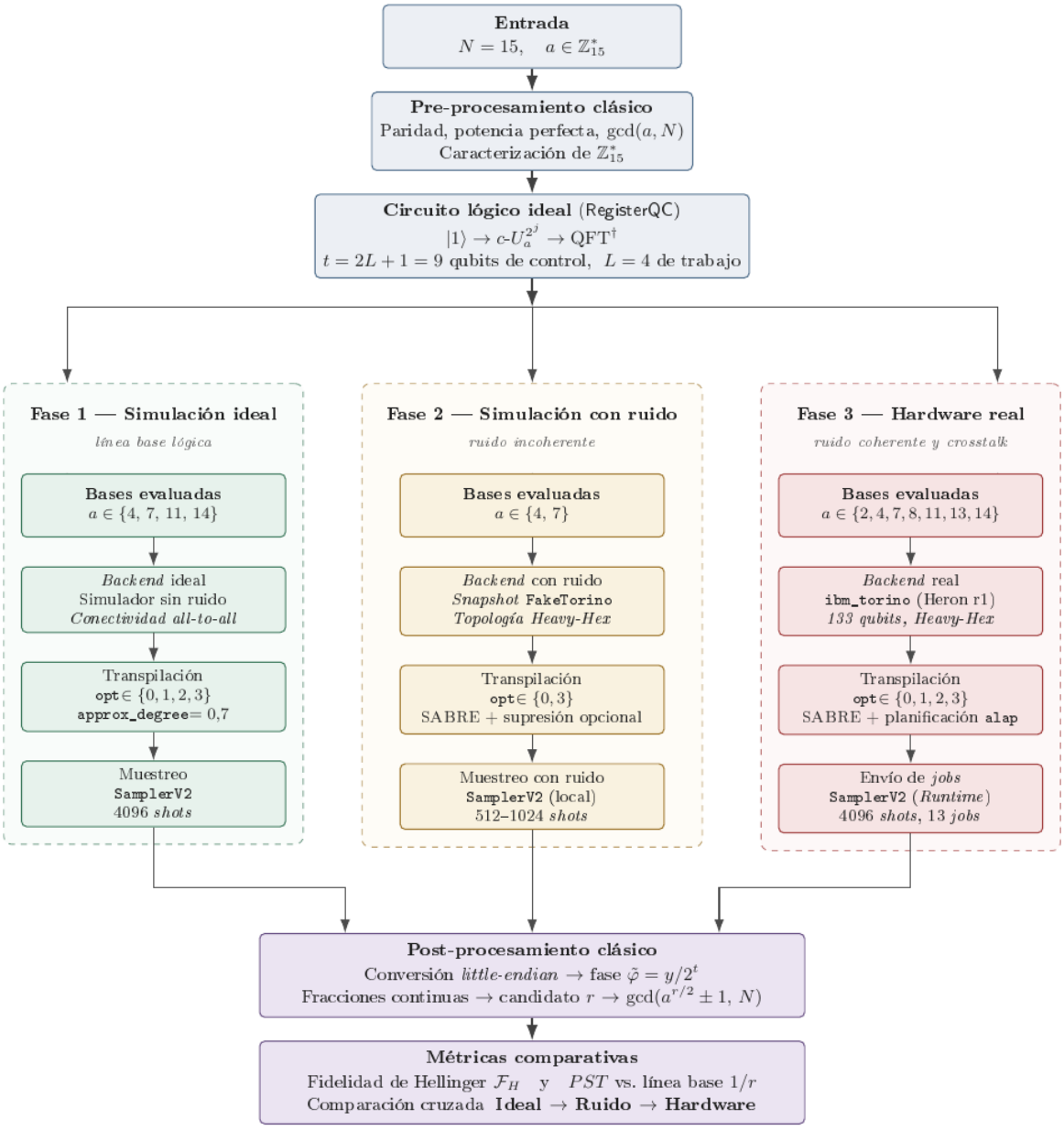

Figura 5.2: Diagrama de flujo del protocolo experimental por fases. (Fuente: creada con Claude, Anthropic, (2026).)

Las diferencias entre fases obedecen a un compromiso entre rigor estadístico y costo computacional: en Fase 1, el bajo costo del simulador ideal permite barrer las cuatro bases diagnósticas y los cuatro niveles de optimización con 4096 *shots*; en Fase 2, el costo del modelo de ruido (aproximadamente dos órdenes de magnitud superior) obliga a restringir el barrido a las dos bases representativas $a \in \{4, 7\}$ y a los niveles extremos $\texttt{opt} \in \{0, 3\}$, el número de *shots* adoptado preserva la capacidad de discriminar los picos teóricos del piso de

ruido (propósito principal de la fase) aunque se reconoce que reduce la resolución estadística absoluta de las métricas reportadas, lo que se tiene presente al interpretar diferencias entre configuraciones de esta fase. En la Fase 3, el cupo de *jobs* disponibles bajo el plan de acceso abierto fija el alcance experimental.

Bases diagnósticas: la base $a = 14$ produce factores triviales tras el post-procesamiento clásico (Sección 5.2), pero su circuito es instancia plenamente válida de la estimación de fase cuántica con orden $r = 2$. Su inclusión en la Fase 3 no busca factorizar N, sino caracterizar la respuesta del procesador para circuitos cortos: en estas ejecuciones las métricas relevantes son exclusivamente la fidelidad de Hellinger $\mathcal{F}_H$ y la PST, que cuantifican la calidad de la señal cuántica con independencia del éxito del filtro clásico.

5.5 Transpilación

Las decisiones metodológicas relevantes en esta etapa son las siguientes:

1. Se barrió (con cierta limitación) el nivel de optimización del transpilador en su rango completo (`opt` $\in \{0, 1, 2, 3\}$) para cuantificar el impacto progresivo de las transformaciones sobre el circuito final.

2. Tanto la asignación inicial de qubits como el enrutamiento posterior se delegaron al algoritmo SABRE descrito en la Sección 4.5 del marco teórico.

3. La semilla del transpilador se fijó (`seed_transpiler`$= 457$) para garantizar la reproducibilidad estadística de las heurísticas.

4. Durante la síntesis se permitió una aproximación controlada al circuito exacto fijando `approximation_degree`$= 0{,}7$, con la justificación expuesta en Apéndice B.3.

5. Para las ejecuciones en hardware se añadió, además, una planificación temporal *as late as possible*, que concentra los intervalos *idle* al inicio del circuito y deja espacio efectivo para la inserción de las secuencias de Desacoplamiento Dinámico descritas en la Sección 5.6.

Como métrica primaria de la calidad del circuito transpilado se adoptó la profundidad de dos qubits, D_{2Q}, definida como la longitud del camino crítico considerando exclusivamente las compuertas que operan sobre dos qubits. Esta es la métrica predictivamente más significativa para hardware NISQ: cada compuerta CZ contribuye un error del orden de $\epsilon_{2Q} \sim 5 \times 10^{-3}$ en Heron r1, de modo que D_{2Q} controla, en primer orden, la fidelidad acumulada del circuito completo. La tabulación de D_{2Q} por nivel de optimización permite, además, separar el efecto del *layout* del efecto del *routing* cuando se compara una asignación trivial frente a SABRE.

La lista detallada de parámetros del transpilador, los valores nominales de los hiperparámetros avanzados de SABRE (número de iteraciones, ensayos de *layout* y de SWAP) y el procedimiento concreto por el cual se inyectan en el *pipeline* de transpilación se documentan en el Apéndice B.3.

5.6 Técnicas de supresión de errores

El estudio del impacto de las técnicas de supresión de errores se planificó inicialmente como un diseño factorial 2×2 que combina la activación o desactivación de Desacoplamiento Dinámico (DD) y de Pauli Twirling (PT). En la Fase 2 (`FakeTorino`) este diseño se ejecutó de forma completa: las cuatro variantes resultantes (sin supresión, solo DD, solo PT, DD+PT) se evaluaron sobre la misma transpilación del circuito, de modo que las diferencias observadas en las métricas se atribuyen exclusivamente al efecto de las técnicas. En la Fase 3 (hardware real), las restricciones de cupo del Open Plan y la retirada anticipada de `ibm_torino` de la flota disponible impidieron completar la cuarta celda (solo PT) como ejecución independiente. En su lugar, esa celda se obtiene por comparación entre la línea base sin supresión del estudio A6/A7 y la corrida V1 (con PT activado y sin DD) sobre el mismo circuito, asumiendo el riesgo metodológico de que ambas ejecuciones corresponden a ventanas de calibración distintas; esta limitación se discute explícitamente en la interpretación de los resultados (§6.3.4) y en las limitaciones del trabajo (§7).

Selección de la secuencia DD

El servicio de ejecución admite varias secuencias de pulsos para DD, entre ellas `XX`, `XpXm` y `XY4`. En este trabajo se adoptó `XY4` en sustitución de `XpXm`, que era la opción heredada del repositorio base. La secuencia `XY4` cancela a primer orden tanto el ruido de fase como el de bit a través de cuatro pulsos en direcciones ortogonales del plano XY, mientras que `XpXm` se limita a refocalizar el desfase de un solo eje. Cuando DD está habilitado, el servicio identifica automáticamente los intervalos *idle* del circuito transpilado, calcula cuántos pulsos completos caben en cada intervalo según la duración del pulso X nativo del dispositivo e inserta la secuencia centrada simétricamente. Esto es especialmente relevante en el circuito de Shor, donde la aplicación secuencial de los $c\text{-}U_a^{2^j}$ genera largos periodos de inactividad en los qubits de control que no están interactuando en un paso dado.

Configuración del Pauli Twirling

En lugar del modo pasivo (en el que cada compuerta de dos qubits se aleatoriza con un único par de Pauli), se seleccionó la estrategia `active`, que aplica el *twirling* únicamente a los qubits efectivamente involucrados en cada compuerta y delega al *runtime* la generación automática de un número de instancias aleatorias coherente con el total de *shots*. Esta configuración es la recomendada por la primitiva V2 cuando el objetivo es convertir errores coherentes sistemáticos en ruido estocástico simétrico (Ec. A.10) y simultáneamente

preservar la economía de *shots*.

La parametrización exacta de los objetos de configuración de DD y PT (nombres de los *flags*, valores escogidos para cada uno y el procedimiento de inyección en `SamplerOptions`) se documenta en el Apéndice B.4.

5.7 Post-procesamiento clásico y métricas de evaluación

Tras el muestreo, los *bitstrings* obtenidos se procesan clásicamente para extraer el orden r y, a partir de él, los factores no triviales de N. El procedimiento, común a las tres fases, sigue seis pasos:

1. Recuperación de la distribución de conteos del registro clásico de salida.
2. Selección de los k resultados con mayor frecuencia (típicamente $k = 10$), ordenados de forma descendente.
3. Para cada *bitstring* seleccionado, conversión a entero y aplicando la convención *little-endian* (Apéndice B.1).
4. Cálculo de la fase estimada $\tilde{\varphi} = y/2^t$.
5. Aplicación del algoritmo de fracciones continuas con denominador acotado por N para obtener los convergentes p_k/q_k con $q_k < N$. El denominador q_k es un candidato al orden r.
6. Verificación clásica: $a^{q_k} \equiv 1 \pmod{N}$, q_k par y factores no triviales $\gcd(a^{q_k/2} \pm 1, N) \notin \{1, N\}$.

Cuando el filtro clásico devuelve un par de factores no triviales, la ejecución se reporta como exitosa. Los detalles de implementación se documentan en el Apéndice B.6.

Métricas adoptadas

La evaluación cuantitativa de cada ejecución se realiza con dos métricas complementarias, definidas formalmente en el marco teórico: la fidelidad de Hellinger $\mathcal{F}_H$ y la probabilidad de éxito experimental PST. Ambas se calculan comparando la distribución experimental contra la distribución ideal obtenida del simulador sin ruido.

Para interpretar los resultados que se presentan en el Capítulo 6 se adoptan umbrales operacionales explícitos sobre ambas métricas. En primer lugar, siguiendo el criterio teórico de [66] discutido en el marco teórico, la PST se compara contra la línea base uniforme que se obtendría si los M *shots* se distribuyeran al azar sobre los 2^t resultados posibles del registro de conteo. Como la subrutina deposita masa sobre r picos teóricos del QPE, dicha línea base

es $b_{\text{unif}} = r/2^t$; para $t = 9$ esto corresponde a $\approx 0{,}4\,\%$ con $r = 2$ y $\approx 0{,}78\,\%$ con $r = 4$. Una PST se considera *informativa* cuando supera de manera estadísticamente significativa esta línea base; el éxito algorítmico, sin embargo, no depende de superar un umbral fijo sino de que los picos teóricos sigan siendo los modos dominantes de la distribución empírica, requisito que el experimento confirma en regímenes donde la PST es modesta pero los picos correctos preservan su rango relativo. En cuanto a la fidelidad de Hellinger, dado que esta métrica está acotada en el intervalo $[0, 1]$ y cuantifica la similitud entre distribuciones clásicas [67], este trabajo adopta criterios operacionales de interpretación consistentes con prácticas habituales de benchmarking NISQ [45, 68]: valores $\mathcal{F}_H > 0{,}9$ se interpretan como alta concordancia con la distribución ideal, el rango $0{,}7 \leq \mathcal{F}_H \leq 0{,}9$ como concordancia moderada y $\mathcal{F}_H < 0{,}7$ como evidencia de degradación significativa inducida por ruido. La interpretación combinada de ambos umbrales constituye el criterio central de análisis en el Capítulo 6.

Todo el código fuente, los archivos de configuración empleados en cada experimento y los identificadores de los *jobs* de IBM Quantum se encuentran documentados en el repositorio público del proyecto [69], garantizando la reproducibilidad completa del estudio.

Capítulo 6

Resultados

6.1 Resultados de la simulación ideal

La validación del algoritmo de Shor para $N = 15$ se aborda en tres fases progresivas (simulación ideal, simulación con ruido calibrado de `FakeTorino` y ejecución en `ibm_torino`). Esta sección reporta la primera fase, ejecutada en el simulador `AerSimulator` de Qiskit.

6.1.1 Caracterización de los bloques fundamentales

Antes de evaluar el circuito completo, se verificó individualmente la corrección de cada sub-circuito constituyente: la Transformada Cuántica de Fourier (QFT) y la exponenciación modular controlada ($c\text{-}U_{a^{2^k}}$). La verificación numérica de la unitariedad arrojó $\|U^\dagger U - I\|_F < 10^{-13}$ para todas las variantes evaluadas (QFT directa e inversa, descompuesta y transpilada), valor compatible con la precisión de punto flotante. Para el operador de exponenciación modular se verificó adicionalmente la propiedad de periodicidad $(U_a)^r = I$, obteniendo $\|(U_a)^r - I\|_F = 0$ en todas las bases. El detalle exhaustivo se documenta en el Apéndice C.

Un resultado estructural relevante para todo lo que sigue es la asimetría entre bases (Figura 6.1). Para cada a, solo los bloques con a^{2^k} mód $N \neq 1$ contribuyen a la profundidad del circuito; los demás colapsan a la identidad. Esto explica por qué las bases con $r = 2$ producen circuitos significativamente menos profundos que $a = 7$ (con $r = 4$): para $r = 2$ únicamente el bloque $k = 0$ es no trivial, mientras que para $a = 7$ los bloques $k = 0$ y $k = 1$ lo son simultáneamente (7^1 mód $15 = 7$ y 7^2 mód $15 = 4$).

Esta diferencia tiene una consecuencia directa para la viabilidad experimental en hardware NISQ. Siguiendo el formalismo establecido por Nielsen y Chuang [24], el operador de exponenciación modular se descompone en una productoria controlada basada en la secuencia de exponentes a^{2^k} mód N. Una base con un orden multiplicativo r mayor siempre activa una secuencia de exponentes más densa y evita que los bloques colapsen prematuramente a la

identidad. Físicamente, esta mayor demanda se traduce en cascadas adicionales de compuertas entrelazantes (las cuales sufren las tasas de error intrínseco más altas del sistema). Puesto que la duración temporal de estos bloques no triviales agota rápidamente el tiempo de coherencia T_2 del procesador, bases como $a = 7$ están estructuralmente predispuestas a sufrir un colapso por decoherencia mucho más veloz que aquellas con $r = 2$. La selección clásica de a como estrategia de pre-procesamiento (Sección 5.2) queda así cuantitativamente justificada como un imperativo para mitigar la profundidad del circuito cuántico.

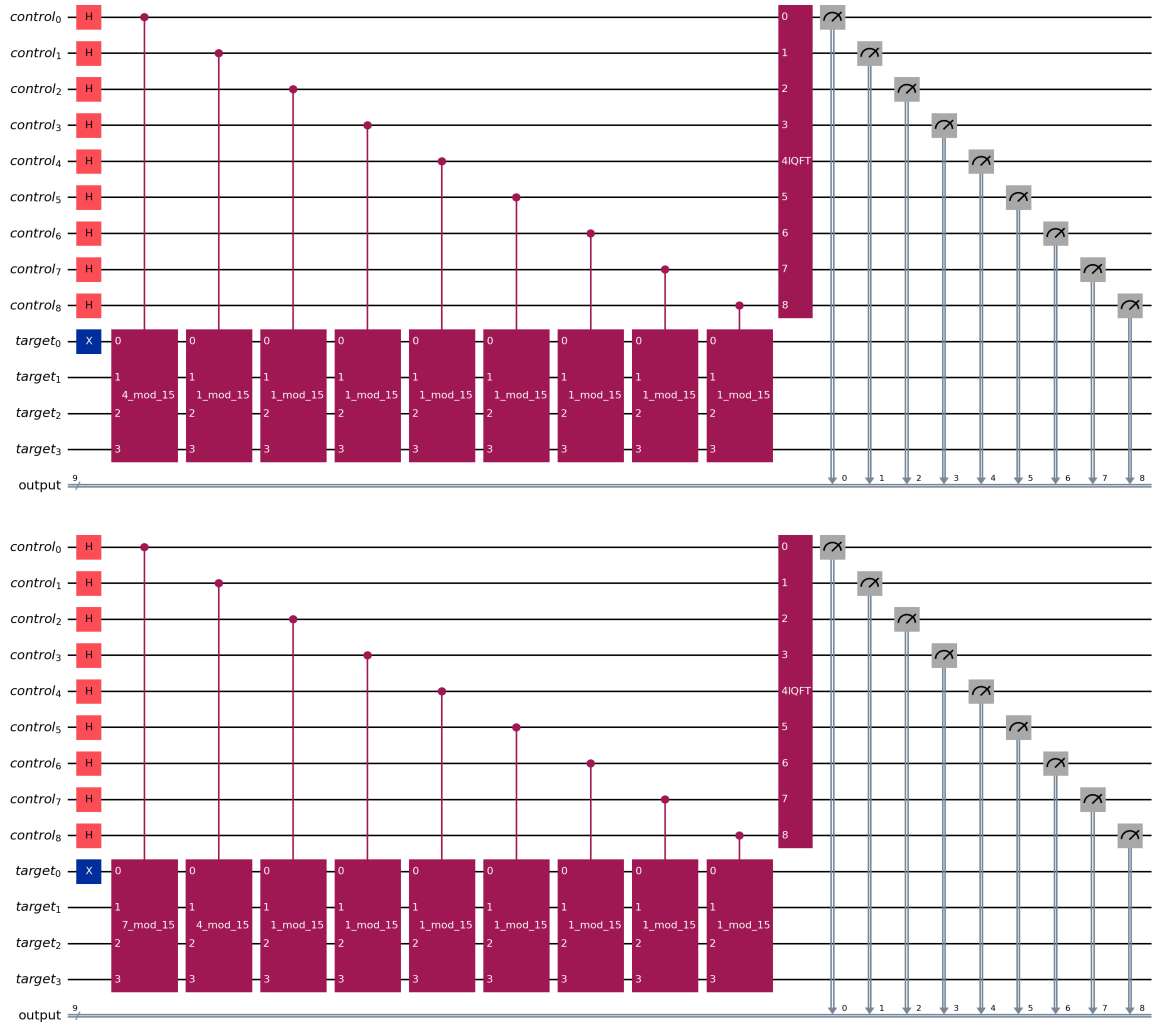

Figura 6.1: Diagrama esquemático del circuito `RegisterQC` para las bases $a = 4$ (superior) y $a = 7$ (inferior) con $N = 15$. La duplicación visible de bloques entrelazantes en $a = 7$ refleja directamente que su orden $r = 4$ activa dos potencias no triviales de U_a.

Tras transpilar a la base lógica $\{\text{CX}, \text{U3}\}$ sin restricción topológica, el costo entrelazante refleja con nitidez la asimetría anterior: las bases con $r = 2$ requieren del orden de 230 compuertas de dos qubits y ~ 460 capas de profundidad total, mientras que $a = 7$ demanda alrededor de

463 compuertas y $\sim$ 919 capas, prácticamente el doble (Tabla C.2). La QFT aporta apenas $\sim$ 30 capas a la profundidad de dos qubits, en concordancia con su complejidad asintótica $\mathcal{O}(t^2)$ frente a la $\mathcal{O}(L^3)$ de la exponenciación modular, que domina abrumadoramente el costo total (Tabla C.1).

6.1.2 Simulación ideal pura: línea base sin restricción topológica

La primera ejecución del circuito completo se realizó con conectividad *all-to-all*, es decir, sin restricción topológica. El propósito es doble: establecer la profundidad intrínseca del circuito y confirmar que, en ausencia de ruido, la probabilidad de señal alcanza el valor teórico del 100 %.

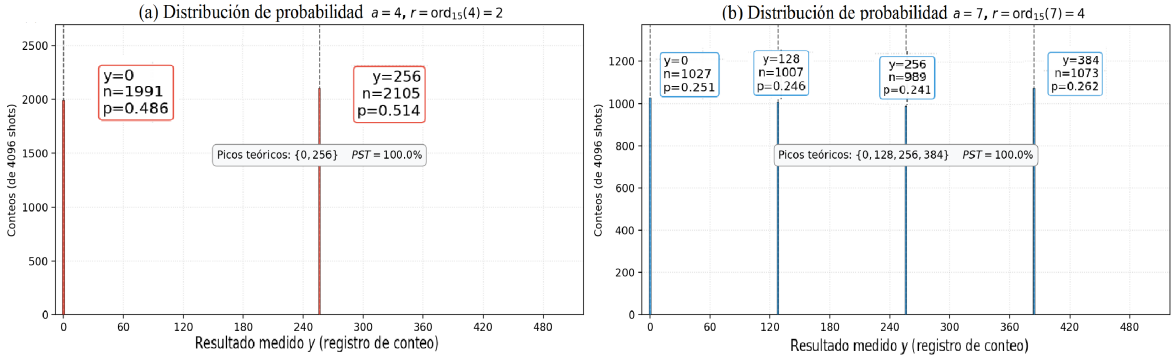

Figura 6.2: Histogramas de medición del registro de conteo en simulación ideal del algoritmo de Shor ($N = 15, t = 9, 4096$ *shots*) para $a = 4$ (a) y $a = 7$ (b), la distribución de las otras dos bases restantes es análoga a $a = 4$. Las líneas verticales punteadas marcan los picos teóricos y_s predichos por el QPE; las barras coloreadas muestran los conteos observados.

El resultado central es contundente: en las 16 configuraciones evaluadas (Tabla C.4), la totalidad de las $M = 4096$ mediciones cae sobre los picos teóricos del QPE, esto es, $PST = 100\,\%$ (Figura 6.3 (a)). Los histogramas de la Figura 6.2 muestran que la masa de probabilidad se concentra con probabilidad $\approx 1/2$ en cada uno de los dos picos $\{0, 256\}$ para las bases con $r = 2$ como $a = 4$, y con probabilidad $\approx 1/4$ en cada uno de los cuatro picos $\{0, 128, 256, 384\}$ para $a = 7$. Esta concordancia cuantitativa con la predicción del QPE valida que los operadores unitarios U_a y la $\text{QFT}^\dagger$ fueron implementados sin errores lógicos. Cualquier desviación observable respecto a este patrón ideal podrá imputarse, en las fases posteriores, al ruido de compuerta y la decoherencia (`FakeTorino`) o a los efectos adicionales del hardware real (*crosstalk*, errores coherentes, deriva de calibraciones).

Por otro lado, la profundidad de compuertas de dos qubits del circuito por base y nivel de optimización se muestra prácticamente invariante (Figura 6.3 (b)): las diferencias entre Opt=0 y Opt=3 son de a lo sumo unas pocas unidades. Esta invariancia es físicamente esperada en conectividad *all-to-all*: el transpilador no necesita insertar compuertas SWAP, y la cadena entrelazante proviene directamente de la descomposición de los U_a controlados, ya cercana

a su mínimo estructural. Lo que sí disminuye monótonamente con Opt es el conteo total de compuertas tal como se observa en la Figura 6.3 (c), debido a que se ejecutan pases de optimización a partir de Opt=1. Aislar esta contribución es metodológicamente relevante, pues permite atribuir cualquier variación posterior de D_{2Q} al proceso de ruteo.

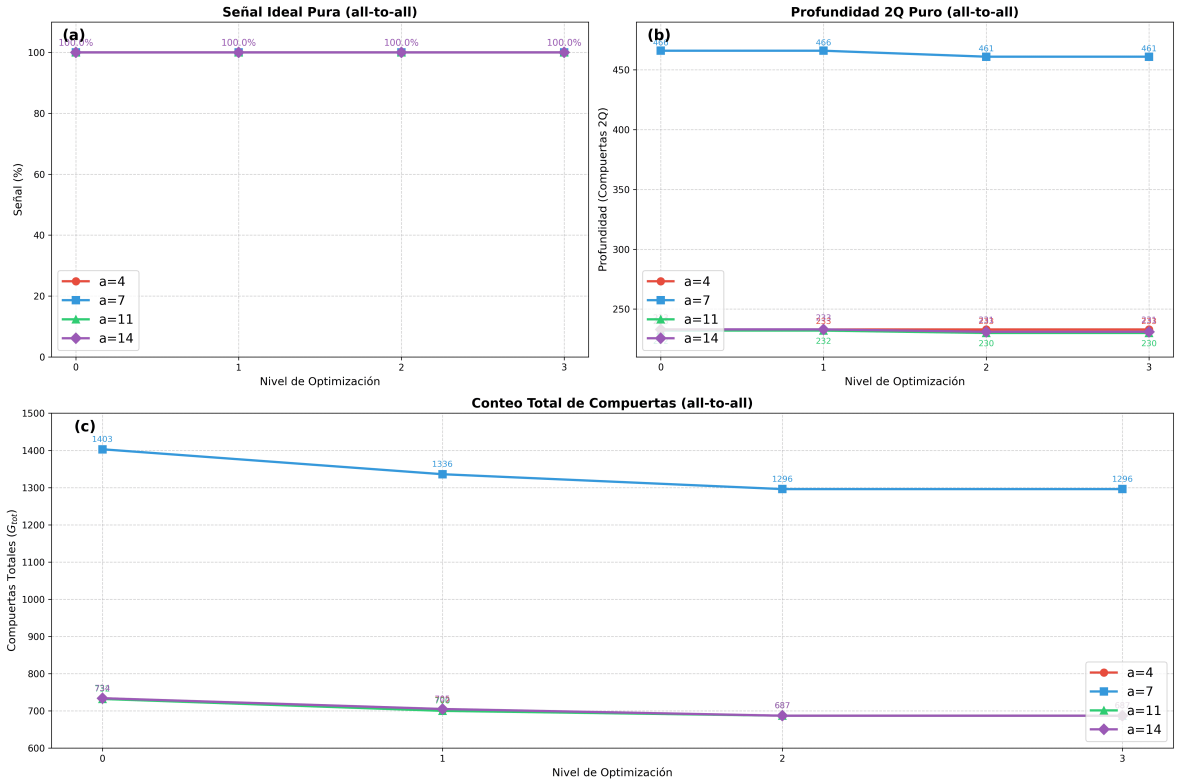

Figura 6.3: Resultados de métricas de circuitos para la simulación ideal del algoritmo de Shor ($N = 15$) asumiendo una topología de conectividad completa (*all-to-all*), evaluado en los niveles de optimización de transpilación del 0 al 3 para las bases coprimas $a \in \{4, 7, 11, 14\}$. **(a)** Señal ideal pura obtenida, demostrando que en ausencia de decoherencia y errores operacionales, la probabilidad teórica se mantiene en un 100 % invariable frente a la optimización. **(b)** Profundidad del circuito en función de las compuertas de dos qubits (2Q). **(c)** Conteo total de compuertas (G_{tot}) presentes en el circuito transpilado. Se destaca que la base $a = 7$ requiere un costo de recursos lógicos significativamente mayor en comparación con las bases $a = 4, 11$ y 14, las cuales presentan un comportamiento superpuesto y una ligera mitigación en la cantidad de compuertas a medida que se incrementa el nivel de optimización.

El caso $a = 14$: factores triviales

Para $a = 14 \equiv -1 \pmod{15}$ el circuito cuántico funciona correctamente (el QPE recupera $r = 2$ con señal del 100 %), pero la fase clásica de extracción produce factores triviales:

$$\gcd(14^{r/2} - 1, 15) = \gcd(13, 15) = 1, \qquad \gcd(14^{r/2} + 1, 15) = \gcd(15, 15) = 15. \quad (6.1)$$

Esta situación es una propiedad algebraica intrínseca del algoritmo de Shor, no un error computacional: cuando $a \equiv -1 \pmod{N}$, $a^{r/2} \equiv -1 \pmod{N}$ viola la condición necesaria $a^{r/2} \not\equiv -1 \pmod{N}$ del Algoritmo 3. En tal caso, el protocolo prescribe reiniciar con un nuevo a [24].

6.1.3 Transpilación a la topología de FakeTorino

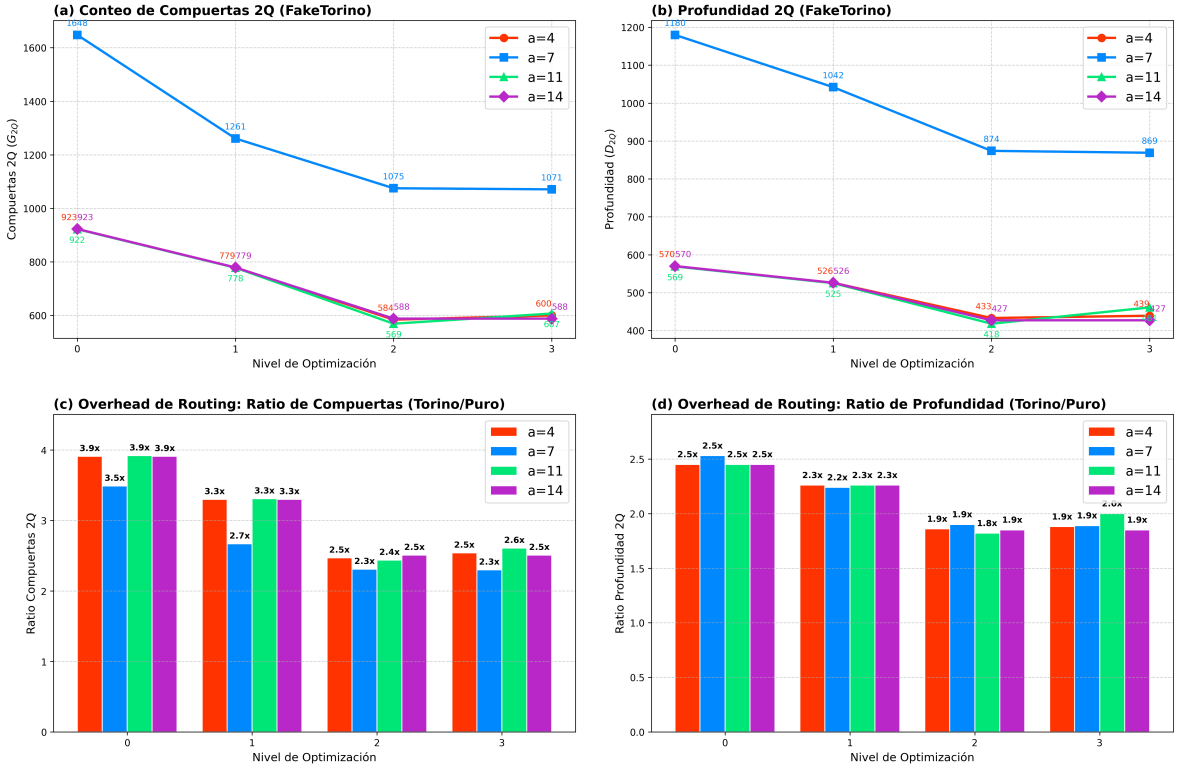

Figura 6.4: Evaluación del *overhead* de ruteo al transpilar el algoritmo para $N = 15$ hacia la topología Heavy-Hex de `FakeTorino`, en función de los cuatro niveles de optimización de Qiskit para las bases $a \in \{4, 7, 11, 14\}$. **(a)** Conteo absoluto de compuertas de dos qubits (G_{2Q}) resultantes tras la adaptación a la topología física. **(b)** Profundidad absoluta del circuito restringida a compuertas de dos qubits (D_{2Q}). **(c)** Factor de amplificación (ratio) del número de compuertas en comparación con la simulación ideal pura (*all-to-all*). **(d)** Factor de amplificación de la profundidad en comparación con el caso ideal. Se evidencia que los niveles de optimización superiores (2 y 3) reducen sustancialmente el costo introducido por la conectividad limitada del hardware.

La segunda fase del análisis transpila `RegisterQC` a la topología Heavy-Hex de `FakeTorino` y la ejecuta en modo ideal, sin modelo de ruido. Este experimento es metodológicamente crucial: aísla y cuantifica el *overhead* de profundidad que introduce el ruteo (la inserción de compuertas SWAP exigidas por la conectividad del chip), sin que ese efecto se confunda con

la degradación que el ruido producirá más adelante.

La Figura 6.4 (paneles (c) y (d)) hace visible el impacto de esta penalización arquitectónica. La métrica del factor de amplificación (Ratio D) sirve como evidencia cuantitativa directa del cuello de botella geométrico: ilustra el dramático sobrecosto multiplicativo requerido para adaptar el grafo del algoritmo a la conectividad de grado limitado del procesador. A pesar del incremento significativo en la profundidad producto de este *overhead* topológico, la señal PST se mantiene en el 100 % para todas las bases, confirmando que la inserción geométrica de SWAPs no introduce errores lógicos en ausencia de ruido (Tabla C.5). La discrepancia entre la profundidad ideal pura y la transpilada cuantifica el costo exclusivo del mapeo. Por otro lado, la profundidad y el número de compuertas de dos qubits disminuye con el nivel de optimización como se observa en la Figura 6.4 (paneles (a) y (b)), lo cual se atribuye a los pases heurísticos de *Opt* que buscan minimizar las secuencias de SWAP necesarias.

6.1.4 Impacto de la estrategia de mapeo: trivial vs. SABRE

Para evaluar cuantitativamente la efectividad del algoritmo SABRE frente al mapeo trivial ($\pi(q_i) = \text{qubit}_i$), se transpiló el circuito bajo ambas estrategias de *layout* para las cuatro bases y los cuatro niveles de optimización, manteniendo el ruteo SABRE en ambos casos (Tabla C.7). La probabilidad de señal se conservó en $PST = 100\,\%$ y la fidelidad de Hellinger en $\mathcal{F}_H \geq 0{,}9996$ para las 32 configuraciones (Figura 6.5 (d)), esta cota se adopta como umbral de referencia para las fases siguientes, cualquier configuración que produzca en `FakeTorino` o en `ibm_torino` una fidelidad sensiblemente inferior deberá atribuirse a ruido físico y no al tamaño finito de la muestra, lo que confirma un punto importante: en simulación ideal, la estrategia de asignación no afecta la corrección lógica del circuito, sino exclusivamente el costo físico de su ejecución.

La Figura 6.5 (a), (b) y (c) muestra que en todas las configuraciones SABRE mejora al mapeo trivial al disminuir la profundidad total, la profundidad y el número de compuertas de dos qubits, no obstante, la magnitud del beneficio depende de manera *no monótona* del nivel de optimización tal como se puede observar en la Figura 6.6, donde se cuantifica esta reducción calculando el error relativo porcentual del número y profundidad de compuertas de dos qubits con respecto al mapeo trivial ($\frac{trivial-SABRE}{trivial} \times 100$). Esta es, quizá, la observación más relevante de la sección. La leve regresión observada en algunos casos al pasar de Opt=2 a Opt=3 (por ejemplo en la Figura 6.6 (a), $a = 11$, donde la reducción de D_{2Q} cae del 17.7 % al 9.3 %) es consistente con la naturaleza estocástica de los pases adicionales activados en este nivel: Opt=3 no garantiza una solución superior a Opt=2; ambas deben entenderse como muestreos de una misma distribución de mapeos válidos. En consecuencia, para la práctica experimental esto demuestra que la optimización de topología no debe tratarse como un proceso de tipo 'caja negra' con garantía de mejora por nivel, sino que exige una iteración sistemática.

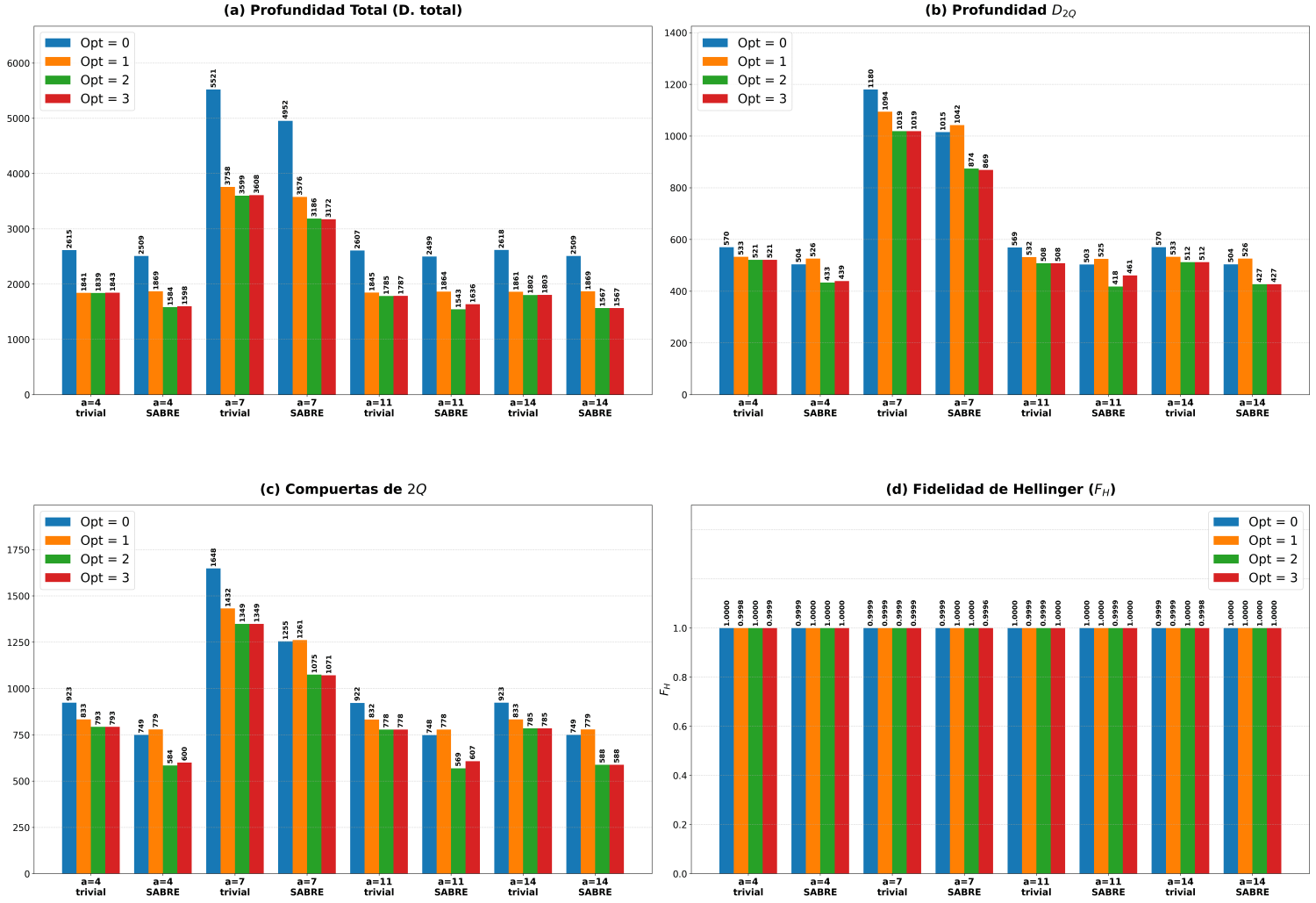

Figura 6.5: Comparación de las métricas de transpilación para el circuito con $N = 15$. Se evalúa el rendimiento del mapeo trivial frente al enrutamiento SABRE bajo distintos niveles de optimización (Opt) y valores base (a). Los paneles muestran: (a) la profundidad total del circuito transpilado, (b) la profundidad exclusiva de compuertas de dos qubits (D_{2Q}), (c) el conteo total de compuertas de dos qubits (G_{2Q}), y (d) la Fidelidad de Hellinger (F_H), la cual se mantiene estable e ideal en la mayoría de los casos.

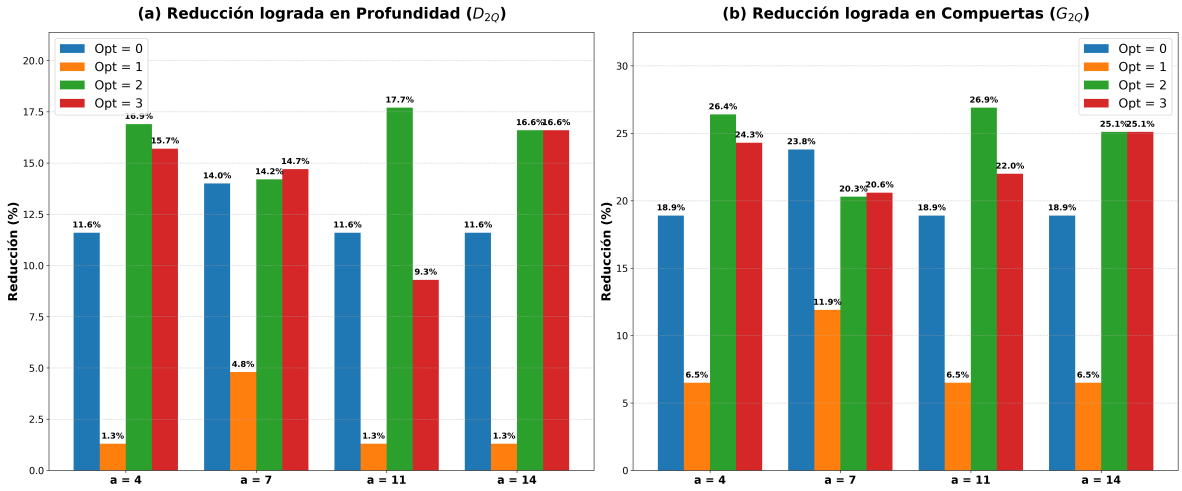

Figura 6.6: Reducción porcentual lograda por el algoritmo de enrutamiento SABRE respecto al mapeo trivial para $N = 15$. (a) Porcentaje de reducción en la profundidad de compuertas de dos qubits (D_{2Q}). (b) Porcentaje de reducción en el conteo de compuertas de dos qubits (G_{2Q}). Se destaca una dependencia no monótona respecto al nivel de optimización de Qiskit, observándose un mínimo local de mejora en Opt = 1 y el máximo desempeño general en Opt = 2.

6.1.5 Post-procesamiento clásico: extracción por fracciones continuas

Establecida en la Subsección 6.1.2, resta verificar que la cadena clásica descrita en la Sección 5.3 extrae efectivamente el período r y los factores no triviales de N para cada pico informativo ($y \neq 0$).

Tabla 6.1: Cadena de post-procesamiento clásico para los resultados con $y \neq 0$ obtenidos en simulación ideal pura ($N = 15$, $t = 9$, $M = 4096$ *shots*).

a	r	y	$\tilde{\varphi}$	**FC**	p_k/q_k	a^{q_k} **mód** N	**Resultado clásico**	
4	2	256	0.500	$[0;2]$	$1/2$	1	$\gcd = 3,\ 5$	✓
7	4	128	0.250	$[0;4]$	$1/4$	1	$\gcd = 3,\ 5$	✓
7	4	384	0.750	$[0;1,3]$	$3/4$	1	$\gcd = 3,\ 5$	✓
7	4	256	0.500	$[0;2]$	$1/2$	4 ($\neq 1$)	Convergente no válido	†
11	2	256	0.500	$[0;2]$	$1/2$	1	$\gcd = 5,\ 3$	✓
14	2	256	0.500	$[0;2]$	$1/2$	1	Triviales: $\gcd = 1,\ 15$	△

El protocolo recupera el orden correcto en todos los casos no degenerados: $r = 2$ para $a \in \{4, 11\}$ y $r = 4$ para $a = 7$, conduciendo en todos ellos a la factorización $15 = 3 \times 5$. Para $a = 14$ se recupera correctamente $r = 2$, pero la fase clásica produce factores triviales, ya discutido en la Subsección 6.1.2.

Existe, sin embargo, un único modo de falla intermedio que merece comentario por su valor pedagógico. Para $a = 7$ y la medición $y = 256$, la fase $\tilde{\varphi} = 1/2$ produce la expansión $[0; 2]$ y un único convergente no trivial $q_1 = 2$. Este denominador respeta la cota $1 < q_1 < N$ pero no sobrevive la verificación modular: 7^2 mód $15 = 4 \neq 1$. La explicación es directa: el valor $y = 256$ corresponde al índice $s = 2$ con $r = 4$, y la fracción $s/r = 2/4$ se reduce a $1/2$ en términos mínimos, por lo que el algoritmo de Euclides devuelve un *divisor* de r, no r mismo. Este caso justifica empíricamente por qué la verificación modular es un segundo criterio indispensable, y no una redundancia: aproximadamente el 24 % de las mediciones simuladas para $a = 7$ produce un q_k que respeta la cota de denominador pero no corresponde al orden buscado. En el experimento, este recurso no resultó necesario, pues las mediciones $y \in \{128, 384\}$ recuperan directamente $r = 4$ con probabilidad combinada $\approx 0{,}508$.

Un último elemento diferencia estructuralmente a $a = 7$ del resto. La recuperación de su período exige convergentes de profundidad $k = 2$ (la expansión $[0; 1, 3]$ que produce $3/4$), mientras que las bases con $r = 2$ se resuelven en $k = 1$. Como la precisión con que $\tilde{\varphi}$ aproxima s/r se degrada bajo ruido y los convergentes de mayor profundidad exigen mayor precisión de fase para satisfacer la cota del Teorema de Convergencia, $a = 7$ acumula dos fuentes simultáneas de vulnerabilidad: mayor profundidad del circuito cuántico y mayor profundidad del árbol clásico de convergentes. Esta doble sensibilidad anticipa que en las Secciones 6.2 y 6.3 $a = 7$ exhibirá la degradación más pronunciada del conjunto.

6.2 Resultados de la simulación con ruido: FakeTorino

Tras validar el circuito en condiciones ideales (Sección 6.1), se evalúa ahora cómo se comporta el algoritmo cuando se le somete a un modelo de ruido realista. Esta fase intermedia entre la simulación pura y el hardware constituye el primer encuentro de `RegisterQC` con la fragilidad del régimen NISQ: la pregunta ya no es si el circuito es lógicamente correcto, sino si la información cuántica sobrevive lo suficiente como para que el post-procesamiento clásico siga siendo capaz de extraer los factores de $N = 15$.

6.2.1 Choque con el ruido: comparación frente a la simulación ideal

Para esta fase se concentró el barrido en las dos bases más informativas, $a = 4$ ($r = 2$) y $a = 7$ ($r = 4$), evaluadas en los niveles extremos del transpilador (`opt=0` y `opt=3`). El contraste con la simulación ideal de la Sección 6.1 es inmediato. Allí los histogramas mostraban toda la masa de probabilidad alojada en los picos teóricos, con líneas verticales nítidas y el resto del eje rigurosamente vacío. Aquí (Figura 6.7), en cambio, los picos teóricos aparecen visiblemente *achicados* y, en su lugar, surge un *piso de ruido* continuo: una multitud de estados que en condiciones ideales eran imposibles ahora aparecen con probabilidad pequeña pero no nula, repartiendo masa por todo el registro de conteo. Esta migración de probabilidad desde los picos hacia el resto del espectro es la firma visual de la decoherencia y de los errores de compuerta acumulados a lo largo del circuito profundo, y se aprecia con claridad en el panel

(c) de la Figura 6.8, donde se junta el PST ideal con el ruidoso.

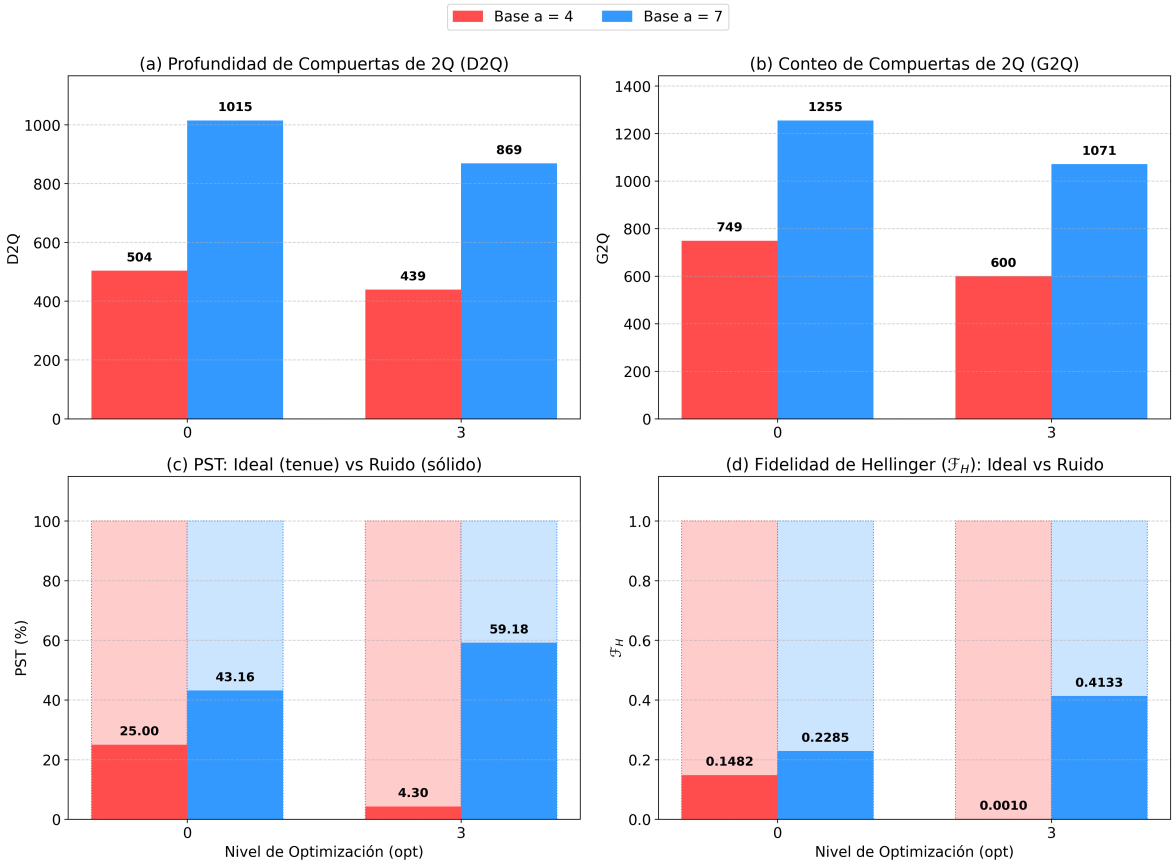

Figura 6.8: Resultados de métricas de rendimiento para la simulación con ruido (FakeTorino) del algoritmo de Shor ($N = 15$) utilizando las bases $a = 4$ (rojo) y $a = 7$ (azul), bajo los niveles de optimización de transpilación 0 y 3. **(a)** Profundidad del circuito en términos de compuertas de dos qubits (D2Q). **(b)** Conteo total de compuertas de dos qubits (G2Q) requeridas en los circuitos transpilados. **(c)** Probabilidad de Señal Teórica (PST), donde las barras sólidas indican los porcentajes obtenidos bajo el modelo de ruido y las áreas tenues punteadas representan el límite ideal del 100 %. **(d)** Fidelidad de Hellinger ($\mathcal{F}_H$) calculada entre las distribuciones de estados obtenidas y las teóricas; el marco tenue ilustra la fidelidad ($= 1{,}0$) correspondiente a un entorno sin ruido. Se observa de manera general que, aunque el nivel de optimización 3 reduce las métricas de profundidad y conteo de compuertas, esto no se traduce necesariamente en una mejora de la PST o la $\mathcal{F}_H$ para todas las bases bajo la influencia del ruido.

La caída cuantitativa es severa pero asimétrica entre bases, y se manifiesta de manera coherente en ambas métricas (Figura 6.8 (c) y (d)). Para $a = 4$ con `opt=3`, el PST cae del 100 % al $4{,}3$ % y $\mathcal{F}_H$ se desploma de $\geq 0{,}9996$ a $0{,}0010$: prácticamente toda la señal cuántica se ha disuelto en el piso de ruido y la distribución empírica es estadísticamente indistinguible del fondo uniforme sobre los $2^9 = 512$ resultados posibles. Para $a = 7$, en cambio, el PST se

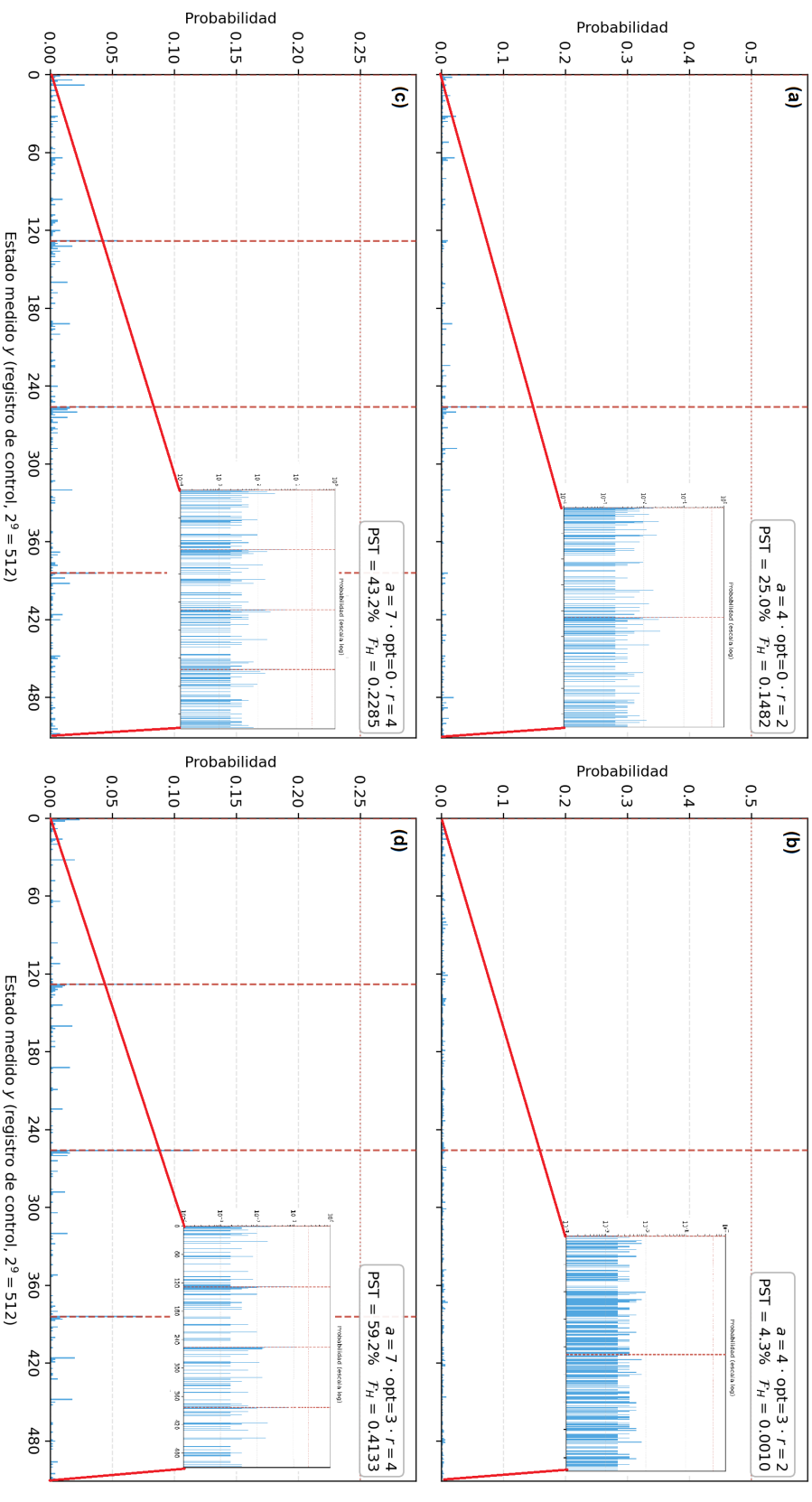

Figura 6.7: Histogramas del algoritmo de Shor ($N = 15$, $t = 9$, $M = 512$ *shots*) con `FakeTorino`. Se presentan cuatro configuraciones: (a) base $a = 4$, opt=0, $r = 2$; (b) base $a = 4$, opt=3, $r = 2$; (c) base $a = 7$, opt=0, $r = 4$; (d) base $a = 7$, opt=3, $r = 4$. Las líneas verticales discontinuas marcan los picos teóricos $y_s = s\,2^m/r$ y la horizontal punteada la altura teórica ideal $1/r$. Los recuadros internos reproducen cada distribución en escala logarítmica para hacer visible el *piso de ruido*.

mantiene cerca del $59\,\%$ con `opt=3` y la fidelidad asciende a $\mathcal{F}_H \approx 0{,}41$, valores sorprendentemente altos si se recuerda que su circuito tiene casi el doble de profundidad entrelazante.

Esta inversión (el circuito más profundo conserva mejor la señal) parece, a primera vista, paradójica, pero tiene una explicación enteramente geométrica. Para $a = 4$, la distribución teórica está concentrada en solo dos picos altos ($p = 0{,}5$ cada uno), de modo que cualquier fuga de probabilidad hacia el piso de ruido afecta proporcionalmente más a la señal y, en consecuencia, también a la fidelidad de la distribución completa. Para $a = 7$, en cambio, la distribución teórica ya está repartida entre cuatro picos más bajos ($p = 0{,}25$ cada uno), lo que la hace estructuralmente más resiliente al efecto uniformizante del ruido: la métrica $\mathcal{F}_H$ es sensible a la forma de la distribución de referencia, y una distribución teórica más uniforme es inherentemente más tolerante al ruido despolarizante, que tiende precisamente a uniformizar la distribución experimental. Esta asimetría es importante metodológicamente porque demuestra que la resiliencia al ruido no depende solo de la profundidad del circuito (que jugaría en contra de $a = 7$), sino también de la geometría de la distribución que se quiere medir (que esta vez juega a favor).

A pesar de esta degradación, el resultado más importante del experimento es positivo: el algoritmo clásico de fracciones continuas extrae correctamente los factores $\{3, 5\}$ en todas las configuraciones evaluadas, incluso en el caso extremo de $a = 4$ con $PST \approx 4\,\%$ y $\mathcal{F}_H \approx 0{,}001$ (Tabla C.9). Lo que se observa, en términos físicos, es que basta con que los picos teóricos asomen por encima del piso de ruido, aunque sea apenas, para que el filtro clásico sea capaz de seleccionarlos: la fidelidad de la distribución global puede haberse desplomado, pero la información mínima necesaria para identificar el período sigue presente en su soporte. La Sección 6.2.3 analiza con más detalle este mecanismo de rescate.

6.2.2 Efecto de las técnicas de supresión de errores

Para evaluar las estrategias de supresión formuladas en la Sección 5.6 se ejecutó el circuito en cuatro configuraciones: *baseline* (sin supresión), Desacoplamiento Dinámico (DD), Pauli Twirling (PT) y la combinación DD + PT. La Figura 6.9 sintetiza el experimento para ambas bases.

El resultado es, a primera vista, contraintuitivo: las técnicas de supresión no mejoran —e incluso degradan ligeramente— las métricas de señal en el simulador. Para $a = 7$, la línea base sin supresión alcanza $PST \approx 56{,}6\,\%$ y $\mathcal{F}_H \approx 0{,}40$, mientras que la combinación DD + PT reduce ambos indicadores a $PST \approx 46{,}5\,\%$ y $\mathcal{F}_H \approx 0{,}29$ (Tabla C.11). La caída concertada de las dos métricas en la misma dirección descarta que se trate de una anomalía: reflejan que, en el simulador estocástico, las técnicas de supresión añaden sobrecosto sin recuperar señal de manera sustancial.

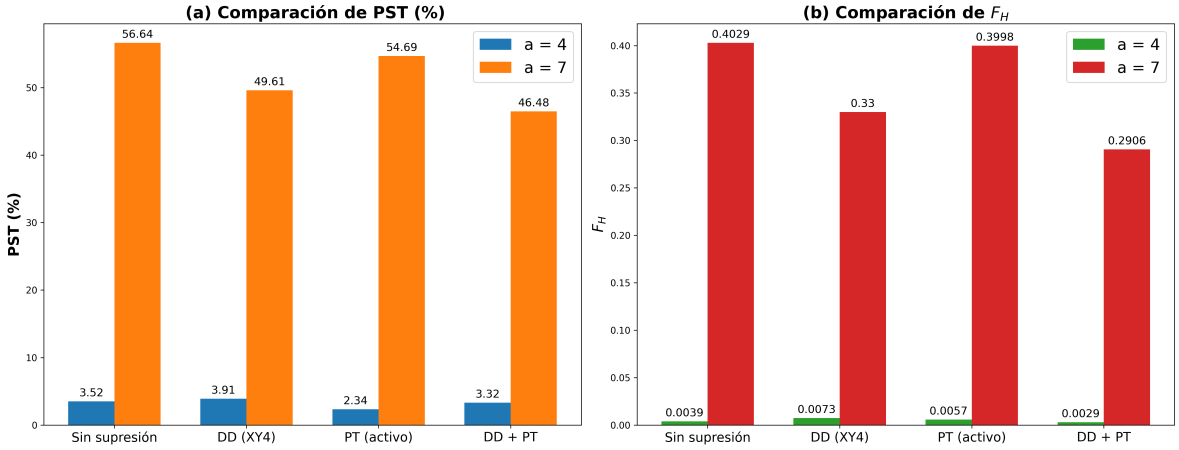

Figura 6.9: Evaluación comparativa de técnicas de supresión de errores en simulación ruidosa. El panel (a) muestra la probabilidad de éxito (PST) y el panel (b) detalla la fidelidad (F_H) para $a = 4$ y $a = 7$. En el eje horizontal se contrastan cuatro escenarios experimentales: ejecución estándar (Sin supresión), DD, PT, y la combinación de ambos métodos (DD + PT).

Lejos de ser un fracaso de las técnicas, este comportamiento es exactamente lo que cabía esperar y constituye una observación metodológica importante. Tanto el Desacoplamiento Dinámico como el Pauli Twirling están diseñados para combatir fenómenos físicos específicos del hardware real que el modelo de `FakeTorino` no reproduce, pues su ruido es puramente estocástico. Al insertar pulsos y compuertas auxiliares sobre un ruido que ya es estocástico, las técnicas no encuentran errores coherentes que estabilizar y, en cambio, alargan ligeramente el circuito, sumando una pequeña cantidad de error adicional que las dos métricas registran de manera consistente.

6.2.3 Post-procesamiento bajo ruido: la robustez del filtro clásico

El éxito de la extracción de factores en condiciones tan adversas merece una explicación detallada, porque ilustra una de las virtudes más importantes del algoritmo de Shor en el régimen NISQ: la asimetría entre lo cuántico (frágil) y lo clásico (robusto). Aun cuando el piso de ruido domina visualmente el histograma y la fidelidad de la distribución global cae varios órdenes de magnitud por debajo de la línea base ideal, las cuatro mediciones más probables para $a = 7$ con DD + PT siguen siendo, en orden de frecuencia, exactamente los cuatro picos teóricos del QPE: $y \in \{0, 256, 128, 384\}$, correspondientes a las fases $\varphi \in \{0,\ 1/2,\ 1/4,\ 3/4\}$. Juntas concentran cerca del 47 % de los *shots*; el resto de la masa se distribuye entre cientos de mediciones espurias, cada una de ellas con frecuencia individual muy baja.

Físicamente, lo que ocurre es que el ruido no es capaz de generar coherentemente una distribución de picos espurios consistente con un orden incorrecto: lo único que hace es desordenar la masa entre estados aislados, y esos estados aislados no satisfacen la verificación clásica. La

cuántica aporta el indicio (la concentración relativa en los picos correctos, capturada conjuntamente por PST y $\mathcal{F}_H$) y la clásica aporta el filtro (la verificación modular), de modo que el algoritmo completo sigue funcionando incluso cuando la calidad de la distribución cuántica es, en términos absolutos, muy pobre.

6.3 Ejecución experimental en hardware real: IBM Torino

En esta sección se presentan los resultados obtenidos al ejecutar el algoritmo de Shor para $N = 15$ en el procesador cuántico `ibm_torino`, perteneciente a la familia Heron r1 de IBM. Los experimentos constituyen la Fase 3 del diseño metodológico descrito en la Sección 5.4 y responden directamente al objetivo de ejecutar las versiones optimizadas del algoritmo en hardware real, aplicando técnicas de supresión de errores y cuantificando la degradación de la señal cuántica respecto a las simulaciones ideales y ruidosas.

6.3.1 Descripción de la ejecución experimental

Todos los circuitos fueron enviados al procesador `ibm_torino` a través del servicio IBM Quantum bajo el plan de acceso abierto (*Open Plan*), utilizando `Qiskit Runtime`. La Tabla 6.2 resume la configuración experimental base.

Tabla 6.2: Configuración del backend y parámetros experimentales.

Parámetro	**Valor**
Procesador (QPU)	`ibm_torino` (Heron r1, 133 qubits)
Topología	Heavy-Hex
Compuertas nativas 2Q	CZ
Compuertas nativas 1Q	RZ, SX, X
T_2 característico	$\sim 132\ \mu$s
Fidelidad mediana CZ	$\sim 99{,}5\,\%$ ($\epsilon_{\text{CZ}} \approx 5 \times 10^{-3}$)
Circuito lógico	RegisterQC ($t = 9$ qubits de control, 4 de trabajo)
N	15
`approximation_degree`	0.7
Shots por ejecución	4096
Semilla del transpilador	457

El parámetro `approximation_degree` $= 0{,}7$ del transpilador resultó determinante para la viabilidad de la ejecución: este valor permite al compilador realizar una síntesis unitaria aproximada, sacrificando algo de fidelidad a cambio de reducir la profundidad del circuito, en particular el número de compuertas CNOT. La Figura 6.10 ilustra cuantitativamente esta reducción para todas las bases del estudio: la profundidad 2Q se comprime aproximadamente en un orden de magnitud al pasar de `approx` $= 1{,}0$ a `approx` $= 0{,}7$.

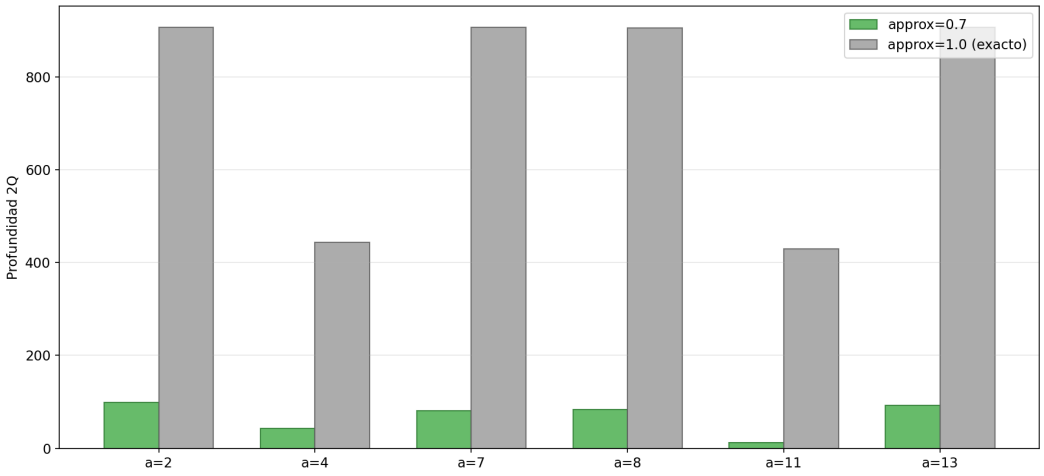

Figura 6.10: Impacto del parámetro `approximation_degree` sobre la profundidad de compuertas de dos qubits (D_{2Q}) tras la transpilación a la topología Heavy-Hex de `ibm_torino`, para las bases a ejecutadas en hardware. La configuración exacta (`approx` $= 1{,}0$) produce profundidades entre 430 y 910 capas CZ, mientras que la configuración aproximada (`approx` $= 0{,}7$) las reduce a entre 12 y 100 capas, una compresión de un orden de magnitud. Esta reducción es la que hace viable la ejecución dentro del tiempo de coherencia del procesador.

Organización del experimento

Por las restricciones de cupo del plan de acceso abierto (Sección 5.4), el barrido factorial completo definido en el Capítulo 5 no pudo replicarse de manera exhaustiva en hardware.

Tabla 6.3: Resumen de *jobs* ejecutados en `ibm_torino`.

Estudio	**Config.**	a	**Depth 2Q**	PST (%)	$\mathcal{F}_H$	**Factores**	**Job ID**
V1 (inicial)	PT, opt=3	4	116	75.5	—	3×5	`d673l15b`
V1 (inicial)	PT, opt=3	7	244	67.5	—	3×5	`d672p6gq`
V1 (inicial)	PT, opt=3	14	117	75.1	—	$1, 15$	`d672j2pv`
A2 (opt level)	opt=0	4	770	1.1	0.011	3×5	`d6agiong`
A2 (opt level)	opt=1	4	526	3.1	0.031	3×5	`d6agiu97`
A2 (opt level)	opt=2	4	116	70.8	0.529	3×5	`d6agj2p7`
A3 (base a)	PT, opt=3	2	225	42.4	0.243	3×5	`d6aghkvg`
A3 (base a)	PT, opt=3	8	212	79.4	0.484	3×5	`d6aghrvg`
A3 (base a)	PT, opt=3	11	75	35.5	0.309	5×3	`d6agi07g`
A3 (base a)	PT, opt=3	13	212	45.3	0.214	3×5	`d6agi595`
A6/A7 (supr.)	Sin supr	4	116	82.8	0.547	3×5	`d6agiah7`
A6/A7 (supr.)	Solo DD	4	116	20.3	0.154	No hallados	`d6agiesn`
A6/A7 (supr.)	PT + DD	4	116	0.8	0.007	3×5	`d6agik97`

En su lugar, los 13 *jobs* ejecutados se organizaron como cuatro estudios complementarios que, en conjunto, cubren las dimensiones más informativas del diseño: **V1** actúa como una cali-

bración inicial del flujo de envío, fijando $\texttt{opt} = 3$ con *Pauli Twirling* activado para tres bases representativas ($a \in \{4, 7, 14\}$); **A2** es un barrido del nivel de optimización del transpilador para la base óptima $a = 4$; **A3** es un barrido de la base a con $\texttt{opt} = 3$ fijo; y **A6/A7** es la sub-exploración del diseño 2×2 de técnicas de supresión sobre el circuito de $a = 4$, $\texttt{opt} = 3$. La organización detallada de los *scripts* y configuraciones empleados en cada estudio se documenta en el Apéndice B.5; los identificadores de cada *job* se registran en la Tabla 6.3 para garantizar la reproducibilidad de los resultados.

6.3.2 Resultados sin supresión de errores

En primer lugar se analizan los resultados obtenidos con la configuración base sin técnicas de supresión activas, es decir, sin *Dynamical Decoupling* ni *Pauli Twirling*, lo cual permite caracterizar el comportamiento intrínseco del procesador frente al algoritmo de Shor.

Impacto del nivel de optimización del transpilador

Según la Tabla 6.3, en el conjunto de configuraciones evaluadas, el nivel de optimización del transpilador de Qiskit fue la variable de control con mayor impacto observado sobre la señal experimental en hardware real, con saltos de aproximadamente dos órdenes de magnitud en PST entre opt=0 y opt=2 sobre la misma base $a = 4$. El estudio A2 fijó $a = 4$ y barrió formalmente $\texttt{opt} \in \{0, 1, 2\}$, manteniendo el resto de la configuración constante. La cuarta configuración ($\texttt{opt} = 3$) no se ejecutó como parte del estudio A2 propiamente dicho, sino que su valor de referencia para esta base se toma del estudio A6/A7 sin supresión ($PST = 82{,}8\,\%$, $\mathcal{F}_H = 0{,}547$, mismo circuito de 116 CZ que la transpilación con $\texttt{opt} = 2$).

Nota metodológica sobre comparaciones cruzadas: las cifras de PST citadas a continuación combinan ejecuciones de los estudios A2 (barrido de opt) y A6/A7 (barrido de supresión), realizadas en días distintos. Aunque el procesador y la transpilación nominal son idénticos, las calibraciones del backend pueden haber diferido entre ambas ventanas de ejecución. La magnitud absoluta del salto reportado debe entenderse, por tanto, como un *orden de magnitud robusto* del impacto del nivel de optimización del transpilador, y no como un coeficiente preciso reproducible al punto porcentual; en particular, una pequeña parte de la diferencia $\texttt{opt} = 2 \rightarrow \texttt{opt} = 3$ puede atribuirse a deriva natural de calibración y no exclusivamente al transpilador.

La transición del nivel 0 al nivel 2 produce una reducción del $84{,}9\,\%$ en la profundidad 2Q (de 770 a 116 capas CZ) y del $90{,}5\,\%$ en el número de compuertas CZ (de 1220 a 116). Esta reducción se traduce en un incremento de la señal PST de $1{,}1\,\%$ a $70{,}8\,\%$, y de la fidelidad de $\mathcal{F}_H = 0{,}011$ a $\mathcal{F}_H = 0{,}529$. El número de *outcomes* únicos (bitstrings distintos observados) disminuye de 512 a 57, lo cual refleja la concentración de la probabilidad en los picos teóricos en lugar de dispersarse uniformemente por ruido.

Con $\texttt{opt} = 0$ (layout trivial, routing *basic*), el circuito requiere 113 SWAPs estimados para adaptar la topología lógica a la malla Heavy-Hex del chip, cada uno descompuesto en 3 compuertas CZ. El circuito resultante ($D_{2Q} = 770$) excede ampliamente el umbral donde la acumulación multiplicativa de errores destruye la coherencia del estado:

$$\mathcal{F}_{\text{est}} \sim (1 - \epsilon_{\text{CZ}})^{1220} \approx (0{,}995)^{1220} \approx 0{,}002, \tag{6.2}$$

Aunque este valor es una estimación de fidelidad y no directamente de PST, ambos comparten el mismo régimen asintótico en este caso extremo: cuando $\mathcal{F}$ cae por debajo del 1 %, la distribución empírica es estadísticamente indistinguible del fondo uniforme sobre los 512 resultados posibles y, en consecuencia, el PST tiende a la línea base uniforme $r/2^t \approx 0{,}4\,\%$ para $r = 2$. La observación de $PST = 1{,}1\,\%$ es, por tanto, coherente con el régimen de fidelidad colapsada predicho por el modelo.

En contraste, $\texttt{opt} = 2$ con SABRE elimina completamente la necesidad de SWAPs adicionales y reduce la profundidad a 116 capas CZ. La estimación correspondiente es:

$$\mathcal{F}_{\text{est}} \sim (0{,}995)^{116} \approx 0{,}56, \tag{6.3}$$

valor coherente con la fidelidad observada $\mathcal{F}_H = 0{,}529$. Esto confirma que el modelo de decaimiento exponencial $\mathcal{F} \sim (1 - \epsilon_{2q})^{N_g}$ constituye una estimación razonable para circuitos de profundidad moderada en el procesador Heron r1. La transición de $\texttt{opt} = 2$ a $\texttt{opt} = 3$ aporta una mejora adicional modesta (de $70{,}8\,\%$ a $82{,}8\,\%$ en la línea base sin supresión de A6/A7), que puede ser atribuible al pase de re-síntesis con conciencia del ruido del *backend* que activa el nivel 3 y que reordena las cadenas de CZ minimizando los acoplamientos a qubits con peor fidelidad.

6.3.3 Efecto de la base a en la señal de hardware

El estudio A3 evaluó la señal experimental para cuatro bases coprimas adicionales de $N = 15$ ($a \in \{2, 8, 11, 13\}$), todas con $\texttt{opt} = 3$ y $\texttt{approx} = 0{,}7$. Los resultados se completan con las ejecuciones iniciales V1 ($a \in \{4, 7, 14\}$) y se sintetizan en la Tabla 6.4. Una precisión metodológica relevante para la lectura de esta tabla: tanto V1 como A3 se ejecutaron con *Pauli Twirling* activado como condición operativa por defecto del flujo de envío adoptado en este trabajo. En consecuencia, la caracterización por base no aísla el efecto de la base *en abstracto*, sino el efecto de la base *en presencia de PT*; esta distinción no compromete la comparación interna entre filas de la tabla (todas comparten la misma condición), pero sí debe tenerse presente al comparar estos números con la línea base sin supresión del estudio A6/A7 para $a = 4$ y al interpretar la magnitud absoluta del PST por base.

Los factores no triviales $\{3, 5\}$ fueron extraídos exitosamente en seis de las siete bases ejecutadas. La excepción, $a = 14$, produce factores triviales $\{1, 15\}$, como se demostró en la simulación ideal (Sección 6.1.2). La Figura 6.11 muestra las distribuciones empíricas obtenidas en hardware para estas cuatro bases superpuestas a la distribución teórica esperada. La

asimetría entre los picos puede atribuirse al sesgo de relajación térmica (T_1) del procesador, que favorece el decaimiento hacia el estado $|0\rangle^{\otimes n}$.

Tabla 6.4: Resultados por base a en hardware real (`ibm_torino`, `opt` $= 3$, `approx` $= 0{,}7$, PT activado en todos los *jobs*).

a	ord$(a, 15)$	Depth 2Q	CZ	PST_{HW} (%)	$\mathcal{F}_H^{\text{HW}}$	Factores
4	2	116	116	75.5	—	3×5
8	4	212	212	79.4	0.484	3×5
14	2	117	117	75.1	—	$1, 15$ (triviales)
7	4	244	244	67.5	—	3×5
13	4	212	212	45.3	0.214	3×5
2	4	225	225	42.4	0.243	3×5
11	2	75	76	35.5	0.309	5×3

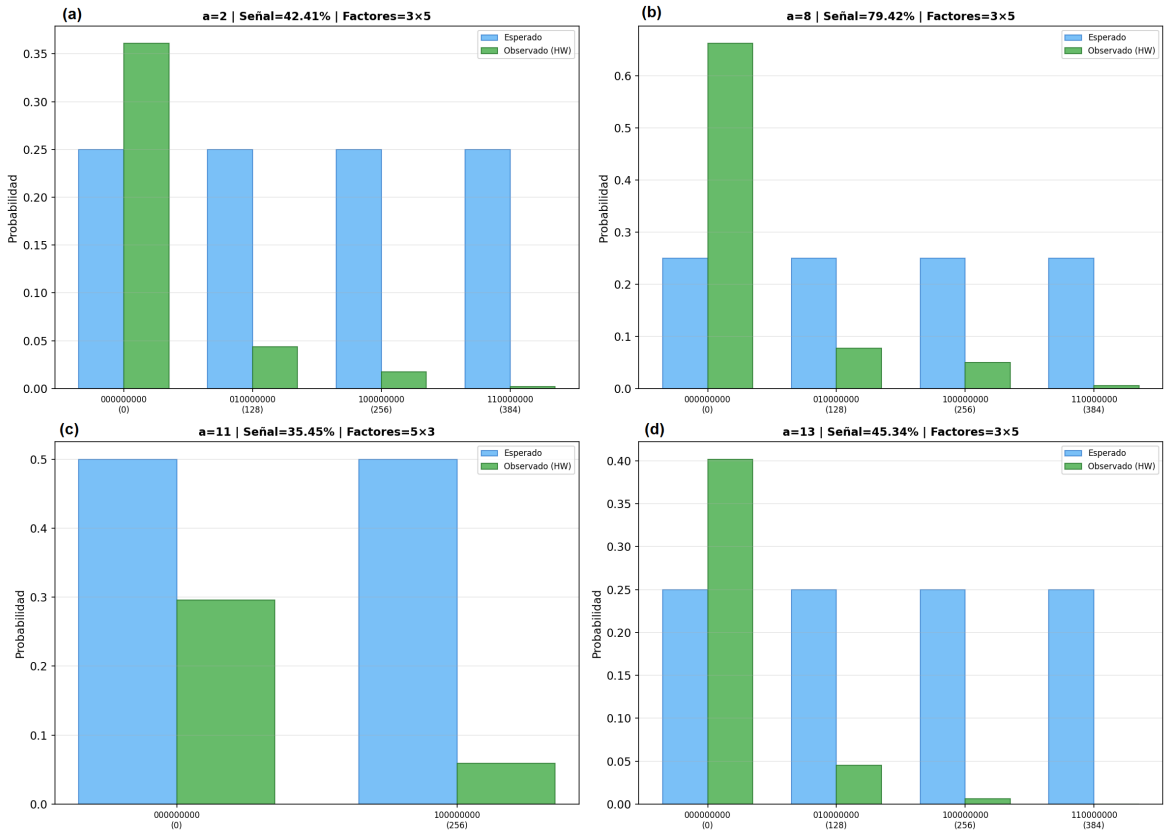

Figura 6.11: Distribuciones de probabilidad observadas en `ibm_torino` (verde) frente a la distribución teórica ideal (azul) para las cuatro bases del estudio A3. En cada panel se reportan únicamente los *bitstrings* asociados a las fases teóricas $\varphi = s/r$ del período correspondiente.

Según la Tabla 6.4, la mayor PST del estudio corresponde a $a = 8$ (79,4 %), y no a $a = 4$. Las tres bases con período $r = 2$ ejecutadas en hardware ($a = 4, 14, 11$) presentan PST que varían entre 35,5 % y 75,5 %, lo que muestra que la ventaja teórica de un período corto no garantiza por sí sola la mejor señal en el procesador. La correlación profundidad–señal no es estrictamente monótona: $a = 4$ y $a = 14$ comparten una profundidad similar (~ 116 CZ) y alcanzan PST muy próximos (75,5 % y 75,1 % respectivamente), pero $a = 8$ las supera con casi el doble de profundidad (212 CZ). Esto indica que la calidad de la señal también depende de la calibración local de los qubits asignados por el transpilador y de la propagación coherente de errores en cada cadena CZ. El caso de $a = 11$ resulta particularmente ilustrativo: a pesar de tener la menor profundidad 2Q del estudio (75 capas), su PST es solo del 35,5 %, lo que se puede atribuir a una variabilidad significativa en la fidelidad CZ del cableado físico seleccionado en esa ejecución concreta.

Caso óptimo: $a = 4$

El caso $a = 4$ ($r = 2$) resultó ser la elección óptima en general, para la demostración del algoritmo de Shor en hardware real. La Tabla 6.5 presenta los 10 *bitstrings* más frecuentes de las 4096 mediciones.

Tabla 6.5: Distribución de probabilidad para $a = 4$, $r = 2$ en `ibm_torino` (4096 *shots*). Los estados $|000000000\rangle$ y $|100000000\rangle$ corresponden a las fases teóricas $\varphi = 0$ y $\varphi = 1/2$, respectivamente.

#	$\lvert y\rangle$	Conteo	p_{obs}	$\varphi = y/2^9$	r_{cand}	Factores
1	$\lvert 000000000\rangle$	2897	0.707	0.000	—	—
2	$\lvert 010000000\rangle$	262	0.064	0.250	4	triviales
3	$\lvert 000100000\rangle$	202	0.049	0.063	—	—
4	$\lvert 100000000\rangle$	196	0.048	0.500	2	$\mathbf{3 \times 5}$
5	$\lvert 000000010\rangle$	110	0.027	0.004	—	—

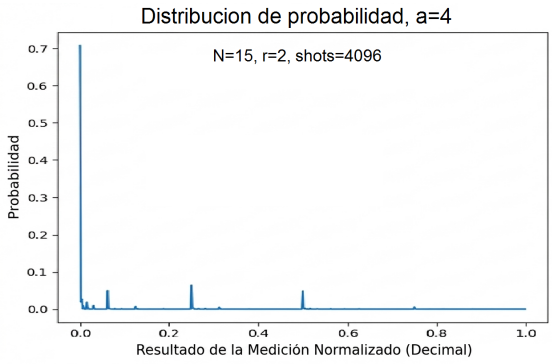

Figura 6.12: Distribuciones de probabilidad medidas en `ibm_torino` para $a = 4$ en el estudio inicial V1. El eje horizontal muestra el resultado de la medición normalizado al intervalo $[0, 1]$.

El pico dominante en $|000000000\rangle$ ($\varphi = 0$, conteo = 2897, $p = 70{,}7\,\%$) y el pico en $|100000000\rangle$ ($\varphi = 1/2$, conteo = 196, $p = 4{,}8\,\%$) corresponden a las fases teóricas del período $r = 2$. La señal total en estos dos picos, que define el PST para $r = 2$, suma $75{,}5\,\%$. A partir de la fase $1/2$ se obtiene la derivación clásica $\gcd(4^1 \pm 1, 15) = \{3, 5\}$, que permanece inalterada.

Viabilidad temporal del circuito

Un criterio fundamental para la ejecución en hardware NISQ es que la duración del circuito sea significativamente menor que los tiempos de coherencia del procesador. La duración total del circuito puede estimarse [20] considerando que las compuertas CZ dominan el tiempo de circuito:

$$T_{\text{circuito}} \approx D_{2Q} \times \tau_{\text{CZ}} \approx 116 \times 68 \text{ ns} \approx 7{,}9\ \mu\text{s}, \tag{6.4}$$

donde $\tau_{\text{CZ}} \approx 68$ ns es la duración típica de una compuerta CZ en el procesador Heron r1. El cociente $T_{\text{circuito}}/T_2 \approx 7{,}9/132 \approx 0{,}06$ confirma que el circuito opera holgadamente dentro del tiempo de coherencia. Para $a = 7$ ($D_{2Q} = 244$), esta razón asciende a $\sim 0{,}13$, aún holgadamente dentro del margen viable, siendo entonces el error de compuerta, y no la decoherencia, el factor limitante. El tiempo cuántico real reportado por el *runtime* fue de 3 segundos para todos los *jobs*, lo que incluye la inicialización de qubits, la ejecución del circuito y la lectura de 4096 *shots*.

6.3.4 Impacto de las técnicas de supresión de errores

El estudio A6/A7 evaluó el efecto del *Dynamical Decoupling* (DD) y el *Pauli Twirling* (PT) sobre la señal experimental en hardware real, comparando tres configuraciones contra la línea base sin supresión, todas con el mismo circuito ($a = 4$). Los resultados se presentan en la Tabla 6.6.

Tabla 6.6: Comparación de técnicas de supresión de errores en hardware real (`ibm_torino`, $a = 4$, `opt` $= 3$, `approx` $= 0{,}7$, 116 CZ, 4096 *shots*). Se reportan la señal PST, la fidelidad $\mathcal{F}_H$, el número de *outcomes* únicos y los factores extraídos.

Configuración	PST (%)	$\mathcal{F}_H$	Outcomes	Factores
Sin supresión	**82.8**	**0.547**	30	3×5
Solo DD (XY4)	20.3	0.154	163	No hallados
PT + DD (XY4)	0.8	0.007	469	3×5

Desacoplamiento Dinámico

El resultado más destacado (y en apariencia contraintuitivo) de la Tabla 6.6 es que el *Dynamical Decoupling* con secuencia XY4 degradó severamente la señal experimental: de $PST = 82{,}8\,\%$ a $PST = 20{,}3\,\%$, una caída de 62.5 puntos porcentuales. La fidelidad se redujo de $\mathcal{F}_H = 0{,}547$ a $\mathcal{F}_H = 0{,}154$, y el número de *outcomes* únicos aumentó de 30 a 163,

indicando una dispersión sustancial de la probabilidad fuera de los picos teóricos. En esta configuración, el algoritmo fue incapaz de extraer los factores correctos.

Este comportamiento se explica por la interacción entre la secuencia DD y la estructura temporal del circuito en el procesador Heron r1. El DD inserta pulsos compensatorios (X-Y-X-Y en el caso de XY4) durante los periodos de inactividad (*idle times*) de los qubits. Sin embargo, para un circuito de profundidad moderada (116 capas CZ) en un procesador con compuertas nativas de alta fidelidad ($\epsilon_{\text{CZ}} \approx 5 \times 10^{-3}$), los tiempos de inactividad son breves y la decoherencia acumulada durante estos intervalos no constituye la fuente dominante de error. En este régimen, los pulsos DD adicionales introducen errores de calibración propios — cada pulso de compensación posee un error finito de implementación — que exceden el beneficio de la refocalización del estado. Adicionalmente, la inserción de pulsos DD puede provocar *crosstalk* con qubits adyacentes en la topología Heavy-Hex, amplificando el ruido correlacionado.

Pauli Twirling

El protocolo experimental del Capítulo 5 (Sección 5.6) prescribe un diseño factorial 2×2 que cruza la activación o desactivación de DD y PT, dando lugar a cuatro celdas: *sin supresión*, *solo DD*, *solo PT* y *DD+PT*. En el simulador con ruido (Fase 2) las cuatro celdas se ejecutaron formalmente bajo idénticas condiciones (Sección 6.2.2). En hardware, en cambio, el cupo del plan abierto solo permitió ejecutar tres celdas formales dentro del estudio A6/A7 (*sin supresión*, *solo DD* y *DD+PT*, todas reportadas en la Tabla 6.6); la cuarta celda, *solo PT*, se infiere indirectamente comparando la línea base sin supresión de A6/A7 ($PST = 82{,}8\,\%$) con la ejecución V1 inicial (PT activado, sin DD, $PST = 75{,}5\,\%$) sobre el mismo circuito de 116 CZ:

Tabla 6.7: Efecto del *Pauli Twirling* para $a = 4$ en `ibm_torino` (116 CZ, 4096 *shots*).

Configuración	PST (%)	Factores
Sin supresión (A6/A7)	82.8	3×5
Solo PT (V1, inferida)	75.5	3×5

La celda solo PT inferida de V1 da $PST = 75{,}5\,\%$, ligeramente por debajo de la línea base sin supresión ($82{,}8\,\%$). La diferencia, de $\Delta PST \approx -7$ pp, debe leerse con cautela: V1 y A6/A7 fueron ejecutados en fechas distintas, con calibraciones del procesador potencialmente diferentes, y la deriva natural del *backend* entre dos ventanas de calibración produce, en circuitos comparables, fluctuaciones que pueden acumular varios puntos porcentuales incluso bajo condiciones nominalmente idénticas. En consecuencia, parte de la brecha observada puede atribuirse a deriva de calibración y no exclusivamente al efecto del PT, y los 4096 *shots* disponibles no permiten afirmar significancia estadística sobre el signo del efecto. La conclusión cualitativa, no obstante, es que *el PT no produce una mejora robusta de la señal en este régimen*, en línea con la observación previa del simulador con ruido: para un circuito de profundidad $D_{2Q} = 116$ en el procesador Heron r1, la contribución de los errores coherentes

al error total es modesta, de modo que la conversión que el PT realiza de error coherente en ruido estocástico no se traduce en un beneficio observable sobre el PST.

Combinación PT + DD

La combinación de ambas técnicas produjo el peor resultado de todas las configuraciones: $PST = 0{,}76\,\%$ con $\mathcal{F}_H = 0{,}007$, una distribución efectivamente indistinguible del ruido uniforme. Se observaron 469 *outcomes* únicos de los 512 posibles, lo cual confirma la pérdida casi total de la estructura periódica del QPE. El hecho de que los factores $\{3, 5\}$ fueran formalmente extraíbles en esta configuración (por la coincidencia estadística de que $r = 2$ fue identificado como candidato) no debe interpretarse como un éxito: la señal en los picos teóricos ($0{,}22\,\%$ en $|000000000\rangle$ y $0{,}54\,\%$ en $|100000000\rangle$) es despreciable frente al fondo de ruido.

La extracción formal de factores con DD+PT (Tabla 6.6) no debe interpretarse como un éxito algorítmico: con un PST por debajo del $1\,\%$ la distribución es estadísticamente indistinguible del ruido uniforme, y la coincidencia entre el modo más frecuente y un *bitstring* compatible con $r = 2$ es estadísticamente plausible bajo aleatoriedad sobre 512 resultados. Por contraste, la configuración solo DD, con PST mayor ($20{,}3\,\%$), presentó un modo dominante en un *bitstring* espurio respecto al cual las fracciones continuas no recuperan el período correcto. Ambos casos ilustran que PST y éxito clásico no están unívocamente acoplados en el régimen de señal colapsada: el filtro de fracciones continuas puede devolver factores correctos por coincidencia cuando la distribución es casi uniforme, y puede no devolverlos cuando la distribución tiene moda fuera de los picos teóricos.

Este resultado demuestra que la combinación PT + DD no es aditiva en sus beneficios: el DD amplifica el ruido por los mecanismos ya descritos, y el PT, al aleatorizar las compuertas de Pauli alrededor de cada CZ, modifica la estructura de cancelación de errores que el circuito original podría exhibir naturalmente, eliminando posibles interferencias destructivas de errores que serían favorables.

Contraste con la simulación ruidosa

La Tabla 6.8 contrasta el efecto de las cuatro variantes de supresión entre la simulación con ruido FakeTorino (Sección 6.2.2) y la ejecución en hardware real, sobre el mismo circuito $a = 4$, `opt` $= 3$. La comparación es aproximadamente directa porque el *snapshot* de FakeTorino se construyó a partir del propio procesador `ibm_torino`, de modo que la diferencia entre las dos columnas mide específicamente lo que el *fake backend* no modela.

En el simulador FakeTorino, las cuatro variantes producen señales prácticamente indistinguibles entre sí ($PST \in [2{,}34, 3{,}91]\,\%$, $\mathcal{F}_H \in [0{,}003, 0{,}007]$), confirmando la observación de la Sección 6.2.2: el modelo de ruido del *fake backend* es estocástico.

Tabla 6.8: Contraste del efecto de la supresión entre el simulador ruidoso (FakeTorino, Fase 2) y el hardware real (`ibm_torino`, Fase 3), para $a = 4$, `opt` $= 3$. La celda *Solo PT* en HW se obtiene por inferencia indirecta de V1. *Nota:* los valores de FakeTorino reportados aquí provienen de la corrida del estudio de supresión (Tabla C.11); por eso, para la fila *sin supresión*, se reproduce el par $(PST_{\text{FT}}, \mathcal{F}_H^{\text{FT}}) = (3{,}52\,\%,\ 0{,}0039)$ del estudio de supresión, en lugar del par $(4{,}30\,\%,\ 0{,}0010)$ que reporta la Tabla C.9 (corrida independiente con la misma configuración nominal).

Configuración	PST_{FT} (%)	$\mathcal{F}_H^{\text{FT}}$	PST_{HW} (%)	$\mathcal{F}_H^{\text{HW}}$
Sin supresión	3.52	0.0039	**82.8**	**0.547**
Solo DD	3.91	0.0073	20.3	0.154
Solo PT	2.34	0.0057	75.5 (V1)	—
DD + PT	3.32	0.0029	0.8	0.007

En hardware, en cambio, las diferencias entre celdas son de aproximadamente un orden de magnitud: la línea base sin supresión alcanza el $82{,}8\,\%$, el PT solo se sitúa ligeramente por debajo ($75{,}5\,\%$, atribuible en parte a deriva de calibración), el DD la colapsa al $20{,}3\,\%$ y la combinación DD+PT la lleva por debajo del $1\,\%$. Esta sensibilidad es exactamente la firma del régimen NISQ que el *fake backend* no captura: una vez que el ruido coherente y el *crosstalk* entran en juego, la elección de la técnica de supresión deja de ser irrelevante y pasa a ser determinante.

6.3.5 Análisis y discusión

Degradación de la señal: simulación ideal → FakeTorino → IBM Torino

La Tabla 6.9 cuantifica la degradación de la señal cuántica a lo largo de los tres entornos de ejecución. Para las bases que cuentan con medición tanto en Fase 2 como en Fase 3 ($a \in \{4, 7\}$), la lectura de la cascada es directa; para las bases incluidas únicamente en el barrido de hardware ($a \in \{2, 8, 11, 13\}$) la columna FakeTorino se marca con "—", indicando ausencia de dato y no inferencia.

El hallazgo central: el *fake backend* sobreestima la degradación en el régimen de circuitos cortos. En las dos bases con medición en ambos entornos, el hardware real *supera* a la simulación con ruido. Para $a = 4$ (116 CZ en HW), la inversión es marcada: $PST_{\text{FT}} = 4{,}3\,\%$ frente a $PST_{\text{HW}} = 75{,}5\,\%$. Para $a = 7$ (244 CZ en HW), la inversión es leve pero del mismo signo: $PST_{\text{FT}} = 59{,}2\,\%$ frente a $PST_{\text{HW}} = 67{,}5\,\%$. La consistencia de la dirección sugiere que se trata de un patrón sistemático, no de una fluctuación, y admite una explicación física en tres componentes que no son mutuamente excluyentes:

Tabla 6.9: Degradación de la señal PST (%) desde la simulación ideal hasta el hardware real, `opt` $= 3$, `approx` $= 0{,}7$. La columna FakeTorino corresponde al *baseline* sin supresión de la Fase 2 y solo está disponible para $a \in \{4, 7\}$. La columna HW agrupa V1 ($a = 4, 7$, PT activado) y A3 ($a \in \{2, 8, 11, 13\}$, PT activado).

a	r	PST_{ideal}	PST_{FT}	PST_{HW}	$\Delta_{\text{ideal}\to\text{FT}}$	$\Delta_{\text{ideal}\to\text{HW}}$	$\mathcal{F}_H^{\text{HW}}$
4	2	100.0	4.3	75.5	95.7	24.5	—
7	4	100.0	59.2	67.5	40.8	32.5	—
8	4	100.0	—	79.4	—	20.6	0.484
2	4	100.0	—	42.4	—	57.6	0.243
11	2	100.0	—	35.5	—	64.5	0.309
13	4	100.0	—	45.3	—	54.7	0.214
Promedio		100.0	31.8*	57.6	68.3*	42.4	0.310**

* Promedio sobre las dos bases con dato FT. ** Promedio sobre las cuatro bases con $\mathcal{F}_H$ reportada.

1. Como se declaró en la Sección 5.4, el *snapshot* de FakeTorino, es por construcción, un modelo de ruido *aditivo y estocástico*: cada compuerta degrada la señal multiplicativamente y los errores no se cancelan entre sí, lo que tiende a producir cotas pesimistas en circuitos donde la compensación coherente es posible.

2. El *snapshot* reporta tasas de error de la ventana de calibración en que fue tomado; la ejecución en `ibm_torino` se realizó con la calibración vigente del día, que para los qubits y enlaces seleccionados por SABRE pudo presentar fidelidades CZ superiores a las del *snapshot*. La deriva natural de calibración entre dos ventanas, lejos de ser un artefacto, es la regla en NISQ.

3. En un circuito como el de $a = 4$ con 116 CZ, las sobre-rotaciones sistemáticas y las fases espúreas asociadas a cada compuerta pueden interferir destructivamente entre sí en la cadena, recuperando parte de la señal que el modelo estocástico, por construcción, da por perdida. Este mecanismo se atenúa progresivamente al aumentar la profundidad, lo cual es consistente con la observación empírica: la inversión es muy fuerte en $a = 4$ ($\Delta_{\text{HW-FT}} \approx +71$ pp, 116 CZ) y se reduce a un valor pequeño en $a = 7$ ($\Delta_{\text{HW-FT}} \approx +8$ pp, 244 CZ).

La lectura conjunta de estos tres factores conduce a un hallazgo metodológico de interés para futuros trabajos en NISQ: *en las dos bases con medición pareada disponibles, el Fake Backend se comporta como predictor pesimista de su propio hardware en el régimen de circuitos cortos estudiado*. Esta observación no debe extrapolarse sin nueva evidencia a circuitos más profundos, a otras bases o a otras configuraciones del modelo de ruido. Cualquier estimación de viabilidad basada en el simulador con ruido podría, en el régimen aquí caracterizado, subestimar la señal real cuando el circuito quepa cómodamente dentro del tiempo de coherencia, y las decisiones de ingeniería tomadas exclusivamente sobre el simulador (como descartar una base a por baja PST simulada) podrían ser, en ese régimen,

injustificadamente conservadoras.

Comportamiento de las bases sin medición en Fase 2: para $a \in \{2, 8, 11, 13\}$ no se dispone de medición en FakeTorino, lo que impide cuantificar la cascada completa. La columna PST_{HW} permite, no obstante, ordenar las bases por dificultad efectiva en hardware: $a = 8$ y $a = 4$ encabezan el ranking con $PST > 75\,\%$, mientras que $a = 11$ ($D_{2Q} = 75$, la menor del estudio) aparece en el extremo opuesto con $PST = 35{,}5\,\%$. Esta no-monotonía respecto a la profundidad fue ya discutida y puede ser atribuida a la heterogeneidad de la fidelidad CZ a lo largo de las cadenas Heavy-Hex que SABRE selecciona en cada ejecución.

La señal promedio en hardware real ($\overline{PST}_{\text{HW}} = 57{,}6\,\%$ sobre las seis bases con factores no triviales) es suficiente para la extracción exitosa de los factores de $N = 15$ en todos los casos, confirmando que el algoritmo de Shor opera en un régimen donde la señal cuántica es distinguible del ruido para esta instancia pedagógica.

Capítulo 7

Conclusiones

El presente trabajo de grado se propuso evaluar la viabilidad técnica y el desempeño del algoritmo de Shor para la factorización del entero $N = 15$ en procesadores cuánticos NISQ de IBM, comparando sistemáticamente los resultados obtenidos en tres entornos de ejecución: simulación ideal, simuladores con modelos de ruido calibrados (*Fake Backends*) y el procesador real `ibm_torino` (familia Heron r1, 133 qubits superconductores). A lo largo de los capítulos precedentes se documentó la implementación completa del algoritmo (desde los fundamentos teóricos hasta la ejecución experimental en hardware), generando evidencia cuantitativa sobre la degradación de la señal cuántica y la efectividad de las estrategias de optimización y supresión de errores. En este capítulo final se sintetizan los hallazgos principales, se discuten las limitaciones del estudio y se proponen perspectivas futuras.

7.1 Resultados principales

La ejecución exitosa de una versión compilada del algoritmo de Shor para $N = 15$ se demostró experimentalmente en el procesador `ibm_torino`, recuperando los factores 3×5 con una señal máxima de $PST = 82{,}8\,\%$, correspondiente a la línea base sin supresión del estudio A6/A7 (`job_id d6agiah7`, $\mathcal{F}_H = 0{,}547$), y con una tasa de éxito cercana al $90\,\%$ en la extracción de los factores no triviales sobre las ejecuciones con bases coprimas válidas (Tabla 6.3). Hasta donde se tiene conocimiento, ésta constituye un primer paso en la demostración experimental de una versión compilada del algoritmo de Shor en hardware cuántico real reportada desde la Universidad de Nariño.

La Transformada Cuántica de Fourier ($n = 9$ qubits) y los bloques controlados de exponenciación modular se implementaron en Qiskit y se verificaron numéricamente con errores de unitariedad $\|U^\dagger U - I\|_F < 10^{-13}$ en todas las variantes evaluadas (Tabla C.1). El mapeo a la topología Heavy-Hex de FakeTorino evidenció un *overhead* sobre el conteo de compuertas de dos qubits que llega hasta un factor de $\sim 3{,}9\times$ con mapeo trivial respecto a la conectividad *all-to-all* (Tabla C.6). Sobre esa base, el algoritmo SABRE redujo de forma sistemática el sobrecosto, con disminuciones de entre el $11\,\%$ y el $18\,\%$ en profundidad de dos qubits y

entre el $19\,\%$ y el $27\,\%$ en el conteo de compuertas de dos qubits para las bases evaluadas (Tabla C.8).

Más allá de la transpilación, en el conjunto de configuraciones evaluadas el nivel de optimización del transpilador de Qiskit fue la variable de control con *mayor impacto observado* sobre la señal experimental. El estudio A2 (Tabla C.12, Sección 6.3.2) registró una transición de $PST = 1{,}1\,\%$ (`opt` $= 0$, 770 capas CZ y 1220 compuertas CZ con 113 SWAPs estimadas) a $PST = 70{,}8\,\%$ (`opt` $= 2$, 116 capas CZ sin SWAPs adicionales), una ganancia de aproximadamente dos órdenes de magnitud en señal obtenida al sólo modificar este parámetro. La transición adicional de `opt` $= 2$ a `opt` $= 3$ con *Pauli Twirling* desactivado (línea base de A6/A7) elevó la señal hasta el máximo de $82{,}8\,\%$, atribuible al pase de re-síntesis con conciencia del ruido del *backend*. Conviene recordar que parte de la diferencia entre `opt` $= 2$ (estudio A2) y `opt` $= 3$ (estudio A6/A7) corresponde a ventanas de calibración distintas. Esta evidencia cuantifica por qué la compilación cuántica no constituye una optimización accesoria, sino una pieza estructural del algoritmo en la era NISQ.

La estrategia de pre-procesamiento clásico propuesta en la literatura reciente [20] se validó experimentalmente bajo condiciones equivalentes. El módulo desarrollado en Python caracterizó exhaustivamente el grupo $\mathbb{Z}_{15}^*$ e identificó las bases $a \in \{4, 11, 14\}$ con período $r = 2$ como las de menor complejidad estructural de circuito (Sección 5.2, Tabla 5.1). En la serie V1 (`opt` $= 3$, PT activado), la base $a = 4$ ($r = 2$, 116 CZ) alcanzó $PST = 75{,}5\,\%$ frente a $a = 7$ ($r = 4$, 244 CZ) con $PST = 67{,}5\,\%$ —una diferencia de 8.0 puntos porcentuales atribuible directamente a la profundidad del circuito (Tabla 6.4)—, lo que convierte una predicción teórica sobre el escalamiento con r en una ventaja medible y reproducible en hardware NISQ.

La cadena completa de degradación se cuantificó utilizando la fidelidad de Hellinger $\mathcal{F}_H$ y la probabilidad de éxito experimental PST. Para las dos bases con medición tanto en FakeTorino como en hardware real ($a \in \{4, 7\}$), la señal pasó de $PST = 100\,\%$ en simulación ideal a un promedio de $31{,}8\,\%$ en *fake backend* y $57{,}6\,\%$ en hardware real, con una fidelidad promedio en hardware $\overline{\mathcal{F}}_H \approx 0{,}310$ sobre las cuatro bases con $\mathcal{F}_H$ reportada (Tabla 6.9). El modelo de decaimiento exponencial $\mathcal{F} \sim (1-\epsilon_{\text{CZ}})^{N_g}$ con $\epsilon_{\text{CZ}} \approx 5 \times 10^{-3}$ resultó un estimador razonable para circuitos de profundidad moderada en el procesador Heron r1: para $a = 4$ con 116 CZ predice $\mathcal{F}_{\text{est}} \approx 0{,}56$, frente a la fidelidad observada $\mathcal{F}_H = 0{,}529$ en el estudio A2 con `opt` $= 2$.

Quizá el hallazgo más inesperado, y a la vez más informativo desde un punto de vista físico, se obtuvo al evaluar las técnicas de supresión de errores en hardware real. El Desacoplamiento Dinámico con secuencia XY4, lejos de mejorar la señal, la degradó en $62{,}5$ puntos porcentuales: el PST se redujo de $82{,}8\,\%$ (línea base) a $20{,}3\,\%$, la fidelidad cayó de $\mathcal{F}_H = 0{,}547$ a $0{,}154$, y la dispersión pasó de 30 a 163 *outcomes* únicos (Tabla 6.6). La combinación PT+DD acentuó este efecto hasta llevar la señal por debajo del $1\,\%$ ($PST = 0{,}8\,\%$, $\mathcal{F}_H = 0{,}007$, 469 *outcomes* únicos sobre los 512 posibles), produciendo una distribución prácticamente indistinguible del ruido uniforme. Como se discutió en la Sección 6.3.4, este comportamiento

se atribuye a que, para circuitos de profundidad moderada ($D_{2Q} \sim 116$) en procesadores con compuertas nativas de alta fidelidad, los intervalos de inactividad son demasiado breves para que la refocalización compense los errores de calibración propios de los pulsos de DD; adicionalmente, el PT modifica patrones de cancelación coherente potencialmente favorables. La implicación práctica es directa: la aplicación indiscriminada de técnicas de supresión puede ser contraproducente, y su eficacia debe evaluarse empíricamente para cada régimen operativo.

A los resultados anteriores se añade una observación de carácter metodológico: en el régimen de circuitos cortos aquí estudiado y sobre las dos bases con medición disponibles, el *Fake Backend* utilizado subestima la calidad del hardware real. El contraste de la Tabla 6.8 muestra que, para el mismo circuito $a = 4$ con `opt` $= 3$, FakeTorino predice $PST_{\text{FT}} = 3{,}5\,\%$ sin supresión, mientras que `ibm_torino` entrega $PST_{\text{HW}} = 82{,}8\,\%$. Además, las cuatro variantes de supresión producen señales prácticamente indistinguibles en el simulador ($PST \in [2{,}34, 3{,}91]\,\%$, $\mathcal{F}_H \in [0{,}003, 0{,}007]$), pero difieren en aproximadamente un orden de magnitud en hardware. Esta firma del régimen NISQ —ruido coherente, deriva de calibración entre ventanas y posibles cancelaciones destructivas en cadenas largas de CZ— no la captura el modelo estocástico del *snapshot*, lo que tiene implicaciones claras para el uso de simuladores ruidosos como filtro previo a la ejecución en hardware. Esta observación se sostiene sobre dos bases con medición pareada y no debe extrapolarse sin nueva evidencia a circuitos más profundos o a otras configuraciones del modelo de ruido.

7.2 Limitaciones del trabajo

Es importante reconocer las limitaciones que acotan el alcance de los resultados obtenidos.

La limitación más fundamental es la restricción al caso $N = 15$, la instancia pedagógica más pequeña del algoritmo de Shor. Si bien este caso es canónico en la literatura y permite ilustrar de forma rigurosa la cadena completa de factorización, la extrapolación a enteros de relevancia criptográfica resulta inviable en el hardware actual.

En segundo lugar, el número de ejecuciones en hardware real estuvo doblemente condicionado. Primero, por el cupo del IBM Quantum Open Plan, que limita el tiempo mensual de procesador real disponible. Segundo, por la propia disponibilidad de `ibm_torino`: durante el período experimental, la cola del servicio acumulaba decenas de *jobs* en espera y el procesador atravesó al menos una ventana de mantenimiento, lo que obligó a ejecutar el estudio en días no consecutivos. Tras el cierre de la fase experimental, `ibm_torino` fue retirado de la flota disponible de IBM Quantum, lo que impidió de manera definitiva la ejecución de réplicas adicionales o de un barrido factorial completo. La opción de migrar a otros procesadores Heron disponibles fue evaluada y descartada por motivos de cronograma: cada nuevo *backend* habría requerido instanciar su correspondiente *Fake Backend*, reproducir íntegramente la Fase 2 sobre ese *snapshot* y rehacer el protocolo de transpilación específico para

su topología y calibración. En consecuencia, los 13 *jobs* reportados corresponden a corridas únicas por configuración, sin réplicas, y deben interpretarse con la cautela estadística que ello implica: una parte de las diferencias observadas (en particular las del orden de unos pocos puntos porcentuales) es compatible tanto con efectos sistemáticos como con la variabilidad natural del procesador entre ventanas de calibración. La anomalía observada para $a = 11$ ($PST = 35{,}5\,\%$ a pesar de tener la menor profundidad 2Q del estudio, 75 capas, Tabla 6.4) ilustra esta limitación: sin réplicas, no es posible distinguir formalmente entre variabilidad en la calidad de los qubits asignados por SABRE y un efecto sistemático asociado a la base.

En tercer lugar, los simuladores con modelos de ruido (*Fake Backends*) mostraron limitaciones importantes como predictores del comportamiento en hardware real. Como ya se documentó en las Secciones 6.2.2 y 6.3.4, el *snapshot* de FakeTorino reproduce despolarización, relajación térmica T_1/T_2 y matrices de confusión de lectura, pero no captura *crosstalk* dinámico entre qubits vecinos, errores coherentes de compuerta ni la deriva temporal de las calibraciones. El resultado neto, sintetizado en la Tabla 6.8, es que el *snapshot* produce un veredicto cualitativa y cuantitativamente incorrecto sobre la efectividad de DD y PT en esas dos bases; esta observación no debe extrapolarse sin nueva evidencia a circuitos más profundos o a otras configuraciones del modelo de ruido.

Finalmente, la implementación de la exponenciación modular se basó en matrices unitarias precomputadas (clase `UnitaryGate` de Qiskit). Esta opción ubica el presente trabajo dentro de las implementaciones *compiladas* de Shor descritas en la literatura [5–7], no en una implementación de propósito general. La elección fue deliberada y, dado el tamaño del registro objetivo ($t_t = 4$ qubits), permitió minimizar la profundidad del circuito y centrar la atención en las preguntas de transpilación y supresión que son el foco del trabajo. No obstante, no constituye una implementación escalable: una demostración general del algoritmo requeriría descomponer la exponenciación modular en compuertas elementales reversibles, lo que incrementaría sustancialmente la profundidad y, por tanto, agudizaría los efectos de ruido aquí caracterizados.

7.3 Perspectivas futuras

Los resultados obtenidos abren varias líneas de investigación viables para futuros trabajos de grado o proyectos en la Universidad de Nariño.

Una extensión natural es la implementación de la estimación de fase iterativa (*Iterative QPE*), que recicla un único qubit de control mediante mediciones intermedias y realimentación clásica, comprimiendo el registro de conteo de $t = 9$ qubits a un solo qubit [24]. Esta técnica reduciría drásticamente la conectividad requerida y la profundidad efectiva del circuito, y abre la posibilidad concreta de ejecutar Shor para numeros mas grandes en el hardware actual.

Otra dirección prometedora, sugerida directamente por los resultados aquí obtenidos, es el

estudio de Desacoplamiento Dinámico adaptativo. La caída de $PST = 82{,}8\,\%$ a $20{,}3\,\%$ con DD XY4 (Tabla 6.6) muestra que esta secuencia no es adecuada para circuitos de profundidad moderada con compuertas nativas de alta fidelidad. Sería valioso evaluar secuencias más cortas (como el eco de espín simple X-X) o protocolos de DD que se activen únicamente cuando los *idle times* superen un umbral determinado por las tasas de error de compuerta. Un análisis análogo aplica al *Pauli Twirling*: el régimen en el que su aleatorización resulta realmente beneficiosa puede caracterizarse experimentalmente como función de la profundidad del circuito y de la calidad de calibración.

También resultaría valioso extender el estudio a procesadores de generaciones más recientes de la hoja de ruta de IBM (Heron r2, Flamingo) o a otras arquitecturas con conectividad distinta, cuyas mejoras en fidelidad de compuerta y tiempos de coherencia podrían ampliar el umbral de profundidad viable y acercar la posibilidad de factorizar números mas grandes en hardware real.

Finalmente, la metodología reproducible documentada en este trabajo —con el repositorio público de código, los `job_id` de IBM Quantum y los protocolos de transpilación— sienta las bases para que futuros estudiantes de la Universidad de Nariño repliquen, extiendan y mejoren estos experimentos, contribuyendo a la consolidación de competencias en computación cuántica experimental en la institución.

7.4 Reflexión final

Este trabajo de grado se concibió con el propósito de tender un puente entre la formulación teórica del algoritmo de Shor y su realidad práctica en el hardware cuántico contemporáneo. Los resultados aquí reportados confirman que, en la era NISQ, la ejecución exitosa de algoritmos cuánticos profundos no depende únicamente de la calidad del hardware ni de la corrección formal del algoritmo, sino de un entendimiento integral que abarca la ingeniería de compilación, la selección inteligente de parámetros clásicos, la comprensión del modelo de ruido del dispositivo y la evaluación empírica de las técnicas de supresión de errores.

La demostración de que el algoritmo de Shor puede factorizar $N = 15 = 3 \times 5$ con una señal del $82{,}8\,\%$ en un procesador real de IBM, accesible de forma remota desde Pasto, ilustra tanto el potencial como las limitaciones actuales de la computación cuántica. La brecha entre factorizar $N = 15$ y un entero criptográficamente relevante (miles de millones de compuertas lógicas, qubits lógicos tolerantes a fallos) sigue siendo inmensa. Pero cada mejora en la fidelidad de las compuertas, cada reducción del *overhead* de transpilación y cada avance en las técnicas de supresión y mitigación de errores acortan esa distancia.

Más allá del rigor técnico, este proyecto significó el paso del aula a la consola: del diagrama del circuito en el cuaderno al envío real del *job* y a la espera, días después, del histograma medido a unos 50 mK. Aprender a leer ese histograma —a distinguir cuánto del pico es señal

y cuánto es deriva de calibración, a aceptar que un experimento aparentemente contraintuitivo como la degradación con DD también es un resultado válido— fue, en buena medida, parte central del aprendizaje que dejó el trabajo.

Se confía en que este aporte a la Universidad de Nariño y a su comunidad académica constituya un precedente concreto de investigación experimental en computación cuántica: una metodología documentada, un conjunto de datos reproducibles —incluidos todos los `job_id` y archivos de configuración— y la evidencia de que es posible realizar investigación significativa en esta disciplina con los recursos de acceso abierto disponibles. Se espera que este estudio sirva como punto de partida para futuros trabajos que continúen explorando las fronteras de la computación cuántica desde la institución.

Apéndice A

Preliminares matemáticos y formalismo cuántico

Este apéndice complementa la Sección 4.2.5 con los elementos formales mínimos del modelo de ruido empleados para describir, simular y mitigar los errores observados en la ejecución del algoritmo de Shor sobre los procesadores de IBM Quantum.

Nota sobre la notación. En el cuerpo principal del trabajo y a partir de la metodología experimental (Capítulo 5) se utiliza la variable a para denotar la base de la búsqueda de orden, siguiendo la convención habitual de la literatura NISQ. En este apéndice y en aquellos que reproducen textualmente el desarrollo de Nielsen & Chuang [24] se conserva la notación original x para la misma cantidad, con el único propósito de facilitar el cotejo con la referencia.

A.1 Canales cuánticos y representación de Kraus

El formalismo general para describir procesos cuánticos ruidosos es el de los *canales cuánticos*, mapas completamente positivos y que preservan la traza (CPTP) [24, 40]. Todo canal cuántico $\mathcal{E}$ admite una *representación de operadores de Kraus*

$$\mathcal{E}(\rho) = \sum_k E_k \, \rho \, E_k^\dagger, \qquad \sum_k E_k^\dagger E_k = I, \tag{A.1}$$

donde $\{E_k\}$ son los *operadores de Kraus* [24]. La evolución unitaria ideal corresponde al caso de un solo operador $E_0 = U$; los procesos no unitarios (relajación, desfase, despolarización) requieren múltiples operadores de Kraus.

A.2 Canal despolarizante post-compuerta

En un dispositivo NISQ, la aplicación de una compuerta cuántica U no produce la transformación ideal $\rho \mapsto U\rho U^\dagger$, sino una versión ruidosa $\mathcal{E}_{\text{gate}}$ [2]. Componiendo el canal despolarizante

de la Ec. 4.9 con la operación ideal se obtiene el modelo de error post-compuerta empleado en este trabajo:

$$\mathcal{E}_{\text{gate}}(\rho) = \mathcal{E}_{\text{depol}}\big(U\rho U^\dagger\big) = (1-p)\,U\rho U^\dagger + \frac{p}{d}\,I. \tag{A.2}$$

El parámetro p se identifica con la tasa de error de compuerta reportada por IBM para cada qubit y par de qubits en sus calibraciones diarias [3]. En las simulaciones clásicas con el *noise model* de Qiskit Aer, este canal se aplica tras cada compuerta del circuito transpilado.
Las siguientes secciones recogen el desarrollo formal de los resultados algebraicos empleados en §4.3.1, §4.3.3 y §4.3.4, presentados allí en forma compacta.

A.3 Exponenciación modular

El efecto combinado de las operaciones controladas $c\text{-}U_{x,N}^{2^j}$ sobre los dos registros del algoritmo de búsqueda de orden es la transformación [24]

$$\begin{aligned} |z\rangle|y\rangle &\to |z\rangle\, U_{x,N}^{z_t 2^{t-1}} \cdots U_{x,N}^{z_1 2^0}\,|y\rangle \\ &= |z\rangle\,|x^{z_t 2^{t-1}} \times \cdots \times x^{z_1 2^0}\, y \pmod N\rangle \\ &= |z\rangle\,|x^z\, y \pmod N\rangle, \end{aligned} \tag{A.3}$$

donde $z = z_t 2^{t-1} + \cdots + z_0$ es la representación binaria del contenido del primer registro. La secuencia de operaciones controladas-U^{2^j} equivale a multiplicar el contenido del segundo registro por la exponencial modular $x^z \pmod N$.

Estrategia de cuadrado repetido

El cálculo de $x^z \pmod N$ se descompone en dos etapas [24]:

1. *Pre-computación de potencias.* se calculan sucesivamente $x^2 \pmod N$, $x^4 = (x^2)^2 \pmod N$, $x^8 = (x^4)^2 \pmod N$, y así hasta $x^{2^{t-1}} \pmod N$. Esto requiere $t-1 = \mathcal{O}(L)$ operaciones de cuadrado modular, cada una con un costo de $\mathcal{O}(L^2)$ operaciones elementales, para un total de $\mathcal{O}(L^3)$.

2. *Multiplicaciones modulares.* una vez obtenidas las potencias, el exponencial se ensambla mediante la factorización

$$x^z \pmod N = \big(x^{z_t 2^{t-1}} \pmod N\big)\big(x^{z_{t-1} 2^{t-2}} \pmod N\big)\cdots\big(x^{z_1 2^0} \pmod N\big). \tag{A.4}$$

   Esto requiere $t-1$ multiplicaciones modulares adicionales, cada una con costo $\mathcal{O}(L^2)$, para un total de $\mathcal{O}(L^3)$.

El costo total de la exponenciación modular es, por tanto, $\mathcal{O}(L^3)$ compuertas, dominando el costo computacional del algoritmo completo [24].

Requisito de reversibilidad

Un computador cuántico exige que todas las operaciones sean unitarias. El circuito clásico de multiplicación modular $|z\rangle|y\rangle \rightarrow |z\rangle|x^z y \text{ mód } N\rangle$ se implementa de forma reversible mediante la técnica de descomputación: se computa $x^z \pmod N$ en un registro auxiliar, se multiplica reversiblemente el segundo registro por este valor, y finalmente se borra el registro auxiliar invirtiendo la computación [24].

Modos de fallo de la búsqueda de orden

El éxito de la subrutina puede verse comprometido por dos modos de fallo. El primero ocurre cuando el procedimiento de estimación de fase produce una aproximación inexacta de s/r; teóricamente este error ϵ se mitiga aumentando el número de qubits auxiliares, pero en hardware NISQ esta solución es limitada por la acumulación de errores por ruido y decoherencia en circuitos profundos. El segundo fallo, de naturaleza algorítmica, ocurre cuando el entero aleatorio s y el orden verdadero r comparten un factor común, lo que provoca que el algoritmo de fracciones continuas retorne un divisor r' de r en lugar del orden exacto [24]. Para solventar esta limitación se emplean tres estrategias: (1) repetir el procedimiento $2\log(N)$ veces para garantizar con alta probabilidad la observación de un par coprimo; (2) iterar el proceso clásico reemplazando la base x por $x' \equiv x^{r'} \pmod N$ para deducir el orden de forma multiplicativa; o (3) ejecutar el circuito completo dos veces de forma independiente para obtener dos estimaciones r'_1 y r'_2. La tercera es la más eficiente, ya que la probabilidad de que los numeradores s'_1 y s'_2 no compartan factores comunes está acotada inferiormente por $1/4$, y el orden verdadero se extrae de manera determinista calculando $\text{mcm}(r'_1, r'_2)$ [24].

A.4 Extracción del orden mediante fracciones continuas

Tras la medición del registro de conteo se obtiene un entero $y \in \{0, 1, \ldots, 2^t - 1\}$. La fase estimada es $\tilde{\varphi} = y/2^t$, que aproxima al valor real $\varphi = s/r$ para algún $s \in \{0, \ldots, r-1\}$. La extracción del orden r procede como sigue [24]:

1. Se calcula la expansión en fracción continua de $y/2^t$:

$$\frac{y}{2^t} = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots}}}, \tag{A.5}$$

   donde $a_0 = 0$ y los coeficientes parciales a_i se obtienen mediante el algoritmo de Euclides aplicado a y y 2^t.

2. Se calculan los convergentes sucesivos p_k/q_k de la expansión, definidos recursivamente por
$$p_k = a_k p_{k-1} + p_{k-2}, \qquad q_k = a_k q_{k-1} + q_{k-2}, \tag{A.6}$$
con condiciones iniciales $p_{-1} = 1$, $p_{-2} = 0$, $q_{-1} = 0$, $q_{-2} = 1$. El Teorema 1 (Ec. 4.18) garantiza que si $t \geq 2L + 1$, donde $L = \lceil \log_2 N \rceil$, entonces s/r aparece como uno de los convergentes [24].

3. Para cada convergente con denominador $q_k < N$, se verifica computacionalmente si $x^{q_k} \equiv 1 \pmod{N}$. Si la condición se cumple, $r = q_k$ (o r divide a q_k); en caso contrario, se prueban los convergentes subsiguientes o se repite la ejecución cuántica con un nuevo resultado de medición.

A.5 Formalismo del Desacoplamiento Dinámico

Esta sección complementa la presentación del Desacoplamiento Dinámico introducida en el cuerpo principal (Sección 4.6). El propósito es presentar el formalismo subyacente: la descripción del Hamiltoniano de interacción sistema–entorno, la derivación de la cancelación a primer orden mediante pulsos π, el desarrollo explícito de las secuencias X-X y XY4, y los detalles de planificación temporal en la implementación.

Hamiltoniano sistema–baño

Para formalizar la situación, se parte del Hamiltoniano total del sistema (qubit) acoplado a un baño (entorno) [9, 59]:
$$H = H_S \otimes \mathbb{I}_B + \mathbb{I}_S \otimes H_B + H_{SB}, \tag{A.7}$$
donde H_S describe la dinámica propia del qubit, H_B la del entorno y H_{SB} el acoplamiento indeseado entre ambos. Es precisamente el término H_{SB} el responsable de la decoherencia: sin intervención, este término entrelaza irreversiblemente el qubit con los grados de libertad del baño, provocando una pérdida neta de información cuántica. En qubits superconductores tipo *transmon*, H_{SB} engloba en particular la interacción ZZ residual siempre activa con qubits vecinos, el *crosstalk* estático y el acoplamiento con modos del resonador [3, 9]. El Desacoplamiento Dinámico no intenta eliminar físicamente este acoplamiento, sino *promediarlo a cero* mediante la aplicación rápida de pulsos de control sobre el qubit [59].

Eco de espín y secuencia X-X

La idea más sencilla de DD se remonta al célebre *eco de espín* (*spin echo*), descubierto por Hahn en 1950 en el contexto de resonancia magnética nuclear (NMR) [60]. El principio es directo: si durante un intervalo de tiempo τ el qubit acumula una fase errónea $+\phi$ debido al ruido, la aplicación de un pulso π (una rotación de $180°$ alrededor del eje x, equivalente a una compuerta X) invierte el signo de la evolución del ruido. Si a continuación se deja pasar otro intervalo τ, la fase errónea acumulada es $-\phi$, y ambas contribuciones se cancelan. El

resultado neto es que el efecto del ruido se "deshace" a primer orden en τ.

La extensión directa del eco de espín es la secuencia X-X (también conocida como CPMG, una implementación concreta de la idea X-X según [62]), que aplica dos pulsos π alrededor del eje x de forma simétrica durante el intervalo de inactividad [9, 62]:

$$X\text{-}X \equiv \frac{\tau}{2} - X - \tau - X - \frac{\tau}{2}. \tag{A.8}$$

Esta secuencia cancela a primer orden el ruido de defasamiento (σ_z), ya que $X\sigma_z X = -\sigma_z$: la primera mitad del intervalo acumula una fase errónea que la segunda mitad invierte [9]. Sin embargo, al rotar exclusivamente alrededor del eje x, la secuencia X-X no cancela errores que conmutan con el pulso, como ciertos acoplamientos σ_x estáticos, y ofrece solo protección parcial [9]. Pese a esta limitación, su sencillez la convierte en una opción eficaz y de bajo costo para suprimir el defasamiento dominante en qubits superconductores, y fue la secuencia empleada en las simulaciones con modelo de ruido de este trabajo.

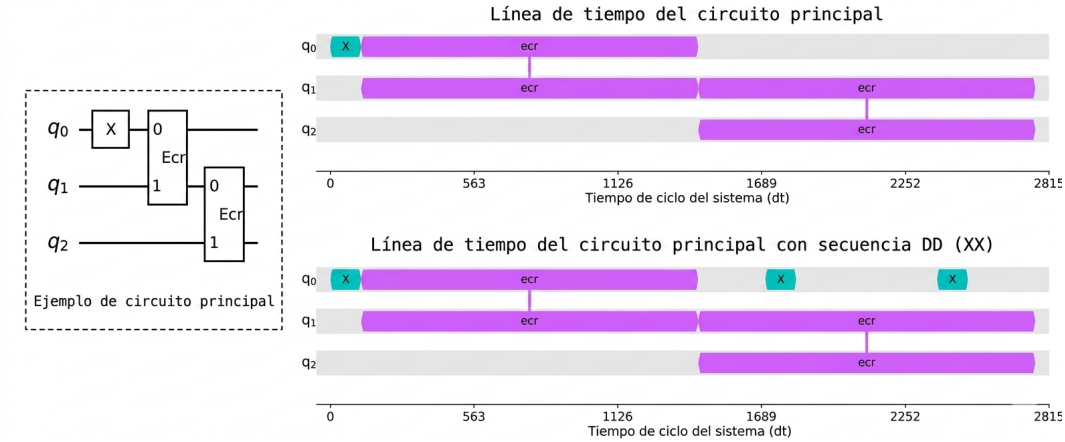

Figura A.1: Ejemplo de DD de una secuencia simple X-X en un circuito sencillo [64].

Cancelación universal a primer orden con XY4

En la práctica, el ruido en un procesador real no se limita a un solo eje. Los qubits superconductores están sujetos a errores tanto de fase (σ_z) como de *bit-flip* (σ_x) e interacciones cruzadas (σ_y). La secuencia XY4 (Ec. 4.21) es *universal* de primer orden: en la idealización de pulsos instantáneos y perfectos, cancela interacciones arbitrarias del qubit con el baño hasta $\mathcal{O}(\tau^2)$ [9, 59]. Formalmente, el Hamiltoniano efectivo de error tras un ciclo completo de XY4 se reduce a [9]:

$$H_{\text{XY4}}^{\text{eff}} = \mathbb{I}_S \otimes \tilde{B} + \mathcal{O}(\tau^2), \tag{A.9}$$

donde $\tilde{B}$ es un operador que actúa exclusivamente sobre el baño. En otras palabras, a primer orden el qubit queda completamente desacoplado del entorno: la evolución residual solo

afecta al baño, sin contaminar la información cuántica del sistema. Este resultado es una consecuencia directa del formalismo de simetrización grupal desarrollado en [59]: en esencia, los pulsos DD aplicados periódicamente generan un promedio del Hamiltoniano de interacción sobre un grupo finito de operadores, de modo que los términos indeseados se anulan en el límite de pulsos rápidos.

A.6 Formalismo del Pauli Twirling

Esta sección complementa la presentación del Pauli Twirling introducida en el cuerpo principal (Sección 4.6). El propósito es presentar el formalismo del *twirling* sobre el grupo de Pauli, demostrar la conversión de un canal de error general en un canal estocástico de Pauli, describir el conjunto finito de pares válidos para las compuertas CNOT y ECR, y detallar la implementación *circuit-by-circuit* en Qiskit Runtime.

Demostración de la conversión a canal de Pauli

Para formalizar la idea, considérese un canal de ruido general $\mathcal{E}$ asociado a una compuerta, cuya acción sobre el estado ρ se escribe como $\mathcal{E}(\rho) = \sum_M M\rho M^\dagger$, donde los operadores M representan los distintos mecanismos de error. El *twirling exacto* sobre un conjunto de operadores de Pauli $W \subseteq \{I, X, Y, Z\}^{\otimes n}$ se define como el promedio [10]:

$$\mathcal{T}_W(\mathcal{E})(\rho) = \frac{1}{|W|} \sum_{w \in W} w\, \mathcal{E}\big(w\rho\, w^\dagger\big)\, w^\dagger. \tag{A.10}$$

En otras palabras, cada vez que se ejecuta el circuito, se conjugan los operadores de error con un par de compuertas de Pauli w distinto, elegido del conjunto W. Al promediar sobre todos los elementos del conjunto (o sobre suficientes muestras aleatorias), las componentes fuera de la diagonal del canal de error en la base de Pauli se cancelan. El resultado es que el canal de ruido original se convierte en un *canal estocástico de Pauli* [10, 63]:

$$\mathcal{T}_W(\mathcal{E})(\rho) = \sum_{g \in G} p_g\, g\, \rho\, g^\dagger, \tag{A.11}$$

donde $G = \{I, X, Y, Z\}^{\otimes n}$ es el grupo de Pauli sobre n qubits y p_g es la probabilidad de que ocurra el error de Pauli g, con $\sum_g p_g = 1$. Este tipo de canal es diagonal en la base de Pauli y se comporta de forma estadísticamente simple: los errores se acumulan linealmente con el número de compuertas, en lugar de cuadráticamente como ocurre con los errores coherentes [63].

Pares de Pauli válidos para CNOT y ECR

Como se indicó en el cuerpo principal (Ec. 4.22), los pares $(P_{\text{antes}}, P_{\text{después}})$ utilizados en el twirling no son arbitrarios: deben preservar la operación lógica de la compuerta original $\mathcal{U}$

salvo por una fase global irrelevante. Para cada compuerta nativa de dos qubits existe, en consecuencia, un conjunto finito de combinaciones válidas, que se enumeran en la Figura A.3 para los casos CNOT y ECR. El motor de ejecución de Qiskit selecciona aleatoriamente entre estas combinaciones cada vez que genera una instancia del circuito.

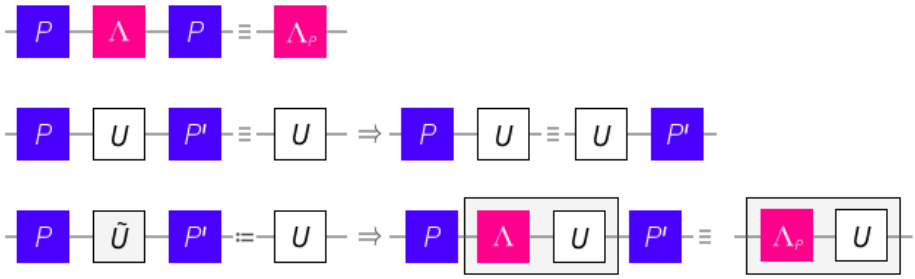

Figura A.2: Principio algebraico del Pauli Twirling. Primera fila: una compuerta de Pauli Λ_P puede absorberse en los operadores envolventes, simplificando el circuito. Segunda fila: una compuerta ideal U envuelta entre pares de Pauli P–P' equivale a U con pares actualizados, preservando la operación lógica. Tercera fila: cuando la compuerta presenta un error coherente $\tilde{U} = U \cdot E$, el canal resultante tras el twirling es equivalente a $\Lambda_P \cdot U$, convirtiendo el error coherente en un factor de Pauli aleatorizado [64].

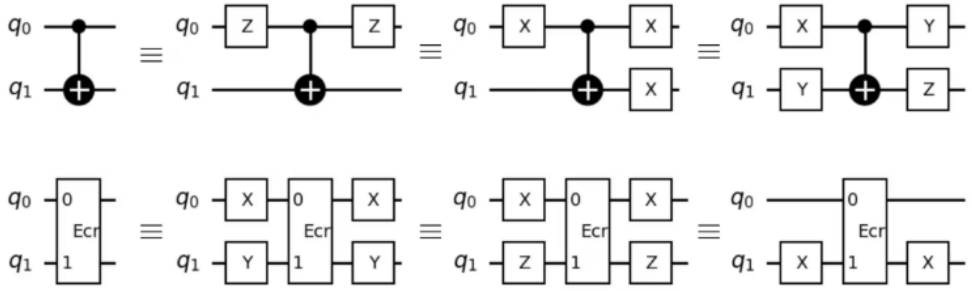

Figura A.3: Pauli Twirling para las compuertas nativas CNOT (filas superiores) y ECR (filas inferiores) en procesadores IBM Quantum. Cada conjunto de circuitos dentro de una fila es lógicamente equivalente, es decir, produce el mismo efecto ideal sobre los qubits, al satisfacer la condición $P_{\text{después}} \cdot U \cdot P_{\text{antes}} = e^{i\varphi} U$. Los pares de Pauli mostrados (X, Y, Z en los qubits de control y objetivo) constituyen el conjunto finito de combinaciones válidas del que Qiskit Runtime selecciona aleatoriamente en cada instancia del circuito [64].

Apéndice B

Arquitectura del software e implementación

Este apéndice documenta los aspectos de implementación del software empleado en el trabajo. Su contenido complementa el Capítulo 5 y permite la reproducibilidad técnica del estudio sin sobrecargar el discurso metodológico del cuerpo principal. Las secciones siguen el orden lógico del flujo de desarrollo: migración técnica respecto al repositorio base, estructura modular del proyecto, configuración detallada del transpilador y del algoritmo SABRE, configuración de las técnicas de supresión, *scripts* de ejecución por fase, módulos de post-procesamiento y, finalmente, versiones del entorno de software.

B.1 Migración técnica del repositorio base a Qiskit 2.x

El repositorio proporcionado por [20] aportó la arquitectura general de clases (`RegisterQC`, `Shor`, `Factorizer`, `Transpiler`) y el flujo de ejecución basado en archivos de configuración. La presente sección detalla las modificaciones técnicas concretas que fueron necesarias para adaptar dicha base al ecosistema actual de IBM Quantum (Qiskit SDK $\geq$ 2.x, primitivas V2 y *Fake Backends* migrados al paquete `qiskit_ibm_runtime.fake_provider`). La Tabla B.1 sintetiza el conjunto de sustituciones aplicadas.

Tabla B.1: Delimitación de aportes y resumen de la migración a Qiskit 2.x: repositorio base frente a contribución propia.

Elemento	**Aporte del repositorio base [20]**	**Contribución de este trabajo**
Arquitectura y SDK	Arquitectura general de clases y flujo de ejecución gobernado por archivos de configuración, basado en versiones previas del SDK de Qiskit.	Migración integral al ecosistema Qiskit 2.x, conservando la arquitectura general pero reescribiendo los módulos incompatibles con las primitivas V2 y los nuevos proveedores.
Acceso a *Fake Backends*	`qiskit.providers.fake_provider` (deprecado).	`qiskit_ibm_runtime.fake_provider`, con instanciación directa `FakeTorino()` correspondiente al procesador Heron r1.
Simulador con ruido	Composición manual del modelo de ruido a partir del *provider* antiguo.	`AerSimulator.from_backend(FakeTorino())`, que carga automáticamente el *snapshot* de calibración.
Parámetro `approximation_degree`	Uso ambiguo: aplicado sin distinguir entre su acepción de truncamiento entero (AQFT lógica) y su acepción de umbral flotante en la síntesis del transpilador.	Identificación y corrección de la ambigüedad: `approximation_degree`$=0{,}7$ se fija exclusivamente en la etapa de transpilación (flotante en $[0, 1]$); la `QFT` lógica se instancia sin truncamiento adicional.
Técnicas de supresión (DD y *Pauli Twirling*)	Configuración por defecto: secuencia DD heredada sin selección explícita y sin activación de *Pauli Twirling*.	Configuración manual mediante `SamplerOptions`, conforme a las exigencias propias de `SamplerV2`.
Conversión *bitstring*-a-entero	Conversión implícita y dependencia del *Endianness* sin documentación ni verificación.	Conversión explícita con la convención *little-endian* de Qiskit, verificada empíricamente en la Fase 1 contra los picos teóricos $y_s = \lfloor s\, 2^t/r \rceil$ obtenidos por simulación ideal.
Protocolo experimental	Ausencia de un protocolo estructurado por fases.	Diseño e implementación de un protocolo en tres fases (simulación ideal, simulación con ruido y hardware real), con *scripts* autocontenidos y configuración independiente para cada una.
Post-procesamiento por fracciones continuas	Utilidades dispersas en el repositorio y post-procesamiento elemental.	Módulo propio diferenciado por fase (`fracciones_continuas_*.py`) con interfaz común, verificación del orden r, extracción de factores no triviales e integración con la verificación del *Endianness*.

B.2 Estructura modular del proyecto

La arquitectura del proyecto se estructura en cinco capas funcionales que reflejan la separación lógica entre las responsabilidades del algoritmo, los servicios de ejecución y las utilidades de configuración. Todo el código se encuentra en el repositorio público del trabajo [46]; la organización por carpetas se ilustra en el árbol siguiente.

```
quantum-shors-algorithm/
+-- main.py                # Punto de entrada
+-- common/
|   +-- settings.py/settings.yaml    # Configuracion global
|   \-- backend.py                   # Factory de backends
+-- algorithm/
|   +-- circuit.py            # Clase RegisterQC
|   \-- phase_estimation.py   # Clase Shor (orquestador)
+-- services/
|   +-- arithmetics.py       # Factorizer(envoltura clasica)
|   +-- sampling.py          # AerSampler / IBMRuntimeSampler
|   \-- transpilation.py     # Transpiler(PassManager+SABRE)
+-- utilities/
|   +-- optimal_a.py           # Seleccion de bases validas
|   +-- job_results.py       # Recuperacion asincrona de jobs
|   \-- arguments_parser.py    # Validacion de argumentos
+-- simulacion_ideal/          # Fase 1: scripts y resultados
+-- simulacion_con_ruido/      # Fase 2: scripts y resultados
\-- ejecucion_hardware/        # Fase 3: scripts y resultados
```

Capa de entrada y configuración (`main.py`, `common/settings.py`, `utilities/arguments`). El punto de entrada del programa carga la configuración centralizada desde un archivo YAML y procesa los argumentos de la línea de comandos o, alternativamente, los parámetros de un archivo `.ini` de configuración masiva. Este esquema sigue la filosofía de los *Qiskit Patterns*, que promueven un flujo cíclico: mapear el problema, optimizar el circuito, ejecutar y post-procesar.

Capa algorítmica (`algorithm/circuit.py`, `algorithm/phase_estimation.py`). Contiene la lógica de construcción del circuito cuántico (clase `RegisterQC`) y la orquestación de la estimación de fase cuántica (clase `Shor`). La clase `Shor` coordina todo el flujo experimental: configura el *backend*, invoca al transpilador, activa las opciones de supresión de errores en la primitiva `SamplerV2`, envía el trabajo (*job*) y, en el caso de simulaciones locales, procesa los resultados sincrónicamente.

Capa de servicios (`arithmetics.py`, `sampling.py`, `transpilation.py`). Implementa las operaciones intermedias. La clase `Factorizer` encapsula la envoltura clásica del algoritmo

de Shor (verificaciones de paridad, potencias primas y cálculo del gcd), delegando la subrutina cuántica a la clase `Shor`. La clase `Transpiler` envuelve el `PassManager` de Qiskit y gestiona la inyección de parámetros SABRE personalizados. Las clases `AerSampler` e `IBMRuntimeSampler` abstraen la ejecución en simulador local y en hardware real, respectivamente, unificando la interfaz de muestreo.

Capa de backends (`common/backend.py`). Implementa el patrón de diseño *Factory* para instanciar tres tipos de *backend* de manera intercambiable, alineados con las recomendaciones de Qiskit 2.x: `AerSimulator` sin modelo de ruido (simulación ideal); `AerSimulator.from_backend(FakeTorino())` para simulación con el modelo de ruido calibrado del procesador Heron r1 (que reemplaza al antiguo `FakeProvider`); y `QiskitRuntimeService` + `SamplerV2` en modo `ibm_quantum` para ejecución en hardware real.

Capa de utilidades. Proporciona herramientas de pre-procesamiento clásico (selección de bases óptimas a a través de `get_optimal_a_for_valid_periods`), post-procesamiento (fracciones continuas, extracción de factores, graficación de resultados) y validación de argumentos.

Conexión con IBM Quantum. Se establece mediante el servicio `QiskitRuntimeService`, que utiliza las credenciales almacenadas localmente bajo un nombre de cuenta configurable (por ejemplo, `ibm_quantum`). La ejecución se realiza a través de la primitiva `SamplerV2`. A diferencia de `EstimatorV2`, `SamplerV2` no expone un parámetro `resilience_level`: todas las técnicas de supresión deben configurarse explícitamente a través de `SamplerOptions` (cf. Sección B.4).

B.3 Configuración detallada del transpilador y SABRE avanzado

La Tabla B.2 consolida los parámetros del transpilador empleados en este trabajo, con los valores nominales que se aplican en todas las fases salvo indicación explícita. Estos parámetros se inyectan directamente al método `transpile` de Qiskit a través de la clase envolvente `Transpiler`.

Tabla B.2: Parámetros del transpilador empleados en el trabajo.

Parámetro	**Valor**	**Función**
`optimization_level`	$\{0, 1, 2, 3\}$ (barrido)	Agresividad de las transformaciones aplicadas al circuito.
`basis_gates`	`cz`, `rzz`, `id`, `rx`, `rz`, `sx`, `x`	Conjunto nativo de Heron r1; `cz` es la única compuerta de dos qubits.
`layout_method`	`sabre`	Asignación inicial de qubits lógicos a qubits físicos.
`routing_method`	`sabre`	Inserción de SWAPs durante el enrutamiento.
`approximation_degree`	0,7	Umbral de fidelidad aceptado durante la síntesis de bloques de dos qubits.
`seed_transpiler`	457	Semilla fija para la reproducibilidad de las heurísticas de SABRE.
`scheduling_method`	`alap` (solo Fase 3)	Planifica las instrucciones *as late as possible* para concentrar los `Delay` al inicio del circuito.

Adicionalmente, la clase `Transpiler` permite inyectar parámetros avanzados del algoritmo SABRE directamente en el *pipeline*. El método `_set_transpile_optimization_params` define tres hiperparámetros que controlan la exploración heurística del espacio de soluciones, recogidos en la Tabla B.3.

Tabla B.3: Hiperparámetros avanzados de SABRE.

Hiperparámetro	**Valor**	**Función**
`max_iterations`	10	Número de iteraciones bidireccionales del algoritmo SABRE. Cada iteración recorre el circuito en ambas direcciones, refinando el mapeo lógico–físico.
`layout_trials`	200	Número de semillas aleatorias para la búsqueda del *layout* inicial. A mayor número de ensayos, mayor probabilidad de encontrar un mapeo con menor distancia topológica promedio.
`swap_trials`	200	Número de candidatos SWAP evaluados en cada paso de enrutamiento.

Cuando esta optimización está habilitada, `Transpiler` construye un objeto `SabreLayout` con el mapa de acoplamiento (`CouplingMap`) extraído del *backend* y lo inyecta directamente en la etapa de *layout* del `PassManager`, reemplazando la implementación por defecto. Este procedimiento se implementa en el método `get_sabre_layout`, que maneja explícitamente

los casos en que el *backend* no expone un mapa de acoplamiento (como en el simulador ideal).

Tras la transpilación, el método `get_isa_statistics` extrae las métricas cuantitativas descritas en el cuerpo principal (conteo por tipo de compuerta, número total y profundidad de dos qubits D_{2Q}). Como salvaguarda de seguridad en el envío a hardware, si el circuito transpilado excede las 10^5 compuertas de dos qubits se excluye automáticamente del envío para no desperdiciar recursos en una ejecución cuya señal sería inevitablemente colapsada por el ruido acumulado.

B.4 Configuración detallada de las técnicas de supresión

Las técnicas de supresión de errores se configuran a través de los objetos `DynamicalDecouplingOptions` y `TwirlingOptions`, que se ensamblan en un único `SamplerOptions` y se pasan a la primitiva `SamplerV2` en el momento de la ejecución. Las Tablas B.4 y B.5 consolidan los parámetros adoptados.

Tabla B.4: Parámetros de `DynamicalDecouplingOptions` empleados.

Parámetro	**Valor**	**Función**
`enable`	`True` / `False`	Activa o desactiva la inserción automática de secuencias DD.
`sequence_type`	`XY4`	Secuencia de pulsos aplicada en los intervalos ociosos. `XY4` cancela ruido de fase y de bit a primer orden, frente a `XpXm` heredado del repositorio base.
`scheduling_method`	`alap`	Planifica las instrucciones lo más tarde posible, concentrando los `Delay` al inicio del circuito y maximizando el espacio donde insertar pulsos DD.

Tabla B.5: Parámetros de `TwirlingOptions` empleados.

Parámetro	**Valor**	**Función**
`enable_gates`	`True` / `False`	Inserta pares de operadores de Pauli aleatorios alrededor de cada compuerta de dos qubits.
`enable_measure`	`False` (por defecto)	Aplica *twirling* también a las operaciones de medición; opcional.
`strategy`	`active`	El *twirling* se aplica únicamente a los qubits involucrados en cada compuerta.
`num_randomizations`	`auto`	El número de instancias aleatorias lo determina automáticamente el *runtime* en función del número total de *shots*.

Cuando ambas opciones están habilitadas simultáneamente (configuración DD+PT), el *runtime* de IBM Quantum aplica primero la transpilación con la planificación `alap` y la inserción de secuencias `XY4` en los intervalos ociosos; sobre el circuito así programado, genera múltiples copias con distintas selecciones aleatorias de pares de Pauli para las compuertas de dos qubits, y agrega los conteos de todas las copias en la distribución final reportada al usuario.

B.5 Scripts de ejecución por fase

Los *scripts* que ejecutan de manera autocontenida las tres fases del estudio constituyen el núcleo experimental de este trabajo y no pertenecen al repositorio original. Cada *script* carga su propio archivo `.ini` de configuración y produce reportes automáticos en formato JSON y Markdown junto con las figuras correspondientes. La Tabla B.6 resume los *scripts* principales por fase.

Tabla B.6: *Scripts* de ejecución por fase del protocolo experimental.

Fase	**Script**	**Función**
Fase 1	`ejecutar_shor_ideal_puro.py`	Simulación ideal sobre `AerSimulator` sin modelo de ruido.
Fase 1	`ejecutar_shor_faketorino_ideal.py`	Variante complementaria sobre `AerSimulator.from_backend(FakeTorino())` con el modelo de ruido desactivado (`set_options(noise_model=None)`); aísla el *overhead* topológico.
Fase 2	`ejecutar_shor_faketorino_ruido.py`	Línea base ruidosa sin mitigación. Barre `opt` $\in \{0, 3\}$ para $a \in \{4, 7\}$.
Fase 2	`calcular_fidelidad_ruido.py`	Cálculo detallado de $\mathcal{F}_H$, D_H y D_{KL} frente a la distribución teórica.
Fase 2	`comparar_mitigacion.py`	Diseño factorial 2×2 de las cuatro variantes (sin supresión, DD, PT, DD+PT).
Fase 2	`fracciones_continuas_ruido.py`	Post-procesamiento detallado bajo DD+PT con clasificación señal/ruido por medición.
Fase 3	`ejecutar_estudio_por_lotes_hardware.py`	Envío y registro masivo de *jobs* sobre `ibm_torino`. Agrupa los estudios A2, A3 y A6/A7.
Fase 3	`ejecutar_analisis_transpilacion_hardware.py`	Análisis comparativo de la transpilación al *backend* real previo al envío.
Fase 3	`analizar_caso_a4_hardware.py`	Estudio detallado del caso $a = 4$ (mayor número de configuraciones por estudio A2 y A6/A7).

La Tabla B.7 desglosa los 13 *jobs* ejecutados en `ibm_torino`, organizados en los tres estudios complementarios mencionados en el cuerpo principal. Cada *job* se identificó con su corres-

pondiente `job_id` para permitir la recuperación asíncrona de los resultados días después del envío.

Tabla B.7: Organización de los 13 *jobs* ejecutados en `ibm_torino`.

Estudio	**Configuración**	**Propósito**
V1 (inicial)	PT activado, `opt`= 3, $a \in \{4, 7, 14\}$	Calibración inicial del flujo de envío y verificación cualitativa de la señal.
A2 (`opt` barrido)	$a = 4$ fijo, `opt` $\in \{0, 1, 2\}$ con PT activado	Cuantificar el impacto del nivel de optimización del transpilador en la señal en hardware.
A3 (base a)	`opt`= 3, PT activado, $a \in \{2, 4, 7, 8, 11, 13, 14\}$	Caracterizar la respuesta del procesador en función de la base. Las bases $a = 11$, $a = 14$ se incluyen con propósito diagnóstico (cf. Sección 5.4).
A6/A7 (supresión)	$a = 4$, `opt`= 3; sin supresión, solo DD (XY4), DD+PT	Evaluar el impacto individual y combinado de las técnicas de supresión sobre $\mathcal{F}_H$ y PST.

Los identificadores `job_id` concretos asignados por el servicio de IBM Quantum y las métricas obtenidas para cada *job* se reportan en la Tabla 6.3 del Capítulo 6, donde se discute el análisis de los resultados experimentales.

B.6 Módulos de post-procesamiento clásico

El post-procesamiento se implementa en tres módulos paralelos, uno por cada fase experimental: `fracciones_continuas_simulacion_ideal.py`, `fracciones_continuas_simulacion_ruido.py` y `fracciones_continuas_hardware.py`. Los tres comparten la misma estructura algorítmica documentada en la Sección 5.7 del cuerpo principal y se diferencian únicamente en los detalles de carga de datos: el módulo de simulación lee los conteos del objeto devuelto por `AerSampler`, mientras que el módulo de hardware recupera los *jobs* a partir del `job_id` registrado y accede a la distribución de conteos mediante `job.result()[0].data.output.get_counts()`.

La interfaz común de los tres módulos consiste en una función `procesar_distribucion-(counts, a, N, t)` que devuelve un diccionario estructurado con los siguientes campos: la lista ordenada de los k *bitstrings* más frecuentes, la fase $\tilde{\varphi}$ asociada a cada uno, los convergentes p_k/q_k devueltos por la rutina de fracciones continuas, los candidatos a r que satisfacen la verificación clásica, los factores no triviales obtenidos mediante $\gcd(a^{q_k/2} \pm 1, N)$ y, finalmente, las métricas $\mathcal{F}_H$ y PST calculadas conforme a la Sección 5.7.

La aplicación del algoritmo de fracciones continuas se realiza con `Fraction(phase).limit_denominator(N)` de la biblioteca estándar de Python, que devuelve los convergentes con denominador menor o igual a N. La conversión *bitstring*-a-entero se aplica antes del paso de fracciones continuas y se efectúa de acuerdo con la convención *little-endian* de Qiskit, verificada empíricamente en la Fase 1 contra los picos teóricos $y_s = \lfloor s \cdot 2^t/r \rceil$.

B.7 Configuración del entorno y versiones de software

La Tabla B.8 resume las versiones específicas de cada paquete utilizado en los experimentos. La fijación explícita de estas versiones es necesaria para la reproducibilidad: el ecosistema de Qiskit ha experimentado, en los últimos dos años, sustituciones de API significativas entre versiones consecutivas, especialmente en la transición a las primitivas V2 y a la nueva ubicación de los *Fake Backends*.

Tabla B.8: Entorno de software utilizado en los experimentos.

Paquete / Herramienta	**Versión**
Python	3.11
`qiskit`	2.3.0
`qiskit-aer`	0.16.0
`qiskit-ibm-runtime`	0.34.0
`qiskit-ibm-transpiler`	0.11.1
`numpy`	2.2.2
`scipy`	1.15.1
`matplotlib`	3.10.0
`pandas`	2.2.3
`seaborn`	0.13.2

Apéndice C

Tablas complementarias

Este apéndice consolida las tablas y figuras de validación numérica del capítulo 6. Su contenido respalda las afirmaciones de los capítulos anteriores.

C.1 Verificación de la Transformada Cuántica de Fourier

La QFT sobre $n = 9$ qubits de control se implementó conforme a la descomposición estándar en compuertas Hadamard y rotaciones de fase controladas CR_k (Ec. 4.5), requiriendo teóricamente $n = 9$ compuertas Hadamard, $\binom{9}{2} = 36$ rotaciones CR_k, y $\lfloor 9/2 \rfloor = 4$ compuertas SWAP para la inversión del orden de los qubits de salida. La corrección del operador se verificó numéricamente calculando la norma de Frobenius del error residual $\|U^\dagger U - I\|_F$, tanto para la QFT directa como para su inversa. Adicionalmente, se verificó la relación de inversión $\text{QFT} \cdot \text{QFT}^{-1} = I$ obteniendo $\|\text{QFT} \cdot \text{QFT}^{-1} - I\|_F = 3{,}82 \times 10^{-14}$, lo cual confirma que ambos operadores son mutuamente inversos con precisión numérica de máquina. La discrepancia entre la profundidad descompuesta (Depth Total $= 18$) y la transpilada (Depth Total $= 61$) refleja el costo de traducir las rotaciones CR_k abstractas a la base nativa del procesador objetivo (en Heron r1: $\{\text{RZ}, \text{SX}, \text{X}, \text{CZ}\}$), un factor que se amplifica significativamente al considerar la topología restringida del hardware.

Tabla C.1: Métricas de profundidad y verificación unitaria de la QFT para $n = 9$ qubits. La columna $\|U^\dagger U - I\|_F$ cuantifica la desviación de la unitariedad; valores del orden de 10^{-14} confirman precisión numérica de punto flotante.

Variante	**Depth Total**	**Depth 2Q**	**Gates Totales**	$\|\boldsymbol{U^\dagger U - I}\|_{\boldsymbol{F}}$
QFT directa (decompose)	18	16	49	$3{,}84 \times 10^{-14}$
QFT^{-1} (decompose)	18	16	49	$3{,}79 \times 10^{-14}$
QFT directa (transpilada)	61	30	173	$3{,}84 \times 10^{-14}$
QFT^{-1} (transpilada)	61	30	189	$3{,}79 \times 10^{-14}$

C.2 Bloques de exponenciación modular controlada

El operador unitario de multiplicación modular $U_a|y\rangle = |ay \text{ mód } N\rangle$ se construyó como una matriz de permutación de dimensión $2^L \times 2^L = 16 \times 16$ para cada base a. Como se demostró en la Sección 5.3, este operador satisface la propiedad de periodicidad $(U_a)^r = I$, donde $r = \text{ord}_N(a)$ es el orden multiplicativo. Esta propiedad se verificó numéricamente para todas las bases, obteniendo $\|(U_a)^r - I\|_F = 0{,}00$ en todos los casos. La Tabla C.2 resume la complejidad de los bloques individuales $C\text{-}U_{a^{2^k} \text{ mód } N}$ para cada base.

Tabla C.2: Profundidad de dos qubits (Depth 2Q) y conteo de compuertas de dos qubits (Gates 2Q) para el circuito completo de exponenciación modular controlada, por base. La profundidad total es la suma de los bloques no triviales $C\text{-}U_{a^{2^k}}$ con $a^{2^k} \text{ mód } N \neq 1$.

Base a	$r = \textbf{ord}_{15}(a)$	**Bloques no triviales**	**Depth 2Q total**	**Gates 2Q total**
4	2	$k = 0$ ($b = 4$)	233	236
7	4	$k = 0$ ($b = 7$), $k = 1$ ($b = 4$)	466	472
11	2	$k = 0$ ($b = 11$)	230	233
14	2	$k = 0$ ($b = 14$)	231	234

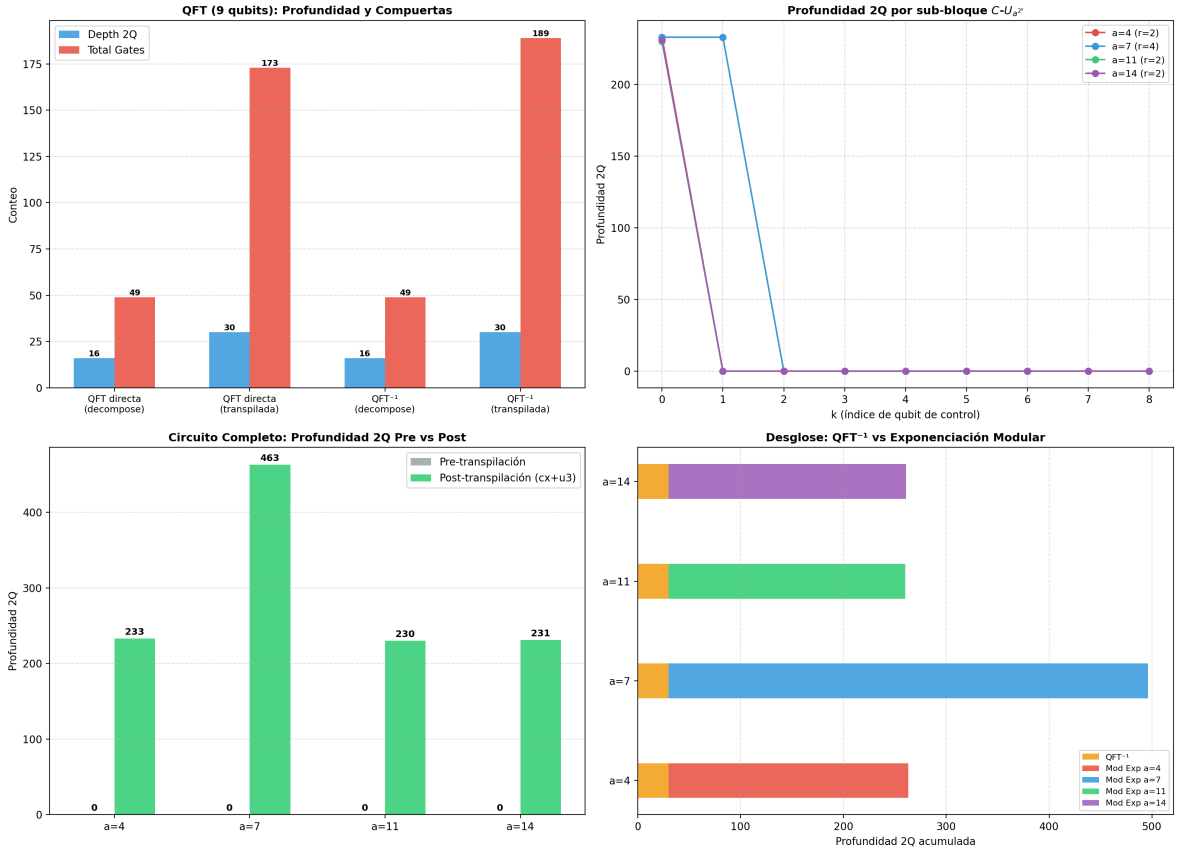

Figura C.1: Caracterización de los bloques fundamentales del algoritmo de Shor para $N = 15$. Se muestra la profundidad y el conteo de compuertas para la QFT ($n = 9$ qubits) y la exponenciación modular controlada $C\text{-}U_{a^{2^k}}$ para cada base $a \in \{4, 7, 11, 14\}$. La verificación numérica confirma la unitariedad de todos los operadores ($\|U^\dagger U - I\|_F < 10^{-10}$) y la periodicidad $(U_a)^r = I$, validando la corrección de la implementación antes de la ejecución del circuito completo.

C.3 Tablas de resultados en simulación ideal

Tabla C.3: Métricas del circuito completo `RegisterQC` (13 Qubits) para $N = 15$ antes (pre-transpilación) y después (post-transpilación a la base $CX + U3$, sin restricción topológica). La *Depth Total (pre)* corresponde a las 12 capas secuenciales del circuito de alto nivel: una compuerta X de inicialización, nueve pares $H - C - U_{a^{2k}}$ aplicados en secuencia sobre cada qubit de control, una QFT^{-1} y una medición final. La post-transpilación revela el costo real en compuertas de dos qubits. Entiéndase (D.) como Depth (profundidad).

Base	**D.Total (pre)**	**D.2Q (pre)**	**D.Total (post)**	**D.2Q (post)**	**CX (post)**
4	12	0	462	233	308
7	12	0	919	463	541
11	12	0	461	230	305
14	12	0	463	231	306

Tabla C.4: Métricas del circuito `RegisterQC` ejecutado en simulación ideal pura (*all-to-all*) para $N = 15$, $t = 9$ y 4096 shots (transpilado con `AerSimulator`), evaluadas bajo los cuatro niveles de optimización del transpilador de Qiskit. Se reportan la profundidad total (D), la profundidad restringida a compuertas de dos qubits (D_{2Q}), el conteo de compuertas de dos qubits (G_{2Q}), el conteo total de compuertas (G_{tot}) y la probabilidad de señal (PST). El estado Δ indica que, pese a la correcta identificación del orden r, los factores clásicos son triviales.

a	**Opt**	**Depth**	**Depth**$_{2Q}$	**Gates**$_{2Q}$	**Gates**$_{\text{tot}}$	**PST (%)**	**Estado**
4	0	463	233	236	734	100.0	✓
4	1	462	233	236	702	100.0	✓
4	2	462	233	236	687	100.0	✓
4	3	462	233	236	687	100.0	✓
7	0	927	466	472	1403	100.0	✓
7	1	925	466	472	1336	100.0	✓
7	2	914	461	466	1296	100.0	✓
7	3	914	461	466	1296	100.0	✓
11	0	464	232	235	732	100.0	✓
11	1	463	232	235	700	100.0	✓
11	2	461	230	233	687	100.0	✓
11	3	461	230	233	687	100.0	✓
14	0	466	233	236	734	100.0	Δ
14	1	465	233	236	705	100.0	Δ
14	2	463	231	234	687	100.0	Δ
14	3	463	231	234	687	100.0	Δ

Tabla C.5: Métricas del circuito `RegisterQC` transpilado a la topología Heavy-Hex de `FakeTorino` (Heron r1) y ejecutado en simulación ideal (sin modelo de decoherencia ni errores de compuerta) para $N = 15$, $t = 9$ y 4096 shots. Se reportan la profundidad y el conteo de compuertas de dos qubits (D_{2Q}, G_{2Q}), la probabilidad de señal (PST), la probabilidad de ruido (masa fuera de los picos teóricos del QPE) y los factores extraídos. El estado Δ marca los casos con factores triviales por $a \equiv -1 \pmod{N}$.

a	**Opt**	**Depth**$_{2Q}$	**Gates**$_{2Q}$	**PST (%)**	**Ruido (%)**	**Factores**	**Estado**
4	0	570	923	100.0	0.0	3, 5	✓
4	1	526	779	100.0	0.0	3, 5	✓
4	2	433	584	100.0	0.0	3, 5	✓
4	3	439	600	100.0	0.0	3, 5	✓
7	0	1180	1648	100.0	0.0	3, 5	✓
7	1	1042	1261	100.0	0.0	3, 5	✓
7	2	874	1075	100.0	0.0	3, 5	✓
7	3	869	1071	100.0	0.0	3, 5	✓
11	0	569	922	100.0	0.0	3, 5	✓
11	1	525	778	100.0	0.0	3, 5	✓
11	2	418	569	100.0	0.0	3, 5	✓
11	3	461	607	100.0	0.0	3, 5	✓
14	0	570	923	100.0	0.0	Triviales	Δ
14	1	526	779	100.0	0.0	Triviales	Δ
14	2	427	588	100.0	0.0	Triviales	Δ
14	3	427	588	100.0	0.0	Triviales	Δ

Tabla C.6: Comparativa del *overhead* de ruteo Heavy-Hex: profundidad y conteo de compuertas de dos qubits entre la simulación pura (*all-to-all*) y la transpilación a `FakeTorino` bajo los cuatro niveles de optimización del transpilador de Qiskit. Los ratios $D_{2Q}^{\text{Torino}}/D_{2Q}^{\text{Puro}}$ y $G_{2Q}^{\text{Torino}}/G_{2Q}^{\text{Puro}}$ cuantifican el factor de amplificación introducido por la topología.

a	**Opt**	**D.Puro**$_{2Q}$	**D.Torino**$_{2Q}$	**Ratio** D	**G.Puro**$_{2Q}$	**G.Torino**$_{2Q}$	**Ratio** G
4	0	233	570	2.45×	236	923	3.91×
4	1	233	526	2.26×	236	779	3.30×
4	2	233	433	1.86×	236	584	2.47×
4	3	233	439	1.88×	236	600	2.54×
7	0	466	1180	2.53×	472	1648	3.49×
7	1	466	1042	2.24×	472	1261	2.67×
7	2	461	874	1.90×	466	1075	2.31×
7	3	461	869	1.89×	466	1071	2.30×
11	0	232	569	2.45×	235	922	3.92×
11	1	232	525	2.26×	235	778	3.31×
11	2	230	418	1.82×	233	569	2.44×
11	3	230	461	2.00×	233	607	2.61×
14	0	233	570	2.45×	236	923	3.91×
14	1	233	526	2.26×	236	779	3.30×
14	2	231	427	1.85×	234	588	2.51×
14	3	231	427	1.85×	234	588	2.51×

Tabla C.7: Métricas de transpilación para el circuito `RegisterQC` ($N = 15$) bajo los mapeos trivial y SABRE, transpilado sobre la topología Heavy-Hex de `FakeTorino` con 4096 *shots* y semilla 457. El conteo de SWAPs explícitas reportado por Qiskit es nulo en todas las configuraciones debido a que el pase de traducción al conjunto de compuertas nativas (`BasisTranslator`) descompone cada SWAP en tres compuertas de dos qubits, que quedan contabilizadas en Gates_{2Q}.

a	Layout	Opt	D. total	Depth_{2Q}	Gates_{2Q}	PST (%)	$\mathcal{F}_H$
4	trivial	0	2615	570	923	100.0	1.0000
4	trivial	1	1841	533	833	100.0	0.9998
4	trivial	2	1839	521	793	100.0	1.0000
4	trivial	3	1843	521	793	100.0	0.9999
4	SABRE	0	2509	504	749	100.0	0.9999
4	SABRE	1	1869	526	779	100.0	1.0000
4	SABRE	2	1584	433	584	100.0	1.0000
4	SABRE	3	1598	439	600	100.0	1.0000
7	trivial	0	5521	1180	1648	100.0	0.9999
7	trivial	1	3758	1094	1432	100.0	0.9999
7	trivial	2	3599	1019	1349	100.0	0.9999
7	trivial	3	3608	1019	1349	100.0	0.9999
7	SABRE	0	4952	1015	1255	100.0	0.9999
7	SABRE	1	3576	1042	1261	100.0	1.0000
7	SABRE	2	3186	874	1075	100.0	1.0000
7	SABRE	3	3172	869	1071	100.0	0.9996
11	trivial	0	2607	569	922	100.0	1.0000
11	trivial	1	1845	532	832	100.0	0.9999
11	trivial	2	1785	508	778	100.0	0.9999
11	trivial	3	1787	508	778	100.0	1.0000
11	SABRE	0	2499	503	748	100.0	1.0000
11	SABRE	1	1864	525	778	100.0	1.0000
11	SABRE	2	1543	418	569	100.0	0.9999
11	SABRE	3	1636	461	607	100.0	1.0000
14	trivial	0	2618	570	923	100.0	0.9999
14	trivial	1	1861	533	833	100.0	0.9999
14	trivial	2	1802	512	785	100.0	1.0000
14	trivial	3	1803	512	785	100.0	0.9998
14	SABRE	0	2509	504	749	100.0	1.0000
14	SABRE	1	1869	526	779	100.0	1.0000
14	SABRE	2	1567	427	588	100.0	1.0000
14	SABRE	3	1567	427	588	100.0	1.0000

Tabla C.8: Reducción porcentual lograda por SABRE respecto al mapeo trivial en profundidad de dos qub125its (D_{2Q}) y conteo de compuertas de dos qubits (G_{2Q}), para cada base a y nivel de optimización. Se observa una dependencia no monótona con el nivel de optimización: un mínimo local en Opt=1 y máximos en Opt=2.

a	Opt	D_{2Q} Triv.	D_{2Q} SABRE	Red. D (%)	G_{2Q} Triv.	G_{2Q} SABRE	Red. G (%)
4	0	570	504	11.6	923	749	18.9
4	1	533	526	1.3	833	779	6.5
4	2	521	433	16.9	793	584	26.4
4	3	521	439	15.7	793	600	24.3
7	0	1180	1015	14.0	1648	1255	23.8
7	1	1094	1042	4.8	1432	1261	11.9
7	2	1019	874	14.2	1349	1075	20.3
7	3	1019	869	14.7	1349	1071	20.6
11	0	569	503	11.6	922	748	18.9
11	1	532	525	1.3	832	778	6.5
11	2	508	418	17.7	778	569	26.9
11	3	508	461	9.3	778	607	22.0
14	0	570	504	11.6	923	749	18.9
14	1	533	526	1.3	833	779	6.5
14	2	512	427	16.6	785	588	25.1
14	3	512	427	16.6	785	588	25.1

C.4 Tablas de resultados en simulación con Ruido

Tabla C.9: Métricas de la simulación con ruido (FakeTorino, sin supresión) para $N = 15$. Se reportan la profundidad de compuertas de dos qubits (D2Q), el conteo de compuertas de dos qubits (G2Q), la probabilidad de señal teórica (PST), la fidelidad de Hellinger ($\mathcal{F}_H$) y los factores extraídos.

Base a	opt	D2Q	G2Q	PST (%)	$\mathcal{F}_H$	Factores	Estado
4	0	504	749	25.00	0.1482	$\{3, 5\}$	✓
4	3	439	600	4.30	0.0010	$\{3, 5\}$	✓
7	0	1015	1255	43.16	0.2285	$\{3, 5\}$	✓
7	3	869	1071	59.18	0.4133	$\{3, 5\}$	✓

Tabla C.10: Análisis detallado de fidelidad con ruido (`opt` $= 3$, FakeTorino). Se reportan la fidelidad de Hellinger ruidosa e ideal, la degradación $\Delta\mathcal{F}_H = \mathcal{F}_H^{\text{ideal}} - \mathcal{F}_H^{\text{ruido}}$.

a	r	$\mathcal{F}_H^{\text{ruido}}$	$\mathcal{F}_H^{\text{ideal}}$	$\Delta\mathcal{F}_H$	Prob. picos
4	2	0.0010	1.0000	0.9990	0,20 %
7	4	0.4559	0.9997	0.5438	46,29 %

Tabla C.11: Comparación de técnicas de supresión ($\texttt{opt} = 3$, FakeTorino, 512 *shots*). Se reportan PST, $\mathcal{F}_H$ y los factores extraídos para cada variante. *Nota:* la fila de *a*=4 sin supresión ($PST = 3{,}52\,\%$, $\mathcal{F}_H = 0{,}0039$) corresponde a la corrida del estudio de mitigación; la línea base equivalente reportada en la Tabla C.9 ($PST = 4{,}30\,\%$, $\mathcal{F}_H = 0{,}0010$) corresponde a una corrida independiente con la misma configuración nominal pero distinta semilla de *sampling*. Las diferencias del orden de un punto porcentual son compatibles con la variabilidad esperada en distribuciones cercanas a la uniformidad sobre $M = 512$ *shots*.

a	Variante	PST (%)	$\mathcal{F}_H$	Factores	Estado
4	Sin supresión	3.52	0.0039	$\{3, 5\}$	✓
4	DD ($XY4$)	3.91	0.0073	$\{3, 5\}$	✓
4	PT (activo)	2.34	0.0057	$\{3, 5\}$	✓
4	DD + PT	3.32	0.0029	$\{3, 5\}$	✓
7	Sin supresión	56.64	0.4029	$\{3, 5\}$	✓
7	DD ($XY4$)	49.61	0.3300	$\{3, 5\}$	✓
7	PT (activo)	54.69	0.3998	$\{3, 5\}$	✓
7	DD + PT	46.48	0.2906	$\{3, 5\}$	✓

C.5 Tablas de resultados de ejecución en hardware IBM

Tabla C.12: Impacto del nivel de optimización del transpilador en hardware real (`ibm_torino`, $a = 4$, $\texttt{approx} = 0{,}7$, 4096 *shots*, Pauli Twirling activado). Se reportan la profundidad 2Q, el conteo de compuertas CZ, el número de SWAPs estimados, la señal PST y la fidelidad $\mathcal{F}_H$.

Nivel opt.	Depth 2Q	CZ	SWAPs est.	PST_{HW} (%)	$\mathcal{F}_H^{\text{HW}}$	Outcomes únicos
0	770	1220	113	1.1	0.011	512
1	526	779	65	3.1	0.031	505
2	116	116	0	70.8	0.529	57

Bibliografía

[1] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*. IEEE Comput. Soc. Press, pp. 124–134.

[2] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, 7 2018. [Online]. Available: http://arxiv.org/abs/1801.00862 http://dx.doi.org/10.22331/q-2018-08-06-79

[3] M. AbuGhanem, "Ibm quantum computers: evolution, performance, and future directions," *Journal of Supercomputing*, vol. 81, pp. 4–5, 4 2025. [Online]. Available: https://link.springer.com/article/10.1007/s11227-025-07047-7

[4] U. de Nariño, "Universidad de nariño – udenar." [Online]. Available: https://www.udenar.edu.co/

[5] U. Skosana and M. Tame, "Demonstration of shor factoring algorithm for $n = 21$ on ibm quantum processors," *Scientific Reports*, vol. 11, p. 16599, 8 2021.

[6] B. P. Lanyon, T. J. Weinhold, N. K. Langford, M. Barbieri, D. F. V. James, A. Gilchrist, and A. G. White, "Experimental demonstration of a compiled version of shor's algorithm with quantum entanglement," *Physical Review Letters*, vol. 99, p. 250505, 12 2007.

[7] J. A. Smolin, G. Smith, and A. Vargo, "Oversimplifying quantum factoring," *Nature*, vol. 499, no. 7457, pp. 163–165, 2013.

[8] G. Li, Y. Ding, and Y. Xie, "Tackling the qubit mapping problem for nisq-era quantum devices," 2018. [Online]. Available: https://arxiv.org/abs/1809.02573

[9] N. Ezzell, B. Pokharel, L. Tewala, G. Quiroz, and D. A. Lidar, "Dynamical decoupling for superconducting qubits: A performance survey," *Physical Review Applied*, vol. 20, no. 6, Dec. 2023. [Online]. Available: http://dx.doi.org/10.1103/PhysRevApplied.20.064027

[10] Z. Cai and S. Benjamin, "Constructing smaller pauli twirling sets for arbitrary error channels," *arXiv preprint arXiv:1807.04973*, 2018, https://arxiv.org/abs/1807.04973.

[11] J. D. Hidary, *A Brief History of Quantum Computing*. Springer International Publishing, 2021, pp. 15–21.

[12] J. Singh and M. Singh, "Evolution in quantum computing," in *2016 International Conference System Modeling & Advancement in Research Trends (SMART)*. IEEE, 2016, pp. 267–270.

[13] T. Norsen, *Foundations of Quantum Mechanics*. Springer International Publishing, 2017.

[14] N. Zettili, *Quantum mechanics : concepts and applications*, second edition ed. Wiley, 2009.

[15] H. Ma, M. Govoni, and G. Galli, "Quantum simulations of materials on near-term quantum computers," 2 2020.

[16] G. Kumar, S. Yadav, A. Mukherjee, V. Hassija, and M. Guizani, "Recent advances in quantum computing for drug discovery and development," *IEEE Access*, vol. 12, pp. 1–4, 2024.

[17] W. Buchanan and A. Woodward, "Will quantum computers be the end of public key encryption?" *Journal of Cyber Security Technology*, vol. 1, pp. 1–22, 1 2017.

[18] A. W. Harrow and A. Montanaro, "Quantum computational supremacy," *Nature*, vol. 549, pp. 203–209, 9 2017. [Online]. Available: https://www.nature.com/articles/nature23458

[19] IBM, "Compute resources | ibm quantum platform," 2025. [Online]. Available: https://quantum.cloud.ibm.com/computers

[20] J. M. V. D'angelo, G. D. Manresa, A. Camps, and J. Schindler, "Insights and limitations of shor's factorization algorithm on ibm real quantum computers," 2025. [Online]. Available: https://hdl.handle.net/2117/439250

[21] CLASSICQ, "Quantum software development: A similar but new paradigm," 10 2025. [Online]. Available: https://www.classiq.io/insights/quantum-vs-classical-software-development

[22] IBM, "Ibm quantum computing | home," 2025. [Online]. Available: https://www.ibm.com/quantum

[23] D. Kleppner and R. Jackiw, "One hundred years of quantum physics," 8 2000.

[24] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*, 10th ed. Cambridge University Press, 2024.

[25] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang, "Experimental realization of shor's quantum factoring algorithm using nuclear magnetic resonance," *Nature*, vol. 414, pp. 883–887, 12 2001.

[26] J. Roffe, "Quantum error correction: An introductory guide," vol. 60, pp. 226–245, 7 2019.

[27] C.-Y. Lu, D. E. Browne, T. Yang, and J.-W. Pan, "Demonstration of shor's quantum factoring algorithm using photonic qubits," 12 2007.

[28] A. Politi, J. C. F. Matthews, and J. L. O'Brien, "Shor quantum factoring algorithm on a photonic chip," *Science*, vol. 325, pp. 1221–1221, 9 2009.

[29] M. Rossi, L. Asproni, D. Caputo, S. Rossi, A. Cusinato, R. Marini, A. Agosti, and M. Magagnini, "Using shor algorithm on near term quantum computers: a reduced version," *Quantum Machine Intelligence*, vol. 4, p. 18, 12 2022.

[30] A. Saxena, A. Shukla, and A. Pathak, "A hybrid scheme for prime factorization and its experimental implementation using ibm quantum processor," *Quantum Information Processing*, vol. 20, p. 112, 3 2021.

[31] IBM, "Ibm quantum computing | qiskit." [Online]. Available: https://www.ibm.com/quantum/qiskit

[32] C. DeCusatis and E. Mcgettrick, "Near term implementation of shor's algorithm using qiskit," in *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 1 2021, pp. 1564–1568.

[33] IBM, "Ibm quantum platform." [Online]. Available: https://quantum.cloud.ibm.com/

[34] U. de los Andes, "El primer computador cuántico llega a colombia | universidad de los andes," 11 2024. [Online]. Available: https://www.uniandes.edu.co/es/noticias/ciencias-aplicadas/el-primer-computador-cuantico-llega-a-colombia

[35] M. de Ciencias de Colombia, "Convocatoria orquídeas: Mujeres en inteligencia artificial, ciencias y tecnologías cuánticas 2025 | convocatoria 963 | minciencias," 2025. [Online]. Available: https://minciencias.gov.co/convocatorias/convocatoria-orquideas-mujeres-en-inteligencia-artificial-ciencias-y-tecnologias

[36] F. A. A. Escobar, "Convocatoria premios - fundación alejandro Ángel escobar," 2020. [Online]. Available: https://www.faae.org.co/convocatoria-premios/

[37] Elsevier. (2026) Scopus | base de datos de resúmenes y citas. Accedido: 12 de junio de 2026. [Online]. Available: https://www.elsevier.com/es-mx/products/scopus

[38] Scopus. (2026) Scopus preview - sources. Accedido: 12 de junio de 2026. [Online]. Available: https://www.scopus.com/sources.uri?zone=TopNavBar&origin=

[39] T. G. Wong, *Introduction to Classical and Quantum Computing*. Rooted Grove, 2022.

[40] J. Watrous, *The Theory of Quantum Information*. Cambridge University Press, 2018.

[41] F. Laloë, "Do we really understand quantum mechanics? strange correlations, paradoxes, and theorems," *American Journal of Physics*, vol. 69, no. 6, p. 655–701, 2001. [Online]. Available: http://dx.doi.org/10.1119/1.1356698

[42] D. Brand, I. Sinayskiy, and F. Petruccione, "Markovian noise modelling and parameter extraction framework for quantum devices," *arXiv preprint arXiv:2202.04474*, 2022, https://arxiv.org/abs/2202.04474.

[43] IBM, "Crear modelos de ruido," 2026. [Online]. Available: https://quantum.cloud.ibm.com/docs/es/guides/build-noise-models

[44] M. G. Lammers, F. H. Holik, and A. Fernández, "Quantum resource management in the nisq era: Implications and perspectives from software engineering," 8 2025. [Online]. Available: https://arxiv.org/pdf/2508.05697

[45] "Benchmarking Quantum Processor Performance at Scale." [Online]. Available: https://arxiv.org/pdf/2311.05933

[46] I. Quantum, "Fidelidad de compuertas de 2q," 2025, incluye fidelidades de ECR. [Online]. Available: https://www.ibm.com/quantum/blog/quantum-volume-256

[47] ——, "Ecrgate | ibm quantum documentation," 2025. [Online]. Available: https://quantum.cloud.ibm.com/docs/api/qiskit/qiskit.circuit.library.ECRGate

[48] A. Szasz, E. Younis, and W. A. de Jong, "Numerical circuit synthesis and compilation for multi-state preparation," arXiv preprint, May 2023, https://arxiv.org/abs/2305.01816.

[49] N. S. Sáenz de Buruaga, R. Bistroń, M. Rudziński, R. M. C. Pereira, K. Życzkowski, and P. Ribeiro, "Fidelity decay and error accumulation in random quantum circuits," arXiv preprint, Apr. 2024, https://arxiv.org/abs/2404.11444.

[50] A. del artículo IEEE (ver página), "The transmon qubit for electromagnetics engineers," 2023. [Online]. Available: https://arxiv.org/pdf/2106.11352

[51] I. cloud, "Información sobre qpu | documentación de ibm quantum," 2025. [Online]. Available: https://quantum.cloud.ibm.com/docs/en/guides/qpu-information#coupling-map

[52] I. Quantum, "swapgate," 2025, sWAP. [Online]. Available: https://quantum.cloud.ibm.com/docs/es/api/qiskit/qiskit.circuit.library.SwapGate

[53] ——, "Transpiler," https://quantum.cloud.ibm.com/docs/es/guides/transpile, 2025, accedido: 2025-03-30.

[54] ——, "Defaults and configuration options," https://quantum.cloud.ibm.com/docs/guides/defaults-and-configuration-options, 2025, accedido: 2025-03-30.

[55] ——, "Transpiler stages," https://quantum.cloud.ibm.com/docs/guides/transpiler-stages, 2025, accedido: 2025-03-30.

[56] ——, "Set optimization level," https://quantum.cloud.ibm.com/docs/guides/set-optimization, 2025, accedido: 2025-03-30.

[57] ——, "Transpilation optimizations with sabre," https://quantum.cloud.ibm.com/docs/es/tutorials/transpilation-optimizations-with-sabre, 2025, accedido: 2025-03-30.

[58] ——, "Optimizaciones de transpilación con sabre | ibm quantum documentation," 2026. [Online]. Available: https://quantum.cloud.ibm.com/docs/es/tutorials/transpilation-optimizations-with-sabre

[59] L. Viola, E. Knill, and S. Lloyd, "Dynamical decoupling of open quantum systems," *Physical Review Letters*, vol. 82, no. 12, p. 2417–2421, Mar. 1999. [Online]. Available: http://dx.doi.org/10.1103/PhysRevLett.82.2417

[60] E. L. Hahn, "Spin echoes," *Physical Review*, vol. 80, no. 4, pp. 580–594, 1950.

[61] A. A. Maudsley, "Modified carr-purcell-meiboom-gill sequence for nmr fourier imaging applications," *Journal of Magnetic Resonance*, vol. 69, no. 3, pp. 488–491, 1986.

[62] A. Rahman, D. J. Egger, and C. Arenz, "Learning how to dynamically decouple by optimizing rotational gates," *Physical Review Applied*, vol. 22, no. 5, Nov. 2024. [Online]. Available: http://dx.doi.org/10.1103/PhysRevApplied.22.054074

[63] J. J. Wallman and J. Emerson, "Noise tailoring for scalable quantum computation via randomized compiling," *Physical Review A*, vol. 94, no. 5, p. 052325, 2016, https://journals.aps.org/pra/abstract/10.1103/PhysRevA.94.052325.

[64] IBM Quantum, "Técnicas de mitigación y supresión de errores," abril 2026. [Online]. Available: https://quantum.cloud.ibm.com/docs/es/guides/error-mitigation-and-suppression-techniques

[65] C. A. Fuchs, "Distinguishability and accessible information in quantum theory," Ph.D. dissertation, University of New Mexico, 1995, arXiv:quant-ph/9601020.

[66] P. Bagourd, J. Jang-Jaccard, V. Lenders, A. Mermoud, T. Hoefler, and C. Hempel, "Shor's algorithm on current quantum platforms," *arXiv preprint*, 2026.

[67] IBM Quantum, "qiskit.quantum_info.hellinger_fidelity," https://quantum.cloud.ibm.com/docs/api/qiskit/2.0/quantum_info, 2026, documentación oficial de Qiskit, consultado el 14 de mayo de 2026.

[68] ——, "Updating how we measure quantum quality and speed," https://www.ibm.com/quantum/blog/quantum-metric-layer-fidelity, 2023, iBM Quantum Blog, consultado el 14 de mayo de 2026.

[69] C. V. Freire, "Quantum shor's algorithm," https://github.com/christophervillotafreire/quantum-shors-algorithm, 2026, repositorio de GitHub, consultado el 14 de mayo de 2026.