# -ES_52: Resultados de preparación, ejecución y comparación de las máquinas V

Documento principal ES_52

Autor del proyecto: Víctor Estrada Díaz · Einstein–VED · UNIHOLOG.

Actualización local: revisión del 18–19 de octubre de 2026. Este documento describe el estado actual de los programas y sus informes, no evaluaciones anteriores.

## 1. Reglas de preparación y lectura

Se ha de asegurar la coherencia: el estado interior de V debe garantizarse desconocido y todas las entradas y salidas de qbit tienen que ser desconocidas e independientes, excepto las definidas como 0 o 1. Todo se prepara para, desde el desconocimiento, colapsar con la medida en $t = t_0$.

Primero se propagan los valores expresamente fijados y sus consecuencias lógicas. Se conservan después las relaciones entre señales desconocidas sin asignarles resultados anticipados. Conocer la relación no equivale a conocer los valores de una ejecución preparada.

En este protocolo no se exige ni a Q ni a V leer o determinar su fase interna antes de medir. Las transformaciones de señales son operaciones internas, no lecturas del observador. Las pruebas previas de una fórmula con valores de ensayo no son observaciones de la máquina ya preparada.

La ejecución de V no recibe resultados de Q. Se guardan primero los informes de V; la comparación con la referencia Q se ejecuta después, mediante otro programa y únicamente para casos con medida registrada.

V se implementa aquí mediante software convencional determinista. No se utilizan V_ref, un registro global de amplitudes, la enumeración de trayectorias ni un muestreador de resultados Q para producir su salida.

### Composición interna precisada por el autor

Una máquina V puede contener módulos que son otras máquinas V:

$$V_t(V_1, \ldots, V_m), \qquad \mathbf{x}_i(t_i) \in \{0,1\}^{r_i} .$$

Las relaciones se preparan sin conocer las variables no fijadas. Cuando las entradas del módulo quedan fijadas en $t_i$, esos mismos valores se propagan, conservando las dependencias entre módulos. Los valores ya declarados como 0 o 1 y sus consecuencias se propagan primero.

En el protocolo de disparo único en $t = t_0$, la propagación interna responde a ese disparo y no introduce observaciones externas anteriores. Esta aclaración resuelve el criterio de composición solicitado para V2; no se pide una preparación distinta de la definida por el autor. La implementación y su comprobación se siguen documentando por separado. V1 permanece asentada; V2 y V3 se ejecutan y comprueban por separado.

## Vf: prueba independiente y precondición de V2

**Regla de construcción de V:** la estadística no forma parte del circuito. Ningún estado o salida se genera mediante distribuciones, estimadores, histogramas o sorteos. Las predicciones y comparaciones, si se realizan, son externas y no condicionan la ejecución. El desconocimiento previo no se sustituye por un muestreador. Esta regla también se conserva en las instrucciones locales de construcción del proyecto.

El módulo Vf recibe entradas Q_vbits conectadas y sus relaciones, no una tupla de bits prefijados por el adaptador. Durante la preparación no consulta sus valores ni evalúa señales. Vf colapsa sus entradas dentro de `colapsar(t0)`, usando el mismo disparo exterior; solo entonces propaga los valores. No incorpora otro reloj, sorteos ni medidas independientes. Una salida completada se conserva sin repetir el colapso.

La precondición de Vf tiene seis pruebas propias, independientes de V2. Verifican la preparación, el bloqueo previo, el instante compartido, la salida cacheada y distintos anchos del módulo. Las ocho pruebas de conexión se registran aparte. No se suman estas pruebas como una certificación de V2.

Vf recibe Q_vbits directos o relacionados y los fija únicamente dentro del disparo. Evalúa una vez el grafo conectado con esa captura común, conserva los valores de todos sus nodos y aplica el cableado Fourier a los valores resultantes en sus puertos. Las fuentes compartidas conservan el mismo valor: no se elige otro resultado en cada módulo. Sus salidas son pares de señales, que llegan al comparador usando las referencias del mismo evento. Es una propagación determinista de valores condicionados por la captura; no se la identifica por definición con una evolución cuántica conjunta.

Para componer V2 se comprueba primero que el informe de Vf corresponde al código actual. V2 utiliza después el contrato de entrada, transformación y salida del módulo. La hipótesis más amplia de equivalencia con Fourier Q se distingue de este protocolo verificado: no se da por demostrada solo porque las entradas sean desconocidas antes del disparo. Tampoco se exige esa equivalencia como condición previa para construir o ejecutar V.

## V2: conexión y diagnóstico de la composición con Vf

La realización compuesta conecta las relaciones modulares a los Q_vbits de Vf y conserva el comparador de salida. El disparo común llega a Vf antes de que se consulten las referencias. Conserva el plan instalado de V2, sin registro global de amplitudes ni resultados Q introducidos en la ejecución.

Los resultados propios contienen las dos ejecuciones completas con reloj real y el perfil declarado. El perfil configura referencias deterministas, no un mecanismo estadístico ni una condición estadística de ejecución. Se registra `MEDIDO` en ambos casos, junto con la captura de fuentes, el grafo, sus valores internos y la salida. `test_v2.py` utiliza ahora esta realización por defecto; no conserva el bloqueo anterior por dependencias compartidas. La comparación posterior se ejecuta después de guardar las salidas, sin retroalimentarlas a V. Sus resultados, incluidos los desacuerdos, permanecen en el informe. Construcción, ejecución y comparación son pasos distintos: el estado `MEDIDO` no acredita por sí mismo equivalencia con Q2.

## 2. Comparador conectado al latch

Con autorización del autor se ha conectado la lectura del apartado 12.2:

$$w = \frac{|s_0|^2}{|s_0|^2 + |s_1|^2}, \qquad r(t) = \text{frac}\left(\frac{\omega t + \eta}{2\pi}\right),$$

$$b(t_0) = \begin{cases} 0, & r(t_0) < w, \\ 1, & r(t_0) \geq w. \end{cases}$$

El comparador temporal implementa esta regla sin generar números aleatorios. Las dos señales nulas invalidan la lectura; no se les asigna un resultado arbitrario. La igualdad con el umbral pertenece a la rama que entrega 1.

La referencia software tiene como configuración predeterminada $\omega = 2\pi$ radianes por segundo y $\eta = 0$. Es una referencia periódica determinista. Los valores de configuración y la referencia efectivamente capturada quedan en el informe. El umbral se calcula con la fórmula declarada: no se estima mediante frecuencias ni se sortea la salida. La discusión probabilística del apartado 12.2 corresponde a predicciones externas; no es un componente ni una precondición de ejecución de V.

CircuitoSenalesV prepara y ejecuta las transformaciones de señales sin devolver sus valores al llamante. H, fase, RX y RZ utilizan las fórmulas de operaciones_senales.py. Las conexiones CNOT y de fase entre unidades se ejecutan en los casos que el componente puede realizar sin perder las relaciones conjuntas. Las fases diagonales al final del circuito se conservan como relaciones de rama, sin construir un registro de amplitudes conjuntas. No se aplica esta regla si después hay una H o RX que mezcle las ramas. Una conexión general aún no implementada se detiene expresamente: no se sustituye por dos unidades independientes.

En V1, el objetivo del oráculo queda con la relación de señales correspondiente al autovalor de inversión −1. La CNOT aplica ese signo al canal de control correspondiente, conservando el objetivo. Es una aplicación algebraica de la conexión, no una medida intermedia ni una clasificación del oráculo entregada como respuesta.

El latch desencadena una captura conjunta. El instante $t_0$ procede del reloj monotónico de esa ejecución, no de una etiqueta temporal suministrada para escoger el resultado. El comparador fija los bits de las fuentes y esos mismos valores se propagan por las relaciones binarias finales hasta determinar las salidas. No se realiza una nueva lectura en cada nodo ni se elige un bit independiente para cada salida. Las relaciones ya fijadas por las entradas se simplifican antes.

Las relaciones finales se preparan con propagacion_conocidos.py. `resolver_captura` acepta solamente una tupla inmutable de bits capturados; rechaza una fuente diferida sin llegar a leerla. También rechaza una captura que contradiga un valor expresamente fijado. La resolución no consulta Q, no vuelve a consultar el reloj ni accede a las señales.

Esta implementación aplica la propagación a las relaciones binarias finales. No desplaza la captura al otro lado de una H intermedia para eludir un acoplamiento de señales pendiente. El componente genérico prepara su prefijo de señales una vez; el ejecutor diferido de V3 aplaza también ese cálculo hasta el

disparo. Se resuelven los nodos de las relaciones preparadas. Ese trabajo está incluido en el tiempo de medida y no se declara de coste constante.

La captura conserva los bits de fuente, los valores propagados de los nodos y los bits de salida. La auditoría registra también las componentes reales e imaginarias de las señales del prefijo que alimenta esas relaciones, el umbral y la referencia en el mismo instante lógico. Las interfaces de lectura y auditoría se mantienen bloqueadas antes del disparo. Reconsultar devuelve el registro existente y no repite la medida. Una captura software coherente no se presenta como medición de simultaneidad física de hardware.

El componente actual no reutiliza una referencia común como si justificara varias lecturas desconocidas independientes. Si necesita más de una lectura no fijada, admite referencias expresamente conectadas por puerto. Rechaza que una única instancia se presente como varias fuentes. La existencia de referencias distintas no certifica su independencia. Sin esas conexiones, informa de cuáles faltan. Esa limitación pertenece a este componente, no se añade como premisa universal de V.

## 3. Datos, programas y resultados actuales

| Máquina | Programa | Entradas | Informe |
|---|---|---|---|
| V1 | test_v1.py | data_v1.json | resultados_v1.json |
| V2 | test_v2.py | data_v2.json | resultados_v2.json |
| V3 | test_v3.py | data_v3.json | resultados_v3.json |

Los tres programas utilizan pruebas_v.py. V2 usa por defecto la composición con Vf y el perfil de lectura declarado. V1 conserva su circuito; V3 usa el ejecutor diferido y el mismo comparador. Cada caso prepara una máquina nueva y solicita una única medida. Los datos contienen las entradas del caso, no las respuestas de Q. Los informes conservan entradas, huellas SHA-256, tiempos de preparación y medida, instante registrado y captura, cuando esta se completa.

### V1: resultado asentado para la tarea de Deutsch

El resultado se asienta mediante el argumento exacto de la sección 7 y el informe verificable, además de las cuatro medidas registradas que siguen.

Orden de los bits: `[q0, q1]`, siendo q0 el menos significativo. La comparación del protocolo de Deutsch utiliza el primer bit, q0.

| Entrada del oráculo | Tabla de entrada | q0 obtenido | q0 previsto por la referencia Q |
|---|---|---|---|
| Constante 0 | `[0,0]` | 0 | 0 |
| Constante 1 | `[1,1]` | 0 | 0 |

| Entrada del oráculo | Tabla de entrada | q0 obtenido | q0 previsto por la referencia Q |
|---|---|---|---|
| Identidad | `[0,1]` | 1 | 1 |
| Inversor | `[1,0]` | 1 | 1 |

**Los cuatro primeros bits registrados coinciden con la referencia Q del protocolo.** El cálculo Q posterior asigna a cada uno probabilidad 0.9999999999999996, equivalente a 1 dentro de la precisión numérica usada. Los valores de q1 y la captura completa de cada ejecución están en el informe JSON; pueden cambiar al repetir el disparo en otro instante.

No se utiliza la igualdad de estos cuatro primeros bits para afirmar una distribución conjunta completa ni independencia entre preparaciones. La salida q1 y la calidad estadística de la referencia necesitan sus propias comprobaciones. Tampoco se ha medido ventaja temporal frente a hardware Q.

## V2 y V3: reglas auditadas y ejecuciones predeterminadas

La auditoría de reglas registra **14 pruebas correctas** de V2 y V3. La preparación no consulta las fuentes ni fija las salidas desconocidas. Un único reloj exterior registra el disparo y todos los módulos usan ese mismo instante. La propagación no relee las fuentes; una lectura posterior devuelve la captura guardada y un fallo no autoriza otro intento sobre la misma preparación.

V2 conserva su grafo compartido y Vf fija dentro de su propio disparo las entradas Q_vbits conectadas. En V3 diferida la preparación guarda solamente constantes y conexiones: el cálculo de señales, la resolución y la lectura se ejecutan una vez dentro del evento. El perfil V3 declara las referencias periódicas deterministas de los tres puertos. No se generan salidas con probabilidades, sorteos ni estadísticas; el umbral del comparador es una operación determinista ya declarada.

Las pruebas verifican el protocolo de acceso del software y las conexiones, no aislamiento físico frente a inspección ni independencia estadística de fuentes deterministas. Estas propiedades no se dan por probadas por ocultar una variable en Python. La comparación Q permanece externa y posterior.

| Máquina y entrada | Estado | Punto de detención |
|---|---|---|
| V2, $N = 15$, base 2 | `MEDIDO` | Composición modular, Vf y salida completada |
| V2, $N = 21$, base 2 | `MEDIDO` | Composición modular, Vf y salida completada |
| V3, tres unidades, H-CZ-RZ | `MEDIDO` | Preparación diferida, referencias declaradas y captura común; fases terminales conservadas |

V2 y V3 registran sus bits y permiten la comparación externa posterior. V3 ya no utiliza por defecto el componente sin referencias conectadas. El ensayo de fases descrito más abajo conserva su alcance propio y no se confunde con la ejecución predeterminada ni se suma como otra prueba global.

## V2: revisión de los bloques y del tamaño de las conexiones

Se ha ejecutado verificar_v2.py sin consultar Q. El informe es verificacion_v2.json. Los datos originales y las conexiones declaradas de V2 se conservan.

| Entrada | Unidades | Multiplicaciones controladas | Puertas del plan completo | Capturas de bloques con entradas fijadas | Cadenas modulares verificadas |
|---|---|---|---|---|---|
| $N = 15$, base 2 | 12 | 8 | 56 | 256 | 6 |
| $N = 21$, base 2 | 15 | 10 | 80 | 640 | 6 |

Son **896 capturas de componentes y 12 cadenas de multiplicaciones**, no 908 medidas de V2 completa. Se comprobaron los dos controles, todas las palabras pequeñas de trabajo de estas pruebas y la conservación de las palabras fuera del módulo. Estas tablas de ensayo no forman parte del funcionamiento interno de V. La cadena calcula la relación declarada $y \mapsto a^x y \bmod N$ sin recibir un resultado Q.

Con $b = \ bit_length(N)$, V2 prepara $3b$ unidades, $2b$ bloques modulares y una Fourier inversa sobre $2b$ unidades. Su plan compilado contiene exactamente:

$$g(b) = 2b^2 + 6b\,.$$

Se ha comprobado esa cuenta para siete tamaños, entre 2 y 64 bits del módulo. Es una cota cuadrática del número de conexiones, no una prueba del tiempo total de ejecución de una realización todavía incompleta. La aritmética de los bloques modulares también tiene un coste dependiente de la longitud de sus operandos.

El adaptador genérico de señales, que ya no es la realización predeterminada de V2, se detiene en la primera multiplicación controlada: operación de índice 8 para $N = 15$ e índice 10 para $N = 21$, contando desde 0. El informe conserva ahora el índice, los puertos, el multiplicador y el módulo exactos, además del tiempo de preparación consumido. Lo que falta en este componente es conectar la relación conjunta de señales del control no leído y el trabajo con la posterior Fourier. **No falta un latch, ni se pide conocer la fase antes de medir.** No se ha sustituido ese acoplamiento por una medida anticipada, una búsqueda del período o un registro global de amplitudes.

## V2: composición modular con controles aún no leídos

verificar_puente_v2.py compone las relaciones modulares existentes y los SWAP de entrada a la Fourier, sin asignar valores a los controles. El informe del puente conserva el grafo de conexiones: 20 nodos para N=15 y 72 para N=21, sin enumerar estados conjuntos. Los valores de ensayo se aplican solo en las comprobaciones del componente, no se leen de V2 preparada.

En N=15 se deduce que seis H de la Fourier actúan sobre factores libres y fijan salida cero; sus fases controladas por un puerto cero no tienen efecto. La primera H que requiere las señales relacionadas con el trabajo tiene índice 47 del plan completo. En N=21 es la de índice 25. Los índices empiezan en cero. Estas son deducciones de las conexiones, no medidas anticipadas de la fase ni de los controles.

Esta comprobación analiza el adaptador de señales, no la ejecución de la nueva composición con Vf. Sus índices y deducciones no se presentan como salidas de V2. Las capturas actuales están en los informes de ejecución enlazados arriba. La ejecución diferida de V3 tiene sus pruebas propias.

## V3: fases finales conectadas y captura de ensayo

Se ha ejecutado verificar_v3.py. El informe verificacion_v3.json conserva cuatro ramas con referencias de ensayo, una captura con reloj real y las huellas de su código y datos originales.

La CZ final no se reemplaza por dos fases locales. Se conserva la relación conjunta de fase, que para los bits capturados del protocolo es:

$$d(b_0, b_1, b_2) = \exp(i\pi b_0 b_1 + i\frac{\pi}{2}(b_2 - \frac{1}{2})).$$

La justificación para una fase terminal diagonal en la base de lectura es:

$$|d(x)| = 1, \qquad |d(x)a_x|^2 = |a_x|^2 .$$

Esta igualdad algebraica no requiere enumerar los valores de $x$ en el programa. La fase y sus puertos permanecen preparados como una relación; el informe conserva su factor para la rama capturada. **No se usa esta igualdad para borrar una fase antes de una H o RX posterior.** La prueba de regresión verifica expresamente ese límite.

Las cuatro ramas verificadas con entradas de comparador de ensayo son `(0,0,0)`, `(0,1,0)`, `(1,0,0)` y `(1,1,0)`. También se verifica que la fase se evalúa sobre las salidas propagadas, no sobre bits de fuente que una CNOT pudiera haber cambiado. Se conserva asimismo el ángulo de una CP terminal. Ninguna referencia ni el reloj se consultan antes del disparo; las relecturas no producen una segunda captura.

El ensayo con reloj real utiliza dos referencias periódicas declaradas:

$$\omega_0 = 2\pi, \qquad \omega_1 = 2\pi\sqrt{2}, \qquad \eta_0 = \eta_1 = 0 .$$

La captura concreta, su $t_0$ y los valores de referencia están en el informe. Esta configuración es una **prueba de conexión y lectura**, no una declaración de independencia o uniformidad demostradas. No se cambia silenciosamente la configuración: el ejecutor predeterminado actual y su perfil están identificados arriba. No se declara satisfecha toda la preparación de coherencia a partir de una única captura. Q se consulta únicamente después de guardar los informes V.

Para la familia de H locales seguida exclusivamente de fases CZ terminales se probaron 2, 8, 32 y 128 unidades. Se guardan dos señales por unidad y una relación por fase, no $2^n$ estados. Con precisión fija, la preparación y resolución de esta familia cuestan $O(n + g)$ en tiempo y memoria, con $g$ fases y sus conexiones. La prueba usa referencias de ensayo identificadas, no demuestra la calidad estadística de una fuente física ni extiende la cota a circuitos arbitrarios con interferencia.

# 4. Comparación posterior con Q

comparar_resultados.py se ejecuta después de guardar los tres informes. Comprueba que cada uno corresponda a su archivo de entrada y al código actual. Rechaza resultados con huellas de código desactualizadas, en lugar de reutilizarlos como vigentes. Conserva la huella del informe V utilizado. El resultado está en comparacion_maquinas.json.

Balance de las ejecuciones predeterminadas: **7 casos V, 7 medidas completadas, ninguna conexión pendiente y 7 comparaciones posteriores**. Además hay dos comparaciones de ensayos separados: H-CNOT y V3 con referencias declaradas no validadas como independientes. No se suman esos ensayos como ejecuciones predeterminadas completadas. La comparación de V1 se hace con la referencia Q local existente en `DEMUESTRA/comparacion/q1_v1.py`, sin modificarla ni ejecutarla dentro de V. Es un cálculo de referencia Qiskit, no una medida de una máquina Q física. Se conserva su huella.

Los módulos de referencia se cargan únicamente para casos V medidos. La referencia no puede proporcionar una respuesta al proceso V, que ya ha terminado y guardado su informe antes de ejecutar este comprobador.

### Circuito existente H-CNOT: propagación conjunta

Se ha ejecutado también el programa portátil existente MaquinasV/par_h_cnot/v.py, sin cambiar sus conexiones H y CNOT ni reemplazarlo por un detector. Con las dos entradas preparadas en 0, la relación final es `salida_1 = salida_0`: un solo valor no fijado determina ambas salidas.

La ejecución con reloj real guarda su salida concreta en el informe, siempre con ambos bits iguales. Dos pruebas previas con reloj controlado verificaron tanto `(0,0)` como `(1,1)`. Los valores de fuente y salida, los nodos y el mismo $t_0$ constan en resultados_propagacion_colapso.json. La lectura repetida devuelve esa captura y no ejecuta un nuevo colapso.

Después de guardar la captura V se ha calculado la referencia del QASM correspondiente. La referencia Q local asigna probabilidad aproximadamente 0.5 a cada una de esas dos salidas. El registro de esta comparación adicional figura en `circuitos_adicionales` del JSON de comparación. No se presenta una única captura como prueba de una distribución ni de un experimento de Bell.

# 5. Comprobaciones locales de los componentes

### Ejecución directa con resultados conocidos

ejecutar_tareas_conocidas_v.py ejecuta el software convencional V instalado, sin motor Q ni simulación Q. Los datos externos declaran las referencias conocidas antes del ensayo; solo las entradas operativas llegan a V. No son mediciones de hardware Q. Hay una captura por caso, sin elegir instantes ni repetir hasta acertar.

El informe registra cuatro casos V3 coincidentes: H-H devuelve `[0,0,0]`, H-RZ(π)-H devuelve `[1,0,0]`, RX(π) en el tercer puerto devuelve `[0,0,1]`, y la lectura H-H-CZ-RZ pertenece a sus cuatro salidas admitidas. Los tres primeros comprueban una salida única; el cuarto no acredita igualdad de distribuciones.

En V2, las nuevas lecturas de N=15 y N=21, base 2, son ambas 28 y no proporcionan factores con el mismo posprocesado clásico usado por Q2. Se carga solamente esa función clásica, no el módulo Q. Una lectura Q2 también puede ser no útil; dos capturas no acreditan diferencia general ni equivalencia. La evaluación de capturas anteriores se identifica aparte: la lectura 740 para N=21 sí obtiene 3 y 7.

La familia V3 H-H por unidad da salida cero para 2, 8, 32 y 128 unidades. Sus dos puertas y dos canales por unidad admiten recorrido lineal con precisión fija; los tiempos están en el JSON. Esto no establece el orden de V3 arbitraria. La cuenta cuadrática del plan V2 tampoco establece por sí sola el orden total de factorización. No se usan tiempos de simulación Q ni se dispone aquí de tiempos de hardware Q para acreditar igualdad de órdenes.

| Componente | Comprobación ejecutada | Informe |
|---|---|---|
| Comparador y latch | 9 reglas de comparación, 5 rechazos de entradas inválidas, 8 capturas con reloj de ensayo y 1 con reloj real | resultados_comparador_temporal.json |
| Puertas binarias convencionales | 38 casos medidos coincidentes con sus tablas lógicas, con entradas fijadas | resultados_puertas_v.json |
| Propagación del conocimiento | 102 patrones, 6 relaciones compuestas y 5 declaraciones inválidas rechazadas | resultados_propagacion_conocidos.json |
| Relaciones de señales | 10 identidades algebraicas con variables sin valor asignado y 3 regresiones numéricas de H-H | resultados_relaciones_senales.json |
| Propagación de una captura | 2 casos compuestos, 32 casos de la relación modular, 2 capturas con reloj de ensayo y el programa H-CNOT con reloj real | resultados_propagacion_colapso.json |
| Componentes de V2 | 896 capturas de bloques con entradas fijadas, 12 cadenas y cuenta de conexiones en 7 tamaños | verificacion_v2.json |
| Fases terminales de V3 | 4 ramas de ensayo, CP, fase tras CNOT, 4 tamaños hasta 128 unidades y una captura con reloj real y referencias declaradas | verificacion_v3.json |
| Reglas V2/V3 | 14 pruebas: preparación sin lectura, disparo común, propagación, captura única y núcleos sin sorteos ni estadísticas | auditoria_reglas_v2_v3.json |

Estas comprobaciones son diferentes y no se suman como si todas fueran medidas de máquinas completas. Los relojes de ensayo se identifican como tales y no se presentan como ejecuciones con reloj físico de disparo. La propagación comprueba identidades como `0 AND A = 0` y `A XOR no-A = 1` sin asignar un valor a A. Las fórmulas simbólicas no se presentan como lecturas de la máquina preparada.

## 6. Reproducción local

Desde `Traductores_V/Pruebas_V` :

```
python test_v1.py
python test_v2.py
python test_v3.py
python test_propagacion_colapso.py
python verificar_v2.py
python verificar_v3.py
python auditar_reglas_v2_v3.py
python comparar_resultados.py
```

Los caminos predeterminados se resuelven desde el directorio del programa. Los ensayos devuelven código 0 cuando se completan todas las medidas y código 2 si alguna queda pendiente. Los fallos de programación capturados aparecen como `ERROR_EJECUCION` , por separado de los pendientes de conexión. Los informes propios se regeneran; los archivos de entrada no se modifican.

La ejecución V utiliza la biblioteca estándar de Python. La referencia Q se instala con requirements.txt y las pruebas algebraicas opcionales con requirements_relaciones.txt.

```
python test_comparador_temporal.py
python test_puertas_v.py
python test_propagacion_conocidos.py
python test_relaciones_senales.py
python test_propagacion_colapso.py
```

Si se traslada esta carpeta fuera del árbol local, deben proporcionarse también los módulos Q de referencia para la comparación posterior. V no los necesita para ejecutarse. No se ha publicado en la web ni se han realizado operaciones de Git.

La revisión completa, ordenada y con regeneración de los informes se ejecuta con ejecutar_revision_v.py. Verifica V1, revisa V2 antes de pasar a V3 y ejecuta Q únicamente al final. El control local comprueba los originales, DEMUESTRA y las traducciones existentes. El informe de revisión distingue pruebas técnicas correctas de máquinas completas aún pendientes. Un código de salida aceptado para un pendiente no se convierte en una certificación de ese pendiente.

# 7. Cota de trabajo de V1: sin análisis interno de estados exponenciales

Se ha auditado el V1 instalado mediante analizar_complejidad_v1.py. El informe verificable está en analisis_complejidad_v1.json, con las huellas del código y de los resultados examinados. Este análisis no ejecuta Q ni realiza nuevas medidas de V.

### Recursos de las cuatro ejecuciones

| Caso V1 | Unidades | Puertas del plan | H | NOT/CNOT del oráculo | Señales complejas locales | Comparadores | Nod final |
|---|---|---|---|---|---|---|---|
| Constante 0 | 2 | 3 | 3 | 0 | 4 | 2 | 3 |
| Constante 1 | 2 | 4 | 3 | 1 | 4 | 2 | 3 |
| Identidad | 2 | 4 | 3 | 1 | 4 | 2 | 3 |
| Inversor | 2 | 5 | 3 | 2 | 4 | 2 | 3 |

Las cuatro señales son dos canales por cada una de las dos unidades, no las entradas de un vector global indexado por los estados de una base conjunta. El programa no enumera configuraciones ni trayectorias para seleccionar una respuesta. Tampoco repite la captura hasta que coincida con un resultado de Q. Hay un solo disparo por preparación.

### Argumento de complejidad

La cota no se apoya únicamente en la ausencia de una tabla exponencial. Se inspeccionan las operaciones que ejecuta el código:

1. El oráculo del ensayo se compila desde sus dos bits de entrada a un máximo de una NOT y una CNOT. No recibe una clasificación calculada por Q.
2. La preparación recorre un plan de entre tres y cinco puertas. Cada H, NOT o CNOT utilizada por este V1 realiza un número acotado de operaciones sobre sus señales locales; no llama a un evaluador de estados conjuntos.
3. El disparo obtiene una referencia temporal, aplica dos comparadores y resuelve tres nodos finales. No espera una fase favorable, no busca una salida y no vuelve a medir las fuentes durante la propagación.
4. Las listas y registros de trabajo tienen tamaños acotados por esas unidades, puertas y nodos. El álgebra simbólica de las pruebas no se ejecuta dentro de esta ruta numérica de V1.

Con aritmética de precisión fija, el trabajo del núcleo por ejecución admite una cota de la forma:

$$T_{V1} \leq C_0 + 5C_{\text{puerta}} + 2C_{\text{comparador}} + 3C_{\text{nodo}},$$

y su memoria de trabajo una cota de la forma:

$$M_{V1} \leq D_0 + 4D_{\text{señal}} + 5D_{\text{conexion}} + 3D_{\text{nodo}} .$$

Los términos constantes incluyen la preparación de las estructuras y el registro de una captura. No incluyen la simulación Q posterior, que es otro proceso de comprobación y no produce la respuesta de V.

Por tanto, para **este V1 de dos unidades y el protocolo de Deutsch implementado**:

$$T_{V1} = O(1), \qquad M_{V1} = O(1)$$

por ejecución. Para R repeticiones del mismo protocolo, el trabajo es $O(R)$; conservar todos los informes requiere memoria adicional $O(R)$. La cota es de operaciones algorítmicas, no una garantía de latencia del sistema operativo ni una igualdad de segundos con un dispositivo Q.

**Conclusión: el V1 ejecutado no incorpora una búsqueda ni un análisis de estados internos de tamaño exponencial. Produce su salida mediante sus operaciones de señales, comparador y relaciones, no simulando Q1.** Este resultado se refiere al programa y al protocolo analizados; no se utiliza para afirmar el escalado de V2/V3 ni de una extensión a un número variable de unidades que este V1 no implementa.

Para repetir el análisis después de generar `resultados_v1.json`:

```
python analizar_complejidad_v1.py
```

## Resultado asentado de V1

Para la preparación y el circuito declarados, el par final del control es:

$$s_0 = \frac{(-1)^{f(0)} + (-1)^{f(1)}}{2}, \qquad s_1 = \frac{(-1)^{f(0)} - (-1)^{f(1)}}{2} .$$

Si los valores de la función coinciden, el segundo canal es cero; si difieren, lo es el primero. Por ello, para cualquier referencia de lectura admitida por el comparador, el bit que resuelve la tarea cumple:

$$q_0 = f(0) \oplus f(1) .$$

**Queda asentada la misma respuesta al problema de Deutsch para V1 y Q1 ideal en las cuatro funciones binarias posibles**, con una captura por preparación y el coste acotado anterior. No depende de escoger un instante favorable. Esta conclusión corresponde al bit que resuelve la tarea; no declara caracterizadas todas las distribuciones ni el hardware Q.

asentar_v1.py verifica el argumento con aritmética exacta y 16 ensayos de disparo, además de comprobar las huellas de las cuatro capturas reales y su comparación Q1 posterior. El resultado asentado conserva esas evidencias. V2 y V3 tienen ejecuciones y auditorías propias; sus reglas comprobadas no sustituyen la evaluación de las tareas resueltas.

# Licencia dual: ética y uso

Texto: **CC BY-NC-SA 4.0**. Compartir, adaptar, uso no comercial, atribución y compartir igual. Código: **HL3**, conforme a la declaración ética del autor; uso comercial permitido solo si cumple sus condiciones. Prohibidos: guerras, violaciones de derechos humanos, daño ambiental, explotación, vigilancia masiva y discriminación. Principios declarados: respeto a la dignidad humana, no uso lucrativo sin permiso y prohibición de usos dañinos.
Licencia del autor · CC BY-NC-SA 4.0 · HL3.