UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Computació clàssica en entorns quàntics

Martín Garcia, Albert

Especialitat de Computació
Primavera 2022

Dirigit per Ametller Congost , Lluís

Resum

La criptografia moderna es basa en la dificultat del problema de factorització de naturals per a garantir una comunicació segura d'informació, de manera que el descobriment d'un algorisme polinòmic per a calcular la descomposició en primers en ordinadors quàntics va posar la computació quàntica en el punt de mira de la comunitat científica.

Des de finals del segle anterior, la recerca en aquest camp ha estat incessant, assolint fites monumentals com la creació física de computadors quàntics de centenars de qubits o la teleportació d'informació quàntica sobre més d'un miler quilòmetres, i no dona pas signes d'aturar-se. La importància de la recerca d'aquest camp per a la criptografia pràcticament garanteix que aquest model de computació esdevindrà prou accessible per a ser usat per a propòsits més enllà de la recerca.

Ja sigui investigació, implementació d'algorismes quàntics actuals, o disseny d'hipotètics processadors quàntics futurs, és imperatiu tenir la capacitat de representar informació clàssica en aquests computadors i de transformar-la per a obtenir el resultat desitjat. És comú en la literatura del camp assumir la capacitat del sistema d'aplicar qualsevol d'aquestes funcions que transformen informació clàssica, però rarament s'estudia com realitzar aquests càlculs en aquest nou model de computació ni se'n detallen els seus efectes sobre les complexitats temporals i espacials dels algorismes quàntics.

Diversos documents de recerca han estat publicats en els últims anys estudiant una gran diversitat de mètodes per a realitzar dits càlculs, però sovint aquests són dissenyats específicament per a algun algorisme o per a una certa arquitectura d'ordinador quàntic.

L'objectiu d'aquest treball és l'estudi de l'estat de l'art en la realització de càlculs clàssics en sistemes quàntics, la compilació i comparació de les dotzenes de dissenys publicats i la modificació d'aquests per a ser aplicables de manera general a qualsevol algorisme desitjat, aportant tota la informació necessària per a la implementació física d'aquests des de zero. Tot lector d'aquest document serà capaç d'obtenir un circuit capaç de realitzar qualsevol càlcul clàssic en un ordinador quàntic, i tindrà diverses opcions per a implementar cadascuna de les operacions per a que s'adapti a les necessitats del seu sistema o arquitectura.

En aquest document es mostren diversos dissenys capaços de realitzar diferents càlculs aritmètics sobre estats quàntics en superposició, se'n donen guies extensives detallant-ne la seva construcció i s'analitza l'impacte de cadascun en les complexitats espacials i temporals dels algorismes en els quals s'usin.

Resumen

La criptografía moderna se basa en la dificultad del problema de factorización de naturales garantizar una comunicación segura de información, por lo que el descubrimiento de un algoritmo polinómico para calcular la descomposición en primeros en ordenadores cuánticos puso la computación cuántica en el punto de mira de la comunidad científica.

Desde finales del siglo anterior, la investigación en este campo ha sido incesante, alcanzando hitos monumentales como la creación física de computadores cuánticos de cientos de cúbits o la teleportación de información cuántica sobre más de un millar kilómetros, y no da signos de detenerse. La importancia de la investigación de este campo para la criptografía prácticamente garantiza que este modelo de computación será suficientemente accesible para ser usado para propósitos más allá de la investigación.

Ya sea investigación, implementación de algoritmos cuánticos actuales, o diseño de hipotéticos procesadores cuánticos futuros, es imperativo tener la capacidad de representar información clásica en estos computadores y transformarla para obtener el resultado deseado. Es común en la literatura del campo asumir la capacidad del sistema de aplicar cualquiera de estas funciones que transforman información clásica, pero raramente se estudia cómo realizar estos cálculos en este nuevo modelo de computación ni se detallan sus efectos sobre las complejidades temporales y espaciales de los algoritmos cuánticos.

Varios documentos de investigación han sido publicados en los últimos años estudiando una gran diversidad de métodos para realizar dichos cálculos, pero a menudo estos son diseñados específicamente para algún algoritmo o para una cierta arquitectura de ordenador cuántico.

El objetivo de este trabajo es el estudio del estado del arte en la realización de cálculos clásicos en sistemas cuánticos, la compilación y comparación de las docenas de diseños publicados y su modificación para ser aplicables de modo general a cualquier algoritmo deseado, aportando toda la información necesaria para la implementación física de estos desde cero. Todo lector de este documento será capaz de obtener un circuito capaz de realizar cualquier cálculo clásico en un ordenador cuántico, y tendrá varias opciones para implementar cada una de las operaciones para que se adapte a las necesidades de su sistema o arquitectura.

En este documento se muestran varios diseños capaces de realizar diferentes cálculos aritméticos sobre estados cuánticos en superposición, se dan guías extensivas detallando su construcción y se analiza el impacto de cada uno en las complejidades espaciales y temporales de los algoritmos en los que se usen.

Abstract

Modern cryptography is based on the difficulty of the factoring problem to ensure secure communication, so the discovery of a polynomial-time algorithm to find prime factors in a quantum computer placed quantum computation in the spotlight of the scientific community.

Ever since the end of the last century, research in this field has been incessant, achieving tremendous feats such as the creation of a quantum computer with hundreds of qubits or the teleportation of a quantum state over more than a thousand kilometers, and it does not show signs of stopping anytime soon. The importance of the research in this field for cryptography practically guarantees that this model of computation will become accessible enough to be used for purposes beyond research.

Be it research, the implementation of current quantum algorithms, or the design of hypothetical future quantum processors, it is imperative to have the ability to encode classical information in these computers and transform it into the desired result. It is common in the literature to assume the ability of the system to apply any of these functions that transform classical information, but it is quite rare to find a discussion about how to perform these operations in this new model of computation or a study of the effects of such functions in the temporal and spatial complexity of the overall quantum algorithm.

Many research papers have been published in the last years studying a great diversity of methods to perform said calculations, but these are usually designed specifically for some algorithm or to fit a certain architecture.

The goal of this project is to study the state of the art in the implementation of these classical calculations in quantum systems, the compilation and comparison of the dozens of published designs, and the modification of these to be used generally in any desired algorithm, giving all necessary information needed to obtain a physical implementation for any of these from scratch. Any reader will be able to obtain a quantum circuit able to perform any classical computation in a quantum computer and will be given many options to implement every operation in order to adapt it to the particular necessities of its system or architecture.

In this document we show many designs able to perform different arithmetic calculations on quantum states in superposition, we give extensive guides detailing the building process of each of them and we analyze the impact they have on the spatial and temporal complexities of the algorithms that use them.

Índex

Índex de figures **3**

Índex de taules **4**

1 Introducció **5**
1.1 Context . . . 5
1.2 Conceptes bàsics . . . 6
1.3 Justificació . . . 8
1.4 Formulació del problema . . . 9
1.5 Actors implicats . . . 10

2 Computació quàntica **11**
2.1 El Qubit . . . 11
2.2 Bases i transformacions . . . 12
2.3 Operadors quàntics d'un qubit . . . 13
2.4 Sistemes de múltiples qubits . . . 15
2.5 Operadors quàntics de múltiples qubits . . . 17
2.6 Model de computador quàntic . . . 19
2.7 Exemples d'ús de computació clàssica en algorismes quàntics . . . 20

3 Operadors quàntics de càlculs clàssics **26**
3.1 Lògica booleana . . . 27
3.2 Suma de naturals i enters . . . 33
3.3 Negació d'enters . . . 62
3.4 Resta de naturals i enters . . . 71
3.5 Producte de naturals i enters . . . 76
3.6 Divisió de naturals i enters . . . 94
3.7 Exponenciació de naturals . . . 99

4 Conclusions **128**
4.1 Lògica booleana . . . 128
4.2 Suma de naturals i enters . . . 128
4.3 Negació d'enters . . . 130
4.4 Resta de naturals i enters . . . 131
4.5 Producte de naturals i enters . . . 131
4.6 Divisió de naturals i enters . . . 132
4.7 Exponenciació de naturals . . . 132

5 Gestió del projecte **134**
5.1 Objectius . . . 134
5.2 Requisits . . . 135
5.3 Obstacles i riscos . . . 135
5.4 Metodologia i rigor . . . 136
5.5 Descripció de les tasques . . . 136
5.6 Taula resum . . . 142
5.7 Diagrama de Gantt . . . 143
5.8 Planificació d'alternatives i gestió d'imprevistos . . . 145
5.9 Gestió econòmica del projecte . . . 145

6 Informe de Sostenibilitat **151**
6.1 Dimensions de la sostenibilitat . . . 151
6.2 Matriu de sostenibilitat . . . 152

7 Referències **153**

A Base matemàtica per a computació quàntica **156**
A.1 Nombres complexos 156
A.2 Àlgebra lineal 157

B Àlgebra booleana i computació clàssica **160**
B.1 Àlgebra booleana 160
B.2 Representació de nombres 160
B.3 Aritmètica sobre bits 163

Índex de figures

1 *Mecanismes d'Anticitera (esquerra) i Analitzador Harmònic (dreta)* 5
2 *Estat d'un únic bit.* 6
3 *Portes clàssiques de múltiples bits.* 7
4 *Portes quàntiques d'un bit* 7
5 *Entrellaçament.* 8
6 *Esfera de bloch* 12
7 *Diagrames quàntics de portes booleanes bàsiques.* 32
8 *Prototips de l'operador ADD.* 33
9 *Implementació bàsica de l'operador ADD per Ripple-Carry.* 36
10 *Implementació Ripple-Carry de l'operador ADD.* 40
11 *Implementació Ripple-Carry modular de l'operador ADD.* 41
12 *Informació de cada node d'un Fenwick Tree* 46
13 *Implementació Carry-Lookahead de l'operador ADD.* 50
14 *Implementació Carry-Lookahead de l'operador ADD modular.* 51
15 *Implementació Carry-Lookahead no modular de l'operador ADD sobre un operand.* . 56
16 *Implementació Carry-Lookahead modular de l'operador ADD sobre un operand.* . . . 57
17 *Transformada de Fourier quàntica.* 59
18 *Implementació mitjançant la QFT de l'operador ADD* 60
19 *Implementació mitjançant la QFT de l'operador ADD no modular.* 60
20 *Prototips de l'operador NEG.* 62
21 *Prototip de l'operador ADD1.* 62
22 *Implementació Ripple-Carry de l'operador ADD1.* 65
23 *Implementació Carry-Lookahead de l'operador ADD1.* 68
24 *Implementació QFT de l'operador ADD1.* 70
25 *Prototips de l'operador SUB.* 71
26 *Implementacions de l'operador SUB mitjançant negació.* 71
27 *Resta modular sobre el segon operand.* 73
28 *Resta modular sobre un registre addicional.* 74
29 *Resta no modular sobre el segon operand.* 74
30 *Resta no modular sobre un registre addicional.* 75
31 *Prototips de l'operador PROD.* 76
32 *Prototip de l'operador ADD condicionat.* 77
33 *Prototip de l'operador ROT-k.* 77
34 *Implementació de l'operador CADD mitjançant anul·lació de l'operand.* 78
35 *Implementació de l'operador CADD mitjançant condicionament de tots els operands.* 79
36 *Implementació de l'operador ROT-1 mitjançant SWAP.* 80
37 *Producte no modular sobre un registre addicional.* 81
38 *Producte no modular d'enters per negació condicionada dels operands.* 82
39 *Producte de naturals basat en Carry-Save.* 92
40 *Prototips de l'operador DIV.* 94
41 *Divisió de naturals basada en l'algorisme tradicional.* 97
42 *Producte de naturals basat en inversió de la divisió.* 98
43 *Prototip de l'operador POW.* 99
44 *Prototip de l'operador* ***CPROD*** 100
45 *Implementació de l'operador* ***CPROD*** *mitjançant anul·lació de l'operand.* 100
46 *Prototip de l'operador POW2.* 101
47 *Modificacions de ADD per a POW2.* 105
48 *Implementació de l'operador POW2.* 105
49 *Implementació bàsica de l'operador POW.* 107
50 *Prototip i implementació de l'operador TRY-SUB-k.* 110
51 *Prototip de l'operador FIB.* 110
52 *Implementacions de l'operador FIB.* 113

53 *Implementació de l'operador POW amb codificació de Fibonacci.* . . . 115
54 *Prototip de l'operador CSPOW2.* . . . 117
55 *Implementació de l'operador CSPOW2.* . . . 119
56 *Prototip de l'operador CSPROD.* . . . 120
57 *Implementació de l'operador CSPROD.* . . . 122
58 *Implementació Carry-Save de l'operador POW.* . . . 126
59 *Diagrama de Gantt* . . . 144
60 *Operacions en àlgebra booleana* . . . 160
61 *Combinació d'operacions lògiques.* . . . 161
62 *Porta XOR* . . . 163
63 *Bloc Half Adder* . . . 164
64 *Bloc Full Adder* . . . 164
65 *Bloc ADD de 4 bits.* . . . 164
66 *Bloc SUB de 4 bits.* . . . 165
67 *Bloc SHL1.* . . . 166
68 *Bloc SHR1.* . . . 166
69 *Bloc SAR1.* . . . 166
70 *Bloc PROD.* . . . 167
71 *Bloc DIV.* . . . 167

Índex de taules

1 *Portes lògiques bàsiques* . . . 27
2 *Nombre de portes de la implementació bàsica de ADD basade en Riple-Carry.* . . . 37
3 *Nombre de portes de la implementació Ripple-Carry de ADD.* . . . 41
4 *Nombre de portes de la implementació Carry-Lookeahead de ADD.* . . . 57
5 *Nombre de portes de la implementació Carry-Lookeahead de ADD.* . . . 61
6 *Nombre de portes de la implementació Ripple-Carry de ADD1.* . . . 66
7 *Nombre de portes de la implementació Carry-Lookahead de ADD1.* . . . 69
8 *Nombre de portes de la implementació QFT de ADD1.* . . . 70
9 *Nombre de portes per a calcular lògica booleana.* . . . 128
10 *Nombre de portes pel sumador Ripple-Carry sense qubits addicionals.* . . . 129
11 *Nombre de portes pel sumador Carry-Lookahead.* . . . 129
12 *Nombre de portes pel sumador basat en la QFT.* . . . 130
13 *Complexitats dels blocs ADD.* . . . 130
14 *Complexitats dels blocs NEG.* . . . 130
15 *Complexitats dels blocs SUB.* . . . 131
16 *Complexitat del condicionament dels blocs ADD.* . . . 131
17 *Complexitats de les implementacions de PROD.* . . . 132
18 *Complexitats de les implementacions de DIV.* . . . 132
19 *Complexitats de les implementacions de POW.* . . . 133
20 *Resum de tasques* . . . 143
21 *Costs del software* . . . 146
22 *Cost RRHH* . . . 147
23 *Imprevistos RRHH* . . . 148
24 *Pressupost de contingència* . . . 149
25 *Pressupost final* . . . 150
26 *Matriu de sostenibilitat* . . . 152

1 Introducció

Aquest treball de fi de grau consisteix en l'estudi i desenvolupament de tècniques per a processar informació clàssica en superposició dins de computadors quàntics. En aquest document s'aporta la base matemàtica que cal pel a descriure el comportament del model quàntic de computació, i es desenvolupa una jerarquia de circuits capaços de realitzar còmputs clàssics en un entorn quàntic.

Aquest document pretén servir com a una guia que permeti qualsevol lector, independentment de la seva experiència amb el camp, obtenir tots els coneixements necessaris sobre l'estat de l'art de la computació quàntica requerits per a implementar circuits capaços d'operar amb informació clàssica de manera eficient.

1.1 Context

Si li preguntes a qualsevol persona què és un computador, molt probablement està imaginant un ordinador de sobretaula o algun altre dispositiu electrònic digital que permet realitzar certs càlculs, però la realitat és que el terme *computador* es refereix, evidentment, a aquell objecte o persona que realitza certs càlculs computacionals. Tot i que el terme es va encunyar en el segle XVII per a descriure operaris que es dedicaven a realitzar els càlculs manualment, actualment fa referència a les màquines computadores, predominantment aquelles que funcionen amb senyals electròniques que codifiquen la informació en **bits**.

Aquest model de computació ha revolucionat el món gràcies a la seva velocitat de càlcul i flexibilitat per a ser aplicat en una gran diversitat de tasques, de manera que és fàcil oblidar que existeixen altres maneres d'usar les mecàniques que regulen el comportament de l'univers per a processar informació. Ja al segle I aC es va construir el mecanisme d'Anticitera, el qual usava una sèrie d'engranatges diferencials per a calcular posicions de cossos celestes, i l'integrador de bola i disc es va usar en el segle XIX per a calcular transformades de Fourier (Figura 1).

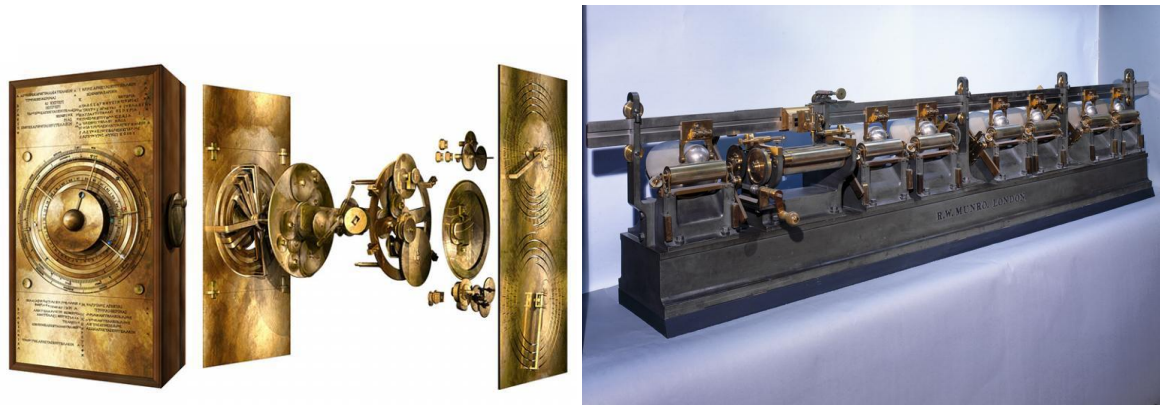


Figura 1: *Mecanismes d'Anticitera(esquerra) i Analitzador Harmònic(dreta).* **Font:** Extretes de https://www.ucl.ac.uk/ [Febrer 2022] i https://collection.sciencemuseumgroup.org.uk [Febrer 2022]

Aquests models alternatius de càlcul han quedat obsolets gràcies als computadors digitals moderns, però són testament que existeixen una gran quantitat de mecàniques de la natura aprofitables pel món de la computació més enllà dels potencials electrònics. El model més recent i amb més promesa de futur és el de computació quàntica, el qual es basa en els postulats de la *mecànica quàntica* i permet aprofitar les interaccions entre sistemes quàntics per a resoldre problemes intractables per ordinadors clàssics.

L'objectiu d'aquest projecte és l'anàlisi i desenvolupament de tècniques per a realitzar còmputs clàssics com el càlcul d'operacions aritmètiques en ordinadors quàntics mitjançant la recerca de l'estat de l'art del camp i possible disseny de nous circuits. Aquest treball es realitzarà a la Facultat d'Informàtica de Barcelona amb la modalitat A, sota la direcció del professor de l'EEBE, Lluís Ametller Congost.

1.2 Conceptes bàsics

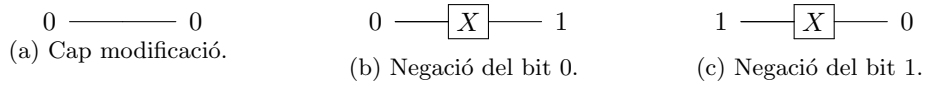
Abans d'entrar més en detall en l'objectiu i la motivació del projecte, cal definir uns quants conceptes bàsics sobre el model de computació quàntica. Aquests seran detallats en més profunditat a la secció d'introducció a la computació quàntica, però considerem imperatiu precedir el document amb cinc cèntims que permetin al lector entendre les diferències bàsiques entre el model clàssic de computació i el quàntic.

Si el lector no és familiar amb el funcionament dels ordinadors moderns recomanem referir-se a l'appèndix B, on es detalla com es representa informació mitjançant *bits* i es donen exemples de possibles circuits que processadors clàssics poden usar per a operar amb aquesta informació codificada en binari.

1.2.1 Computació clàssica: El bit

Tot i que pugui semblar redundant, és útil definir inicialment a grans trets com funciona la computació en ordinadors digitals electrònics, referits equivalentment en la literatura com a computadors clàssics.

La unitat bàsica d'informació clàssica és el *bit*, el qual tan sols pot trobar-se en l'estat 0 o l'estat 1. Aquest estat es pot modificar mitjançant l'aplicació de portes lògiques, que en el cas d'un únic bit tan sols pot ser la porta NOT o equivalentment X, la qual canvia el valor del bit pel seu oposat tal i com es mostra en la figura 2.



(a) Cap modificació. (b) Negació del bit 0. (c) Negació del bit 1.

Figura 2: *Evolució temporal de l'estat d'un bit quan no se li aplica cap porta (a) i quan se li aplica la porta NOT a l'estat inicial 0 (b) o 1 (c).* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Aquests bits es poden agrupar, creant sistemes de múltiples bits capaços de representar informació arbitraria en el seu **estat global**. Aquest estat global del sistema es pot modificar mitjançant portes de múltiples bits, les quals operen sobre un subconjunt dels bits del sistema i en canvien l'estat final en funció dels estats inicials. A la figura 3 es mostra com aquestes portes es poden combinar per a modificar l'estat global del sistema per a sumar els dos primers bits i emmagatzemar el resultat en els dos últims.

Un lector amb una mica d'experiència en el camp veurà que aquestes no són les implementacions típiques de les portes lògiques, però entendre el concepte d'*estat del sistema* és el primer pas per a entendre el model de computació quàntica, ja que en aquest tenim un conjunt finit de "bits quàntics"en un cert estat, i es van aplicant portes quàntiques per a modificar-lo.

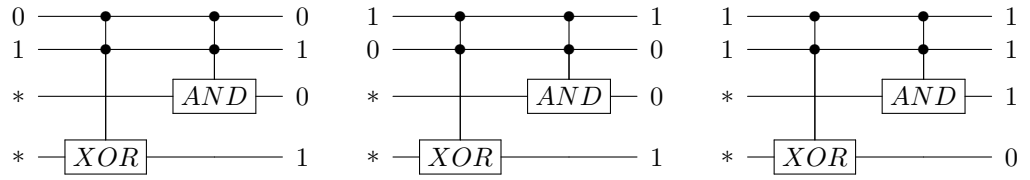


Figura 3: *Half-adder que modifica l'estat global per a emmagatzemar en els dos bits inferiors la suma dels superiors. Els diagrames no usen les convencions típiques i són purament il·lustratius, am únic objectiu la visualització del concepte d'estat del sistema.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

1.2.2 Computació quàntica: El qubit

La unitat d'informació bàsica d'un computador quàntic és el qubit, el qual, com el bit dels ordinadors clàssics, pot estar en l'estat 0 o l'estat 1, però a diferència dels anteriors també es pot trobar en qualsevol superposició d'aquests. Ignorarem els detalls físics de la implementació dels qubits i assumirem l'existència de dos estats quàntics bàsics representats amb $|0\rangle$ i $|1\rangle$. Un estat d'un qubit qualsevol $|\psi\rangle$ es pot expressar com una combinació lineal d'aquests dos:

$$|\psi\rangle = a\,|0\rangle + b\,|1\rangle$$

On a i b són coeficients complexos que no es poden trobar, ja que en mesurar un qubit aquest col·lapsa en un dels dos estats $|0\rangle$ o $|1\rangle$ amb probabilitats $|a|^2$ i $|b|^2$ respectivament.

De la mateixa manera que un qubit pot estar en una superposició qualsevol dels dos estats base, un conjunt de n qubits poden estar en una superposició qualsevol dels 2^n estats base:

$$\begin{aligned}|\psi\rangle &= a_0\,|0\ldots00\rangle + a_1\,|0\ldots01\rangle + a_2\,|0\ldots10\rangle + \cdots + a_{2^n-1}\,|1\ldots11\rangle \\ &= a_0\,|0_2\rangle + a_1\,|1_2\rangle + a_2\,|2_2\rangle + \cdots + a_{2^n-1}\,|(2^n-1)_2\rangle\end{aligned}$$

On $|a_i|^2$ és la probabilitat d'obtenir l'estat $|i_2\rangle$ si es mesuren tots els n qubits.

L'equivalent quàntic de les portes lògiques són les anomenades portes quàntiques o operadors quàntics, les quals actuen sobre un o més qubits i n'alteren els seus estats. L'equivalent quàntic de la porta NOT és l'operador quàntic X, l'efecte del qual sobre diversos qubits inicials es mostra en la figura 4.

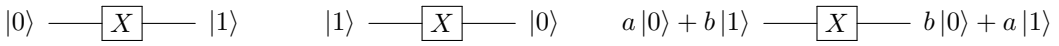


Figura 4: *Efecte de la porta quàntica X sobre els dos estats bàsics i l'estat arbitrari.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Les portes de múltiples qubits són anàlogues a les de bits clàssics, operant sobre un subconjunt dels qubits i modificant-los determinísticament en funció dels seus estats inicials. Cal destacar que aquestes portes poden produir un estat on els qubits individuals no es poden expressar de forma independent, anomenats **estats entrellaçats**. La figura 5 mostra com podem portar un sistema format per dos qubits no entrellaçats $|0\rangle$ i $|0\rangle$ (estat global $|00\rangle$) a l'estat global $\frac{1}{\sqrt{2}} \cdot (|00\rangle + |11\rangle)$, on els estats individuals dels dos qubits no poden ser representats individualment.

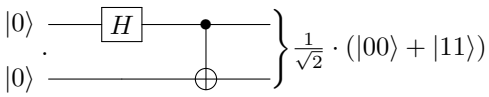


Figura 5: *Obtenció de dos estats entrellaçats mitjançant les portes quàntiques Hadamard i CNOT.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Aquestes són les eines bàsiques de la computació quàntica: un conjunts de qubits en superposició d'un nombre d'estats exponencial en la quantitat de qubits i uns operadors capaços d'actuar sobre tots aquests estats en superposició alhora, possiblement creant entrellaçament entre qubits. Els algorismes quàntics fan ús d'aquestes propietats per a realitzar tot tipus de tasques infactibles en ordinadors clàssics com la factorització d'enters, el càlcul del logaritme discret o la simulació de sistemes quàntics.

El detall més important que cal veure és que un registre de n bits en un ordinador clàssic tan sols pot trobar-se en un estat dels 2^n possibles, de manera que una sola operació sobre aquest registre tan sols el transformarà en un altre estat. En canvi, un registre de n qubits pot trobar-se en **tots** els 2^n estats possibles, de manera que una sola operació transforma tots aquests 2^n estats en uns altres. A aquest fenomen se n'anomena *paral·lelisme quàntic*, ja que permet operar paral·lelament amb un nombre exponencial d'estats, i la seva utilitat hauria de semblar òbvia arribats a aquest punt; si pugéssim aprofitar aquest paral·lelisme, teòricament seríem capaços d'obtenir algorismes amb rendiments exponencialment superiors als clàssics.

1.3 Justificació

L'ús dels anomenats *oracles* o *caixes negres* és una tècnica recurrent en la literatura de computació quàntica que consisteix en assumir l'existència d'un operador quàntic que actua sobre cada estat en superposició, transformant-lo en l'estat resultat: $U_f \left|x\right\rangle_n \left|0\right\rangle_m = \left|x\right\rangle_n \left|f(x)\right\rangle_m$. Aquesta simplificació permet a l'escriptor del document centrar-se en els detalls importants de l'algorisme o bé dissenyar-lo per a usar una funció arbitrària, però sacrifica completesa en ignorar la implementació específica d'aquesta funció, o el seu efecte en la complexitat temporal o espacial de l'algorisme.

Qualsevol intent d'implementar físicament aquests algorismes quàntics, o d'obtenir un disseny complet del seu circuit, requereix de coneixement sobre com implementar càlculs aritmètics en registres de qubits, el qual no és fàcilment accessible ni intuïtiu.

Existeixen multitud de documents que investiguen diverses tècniques per a obtenir circuits capaços de realitzar aquestes operacions. Alguns es basen en modificacions reversibles de la lògica booleana i l'aritmètica usada en processadors ([1], [2], [3], entre altres, detallats a la secció corresponent) mentre que altres aprofiten la transformada de Fourier quàntica per a eliminar la necessitat de qubits auxiliars, entre altres optimitzacions ([4],[5], entre altres, també llistats a la secció corresponent).

A part de la necessitat de donar implementacions per a aquestes caixes negres, cal tenir en compte que aquest model de computació podria revolucionar no tan sols la comunitat científica sinó que també podria esdevenir la nova generació d'ordinadors de sobretaula (o un perifèric quàntic similar a la GPU) si fóssim capaços de dissenyar implementacions físiques de sistemes de qubits adients. Aquest escenari resultaria en la necessitat de dissenyar processadors quàntics de propòsit múltiple, el qual requeriria una *Unitat Aritmètica-Lògica* capaç d'executar operacions aritmètiques bàsiques, i no tan sols funcions específiques.

Aquests fets justifiquen, doncs, la necessitat de realitzar un treball d'investigació sobre el *state of the art* de la implementació d'operacions sobre aquesta informació clàssica superposada dins d'un ordinador quàntic, el qual summaritzi les diverses opcions per a realitzar aquesta tasca i en compari els seus efectes sobre les complexitats espacials i temporals dels algorismes als quals s'apliquen.

1.4 Formulació del problema

Tal i com s'ha mencionat anteriorment, a l'hora de dissenyar algorismes quàntics se sol treballar amb l'assumpció de l'existència d'operadors que efectuen el càlcul de certes funcions numèriques però no se'n donen implementacions específiques. Per exemple, l'algorisme de Shor per a factoritzar enters requereix d'un operador quàntic que actui sobre l'estat de $n+m$ qubits $|x\rangle_n |0\rangle_m$ i el transformi en l'estat $|x\rangle_n |c^x mod N\rangle_m$ per c i N donats i $|x\rangle_n$ qualsevol:

$$U |x\rangle_n |0\rangle_m = |x\rangle_n |c^x mod N\rangle_m$$

$$U \sum_{i=0}^{2^n-1} a_i |i\rangle_n |0\rangle_m = \sum_{i=0}^{2^n-1} a_i |i\rangle_n |c^i mod N\rangle$$

Òbviament és possible implementar aquesta operació amb un operador que actuï sobre el registre de $n+m$ qubits sencer, però el tamany de la matriu de l'operador seria exponencial en $n+m$, de manera que el sol fet de dissenyar un sistema físic capaç d'executar aquesta operació ja és impossible. Cal, per tant, dissenyar estructures de dades i algorismes quàntics que permetin executar operacions aritmètiques bàsiques sobre registres quàntics en superposició d'un nombre exponencialment gran d'estats diferents. Aquests operacions bàsiques que permetrien a un computador quàntic computar funcions numèriques arbitràries serien:

X: Transforma un qubit en el seu oposat.

$$X |x\rangle = |\neg x\rangle$$

OR: OR lògica entre dos qubits.

$$OR |a\rangle |b\rangle |0\rangle = |a\rangle |b\rangle |a \vee b\rangle$$

AND: AND lògica entre dos qubits.

$$OR |a\rangle |b\rangle |0\rangle = |a\rangle |b\rangle |a \wedge b\rangle$$

XOR: XOR lògica entre dos qubits.

$$XOR |a\rangle |b\rangle |0\rangle = |a\rangle |b\rangle |a \oplus b\rangle$$

SUM: Suma de dos enters emmagatzemats en dos registres quàntics de n qubits, ja siguin naturals codificats en base 2 o enters en complement a 2.

$$SUM |a_2\rangle_n |b_2\rangle_n |0_2\rangle_{n+1} = |a_2\rangle_n |b_2\rangle_n |(a+b)_2\rangle_{n+1}$$

NEG: Negació d'un enter codificat en complement a 2 emmagatzemat en un registre de n qubits.

$$NEG |x_2\rangle = |(-x)_2\rangle$$

PROD: Producte de dos enters codificats en base 2 emmagatzemats en dos registres de n i m qubits respectivament.

$$PROD |a_2\rangle_n |b_2\rangle_m |0_2\rangle_{n+m} = |a_2\rangle_n |b_2\rangle_m |(a \cdot b)_2\rangle_{n+m}$$

DIV: Quocient i residu de la divisió de dos enters codificats en base 2 en dos registres de n i m qubits respectivament.

$$DIV\,|a_2\rangle_n\,|b_2\rangle_m\,|0_2\rangle_n\,|0_2\rangle_m = |a_2\rangle_n\,|b_2\rangle_m\,|(a/b)_2\rangle_n\,|(a \mod b)_2\rangle_m$$

EXP: Producte d'un enter emmagatzemat en un registre de n qubits per si mateix tantes vegades com el valor enmmagatzemat en un segon registre de m qubits.

$$EXP\,|a_2\rangle_n\,|b_2\rangle_m\,|0_2\rangle_{n\cdot m} = |a_2\rangle_n\,|b_2\rangle_m\,|(a^b)_2\rangle_{n\cdot m}$$

Si bé és cert que hi ha algorismes per a computar totes les operacions anteriors usant lògica booleana, cal recordar que en aquest model totes les operacions han de ser reversibles ja que totes les matrius unitàries (les quals defineixen els operadors quàntics) tenen inversa, de manera que no podem usar les mateixes implementacions dels algorismes que usen els computadors clàssics.

També cal destacar que, per a realitzar certs càlculs, cal ser capaç d'efectuar operacions condicionades a un cert qubit, el qual pot no ser igual en tots els estats de la superposició i per tant cal dissenyar portes quàntiques que realitzin la seva operació en estats on el qubit de condició és $|1\rangle$ i no modifiqui aquells estats on el qubit sigui $|0\rangle$.

Gran quantitat d'algorismes requereixen també d'informació sobre l'estat per a decidir si finalitzar o no un bloc iteratiu. Com que en un ordinador quàntic treballem amb una superposició d'aquests estats, caldria continuar les iteracions fins que cap estat en requereixi més, el qual implicaria la nostra capacitat d'obtenir informació de tots els estats en superposició i de condicionar l'execució de les operacions d'aquest bloc iteratiu al fet que un cert estat hagi acabat o no. És per això que idealment també caldrà adaptar totes les parts iteratives dels algorismes que estudiarem per a finalitzar en el mateix nombre d'iteracions independentment de l'estat sobre el que actuen.

1.5 Actors implicats

En no ser un treball realitzat per a una empresa o institució externa, els actors que afectaran directament el desenvolupament del projecte són el director, Lluís Ametller Congost, i el desenvolupador, Albert Martín Garcia.

Donat que la legislació vigent garanteix que els treballs de final de grau siguin individuals, serà únicament al desenvolupador sobre qui recaurà la responsabilitat d'investigar i recopil·lar informació sobre l'estat de l'art de la implementació de computació clàssica en computadors quàntics i summaritzar-la en la redacció de l'informe final. Més concretament, la feina del desenvolupador consistirà en realitzar una investigació progressiva sobre les possibles implementacions de les operacions descrites anteriorment sobre estats quàntics en superposició, comparar els diversos dissenys publicats per institucions o individus (o dissenyats per ell mateix en cas que sigui necessari), i comparar els seus requisits i requeriments en el document final, permetent al lector obtenir tot el coneixement requerit per a implementar dites operacions de la manera més eficient.

La figura del director serà clau a l'hora d'enfocar el projecte i de determinar i adreçar errors en la recerca o redacció del document gràcies a la seva experiència en el món científic com a professor de la universitat.

Tot i no formar part del procés de desenvolupament, cal destacar com a beneficiaris del treball tots aquells individus que tractin de desenvolupar o implementar físicament algorismes quàntics i recerquin informació sobre com dissenyar els seus esquemes de portes per a obtenir un sistema quàntic capaç de representar, codificar i operar amb informació clàssica en superposició. Conseqüentment, no s'espera cap benefici econòmic ni per pel desenvolupador ni cap actor o institució implicada en la realització del treball.

2 Computació quàntica

Abans d'entrar en detall en la implementació de càlculs clàssics en entorns quàntics, cal entendre el funcionament bàsic d'aquest model de computació. A continuació es detallen els conceptes bàsics en que es basa la computació quàntica i es descriuen les característiques de la informació quàntica i les diverses maneres de modificar-la o interactuar amb aquesta.

Aquesta secció requereix coneixements matemàtics de nombres complexos i càlcul vectorial. Lectors que no siguin familiars amb aquests temes o que en desitgin un recordatori poden referir-se a l'apèndix A. Tot i no ser vital, es recomana també entendre com es representa informació en bits (o qubits) i els mètodes usats en els computadors clàssics per a transformar aquesta informació, el qual es detalla a l'apèndix B.

2.1 El Qubit

Com s'ha explicat anteriorment, el qubit és la unitat més bàsica d'informació quàntica. El seu funcionament es basa en la capacitat de sistemes quàntics de trobar-se en una superposició d'un conjunt d'estats bàsics. Tot i que no hi ha un límit al nombre d'aquests estats bàsics amb els quals podríem treballar, se sol assumir que s'opera amb tan sols dos d'aquests, anomenats $|0\rangle$ i $|1\rangle$, de manera que un estat arbitrari es pot representar mitjançant una combinació lineal d'aquests:

$$|\psi\rangle = a\,|0\rangle + b\,|1\rangle$$

Al contrari que els bits clàssics, els qubits no es poden mesurar sense pèrdua d'informació, ja que aquests col·lapsen en un dels dos estats bàsics amb probabilitats $|a|^2$ per l'estat $|0\rangle$ i $|b|^2$ per l'estat $|1\rangle$.

Per exemple, en realitzar una mesura sobre l'estat $|\psi\rangle = \frac{1}{2}\,|0\rangle + \frac{\sqrt{3}}{2}\,|1\rangle$, obtindrem $|0\rangle$ amb probabilitat $(\frac{1}{2})^2 = \frac{1}{4}$ o $|1\rangle$ amb probabilitat $(\frac{\sqrt{3}}{2})^2 = \frac{3}{4}$. Tota mesura posterior d'aquest qubit resultaria en el mateix estat, ja que aquest qubit ha col·lapsat en un dels dos estats de la mesura($|0\rangle$ o $|1\rangle$).

Tot i que sembli que calguin quatre nombres reals per a representar un qubit, dos per cada coeficient complex, en realitat tan sols dos en son suficients. Podem escriure els nombres a i b en les seves formes polars, permetent obtenir una expressió equivalent del qubit general:

$$|\psi\rangle = r_0 e^{i\phi_0}\,|0\rangle + r_1 e^{i\phi_1}\,|1\rangle$$
$$|\psi\rangle = e^{i\phi_0}(r_0\,|0\rangle + r_1 e^{i(\phi_1-\phi_0)}\,|1\rangle)$$

La condició de normalitat de les probabilitats de mesura ens permet representar els mòduls dels coeficients (r_0 i r_1) com al sinus i cosinus d'un angle α:

$$|\psi\rangle = e^{i\phi_0}(cos(\alpha)\,|0\rangle + sin(\alpha)e^{i(\phi_1-\phi_0)}\,|1\rangle)$$

La naturalesa dels qubits no ens permet mesurar la fase global $e^{i\phi_0}$, de manera que podem definir un qubit qualsevol en funció d'un angle α que regula els mòduls dels coeficients dels estats bàsics i un segon angle ϕ, el qual representa la fase relativa entre aquests dos coeficients:

$$|\psi\rangle = cos(\alpha)\,|0\rangle + sin(\alpha)e^{i\phi}\,|1\rangle$$

Aquests dos angles permeten construir una bijecció entre estats d'un qubit i punts a la superfície d'una esfera unitària, assignant ϕ com a angle azimutal i 2α com a angle zenital per a garantir que estats oposats es trobin en punts antipodals. L'estat general d'un qubit, doncs, es pot reescriure en termes de la seva posició en aquesta esfera, anomenada **esfera de Bloch**(fig. 6).

Fins ara hem expressat els estats quàntics mitjançant **kets**($|\psi\rangle$), però aquesta notació(anomenada notació **bra-ket**) també defineix els **bras**($\langle\psi|$) de la manera següent:

$$\langle\psi| = |\psi\rangle^\dagger$$

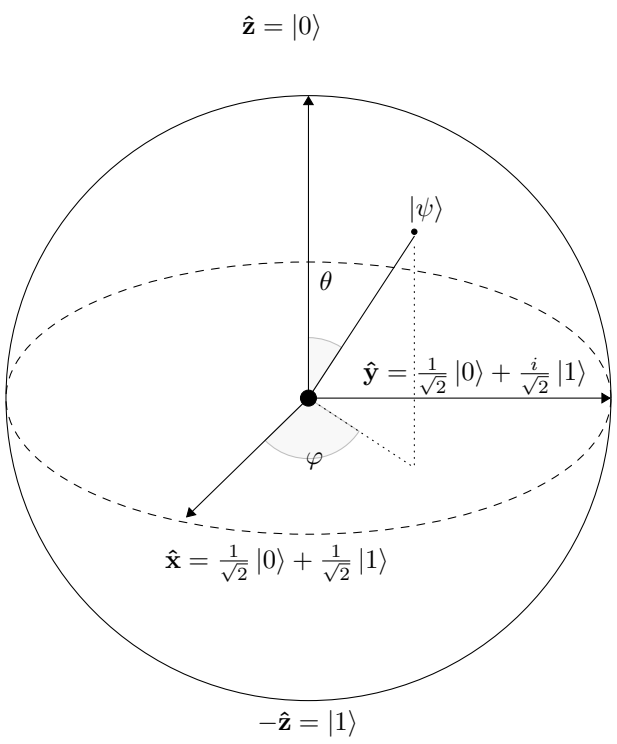


Figura 6: *Representació de diversos estats quàntics a l'esfera de Bloch i visualització de la relació entre un estat arbitrari $|\psi\rangle$ i els angles θ i ϕ.* **Font**: Elaboració pròpia mitjançant Tikz.

El bra de l'estat general d'un qubit és, per tant:

$$\langle\psi| = |\psi\rangle^\dagger = (a\,|0\rangle + b\,|1\rangle)^\dagger = a^*\,|0\rangle^\dagger + b^*\,|1\rangle^\dagger = a^*\,\langle 0| + b^*\,\langle 1|$$

Sobre aquests bras i kets es defineix l'operació de *producte escalar* (o *inner product*), la s'escriu $\langle\psi_1| \cdot |\psi_2\rangle$ o equivalentment $\langle\psi_1|\psi_2\rangle$. Aquesta operació representa la projecció de l'estat ψ_2 sobre ψ_1, de manera que es poden definir els seus resultats per als estats bàsics:

$$\begin{array}{ll}\langle 0|0\rangle = 1 & \langle 0|1\rangle = 0 \\ \langle 1|0\rangle = 0 & \langle 1|1\rangle = 1\end{array}$$

El qual ens permet calcular aquesta projecció entre dos estats arbitraris:

$$\langle\psi_1|\psi_2\rangle = \langle\psi_1|\cdot|\psi_2\rangle = (a\,|0\rangle + b\,|1\rangle)^*\cdot(c\,|0\rangle + d\,|1\rangle) = a^*c\,\langle 0|0\rangle + a^*d\,\langle 0|1\rangle + b^*c\,\langle 1|0\rangle + b^*d\,\langle 1|1\rangle = a^*c + b^*d$$

2.2 Bases i transformacions

Hem vist que els estats $|0\rangle$ i $|1\rangle$ formen una base ja que qualsevol altre estat pot ser escrit com a una combinació lineal d'aquests dos $a\,|0\rangle + b\,|1\rangle$, però aquesta no és pas la única base possible. Per exemple, els estats $|\hat{x}\rangle = \frac{|0\rangle + |1\rangle}{2}$ i $|-\hat{x}\rangle = \frac{|0\rangle - |1\rangle}{2}$ tenen un producte escalar de:

$$\langle\hat{x}| - \hat{x}\rangle = \frac{\langle 0| + \langle 1|}{2} \cdot \frac{|0\rangle - |1\rangle}{2} = \frac{\langle 0|0\rangle - \langle 0|1\rangle + \langle 1|0\rangle - \langle 1|1\rangle}{2} = \frac{1 - 0 + 0 - 1}{2} = 0$$

I els productes escalars amb ells mateixos són 1:

$$\langle\hat{x}|\hat{x}\rangle = \frac{\langle 0| + \langle 1|}{2} \cdot \frac{|0\rangle + |1\rangle}{2} = \frac{\langle 0|0\rangle + \langle 0|1\rangle + \langle 1|0\rangle + \langle 1|1\rangle}{2} = \frac{1+0+0+1}{2} = 1$$
$$\langle -\hat{x}| - \hat{x}\rangle = \frac{\langle 0| - \langle 1|}{2} \cdot \frac{|0\rangle - |1\rangle}{2} = \frac{\langle 0|0\rangle - \langle 0|1\rangle - \langle 1|0\rangle + \langle 1|1\rangle}{2} = \frac{1-0-0+1}{2} = 1$$

Per tant, aquests dos estats permeten expressar qualsevol altre estat $|\psi\rangle$ com a:

$$|\psi\rangle = \langle\hat{x}|\psi\rangle\,|\hat{x}\rangle + \langle -\hat{x}|\psi\rangle\,|-\hat{x}\rangle$$

Visualment aquest canvi de base es pot representar com una rotació de tota l'esfera de Bloch per a alinear la nova base $|\hat{x}\rangle$, $|-\hat{x}\rangle$ sobre $|\hat{z}\rangle$, $|-\hat{z}\rangle$, on $\langle\hat{x}|\psi\rangle$ és la projecció de $|\psi\rangle$ sobre $|\hat{x}\rangle$ i $\langle -\hat{x}|\psi\rangle$ és la projecció de $|\psi\rangle$ sobre $|-\hat{x}\rangle$.

Aquesta habilitat de poder expressar estats com a combinacions lineals d'estats bàsics ens permet representar aquests mitjançant vectors. Per exemple l'estat general $|\psi\rangle = a\,|0\rangle + b\,|1\rangle$ es representaria de manera vectorial com a $\begin{bmatrix} a \\ b \end{bmatrix}$. Cal tenir en compte que tot estat representat d'aquesta manera ha d'anar relacionat a una certa base, la qual serà la base canònica $(|0\rangle, |1\rangle)$ a menys que s'indiqui el contrari.

Podem descompondre aquests coeficients de l'estat en la nova base arbitrària de la manera següent:

$$\begin{aligned}
|\psi\rangle &= \langle\hat{x}|\psi\rangle\,|\hat{x}\rangle + \langle -\hat{x}|\psi\rangle\,|-\hat{x}\rangle \\
&= (\langle 0|\hat{x}\rangle\,|0\rangle + \langle 1|\hat{x}\rangle\,|1\rangle)^* \cdot (a\,|0\rangle + b\,|1\rangle)\,|\hat{x}\rangle + (\langle 0| - \hat{x}\rangle\,|0\rangle + \langle 1| - \hat{x}\rangle\,|1\rangle)^* \cdot (a\,|0\rangle + b\,|1\rangle)\,|-\hat{x}\rangle \\
&= (\langle 0|\hat{x}\rangle^*\,a + \langle 1|\hat{x}\rangle^*\,b)\,|\hat{x}\rangle + (\langle 0| - \hat{x}\rangle^*\,a + \langle 1| - \hat{x}\rangle^*\,b)\,|-\hat{x}\rangle
\end{aligned}$$

El qual permet expressar el canvi de base com a un producte matricial:

$$\begin{bmatrix} a' \\ b' \end{bmatrix} = \begin{bmatrix} \langle 0|\hat{x}\rangle^* & \langle 1|\hat{x}\rangle^* \\ \langle 0| - \hat{x}\rangle^* & \langle 1| - \hat{x}\rangle^* \end{bmatrix} \cdot \begin{bmatrix} a \\ b \end{bmatrix}$$

On a i b són els coeficients de l'estat en la base inicial $(|0\rangle, |1\rangle)$ i a' i b' són els coeficients de l'estat en la base final $(|\hat{x}\rangle, |-\hat{x}\rangle)$. Cal destacar que, tot i que hem expressat la matriu de transformació en funció d'aquestes dues bases, podem usar-la per a obtenir la transformació entre dues bases qualsevol.

Per exemple la matriu de transformació de la base canònica a la base $(|\hat{x}\rangle, |-\hat{x}\rangle)$ és:

$$\begin{bmatrix} \langle 0|\hat{x}\rangle^* & \langle 1|\hat{x}\rangle^* \\ \langle 0| - \hat{x}\rangle^* & \langle 1| - \hat{x}\rangle^* \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Aquest operador pot ser també expressat mitjançant la notació del *producte extern*(o *outer product*):

$$\frac{1}{2}(|0\rangle\,\langle\hat{x}| + |0\rangle\,\langle -\hat{x}| + |1\rangle\,\langle\hat{x}| - |1\rangle\,\langle -\hat{x}|)$$

Realitzar el producte d'un estat en la base canònica amb l'operador anterior és equivalent a multiplicar el vector d'aquest estat amb la forma matricial de l'operador. A vegades serà més convenient definir operadors d'aquesta manera, però cal recordar que aquest es pot expressar mitjançant una matriu i viceversa.

2.3 Operadors quàntics d'un qubit

Un operador quàntic és una funció que transforma un estat quàntic en un altre. Per exemple, el canvi de base és un operador quàntic ja que permet obtenir un nou estat a partir d'un altre, tot i que no és gaire comú ja que el canvi de base sol ser una eina teòrica, rarament usada en algorismes quàntics.

Tres dels operadors quàntics més bàsics són:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Els quals corresponen a rotacions de 180^{o} d'un estat respecte cadascun dels eixos de l'esfera de Bloch. Cal destacar que la porta X és equivalent a una NOT quàntica, en canviar els coeficients dels estats bàsics. Aquestes rotacions es generalitzen a qualsevol angle mitjançant els operadors de rotació:

$$R_x(\theta) = \begin{bmatrix} cos(\frac{\theta}{2}) & -isin(\frac{\theta}{2}) \\ -isin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} cos(\frac{\theta}{2}) & -sin(\frac{\theta}{2}) \\ sin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{bmatrix} \quad R_z(\theta) = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}$$

Altres portes quàntiques de gran importància són:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad T = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix} \quad S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} = T^2$$

Aquestes són les portes més usades però se'n poden dissenyar altres que realitzin transformacions diferents. De fet qualsevol matriu 2x2 defineix una transformació vàlida, sempre que aquesta matriu sigui unitària.

A l'hora de dissenyar processadors d'ordinadors clàssics no podem implementar totes les funcions booleanes usades físicament, sinó que tan sols n'usem un conjunt reduït per a construir la resta. Aquest conjunt sol ser $\{NOT, AND, OR\}$, tot i que els conjunts $\{NOT, AND\}$ o fins i tot $\{NAND\}$ poden ser usats també. Aquests són els anomenats conjunts finits universals, ja que permeten implementar qualsevol porta que actua sobre un conjunt finit de bits mitjançant la composició de portes del conjunt.

En computació quàntica passa el mateix, ja que no és factible implementar físicament les il·limitades portes quàntiques d'un qubit que existeixen. Com s'ha esmentat anteriorment, les portes quàntiques d'un qubit són equivalents a una rotació sobre l'esfera de Bloch, la qual es pot descompondre en tres rotacions independents sobre els tres eixos, de manera que sembla correcte pensar que el conjunt $\{R_x(\theta), R_y(\theta), R_z(\theta)\}$ és universal. Efectivament aquest conjunt d'operadors permet obtenir-ne qualsevol altre mitjançant composició, però aquest no és un conjunt finit ja que les portes depenen d'una variable contínua: l'angle de rotació θ. Existeixen altres conjunts universals infinits de portes quàntiques, els quals permeten la implementació teòrica d'operadors arbitraris però són infactibles a la pràctica en ser impossibles de dissenyar físicament.

La solució radica en obtenir un conjunt capaç de realitzar, no una implementació exacta dels operadors, sinó una aproximació de precisió arbitrària d'aquests. La prova formal de l'existència d'aquest conjunt universal per a portes quàntiques està fora de l'abast d'aquest document, però assumirem que és possible aproximar qualsevol operador quàntic d'un qubit amb un error inferior a ϵ mitjançant $O(log^e(\frac{1}{\epsilon}))$(teorema de Solovay–Kitaev) portes del conjunt$\{H, T\}$. La definició exacta d'aquest error no és important, ja que s'han dissenyat sistemes per a aplicar aquestes portes amb correcció d'errors deguts a la incertesa de l'aproximació, de manera que en aquest document assumirem que tenim la capacitat d'implementar físicament qualsevol operador pel qual puguem determinar la seva matriu o expressió com a producte extern.

Pot semblar que la definició de funcions sobre un conjunt infinit d'estats quàntics és molt més complexa que la definició d'una funció sobre bits, però la realitat és que podem aprofitar la naturalesa matricial dels operadors quàntics per a simplificar el procés. Assumim que U és un operador la matriu del qual desconeixem. Si apliquem aquest operador sobre un estat $|\psi\rangle$ obtenim:

$$U \ |\psi\rangle = U(a\,|0\rangle + b\,|1\rangle) = a\ U\,|0\rangle + b\ U\,|1\rangle$$

De manera que tan sols cal conèixer els efectes de l'operador sobre les bases per a definir la matriu:

$$U = \begin{bmatrix} | & | \\ U\,|0\rangle & U\,|1\rangle \\ | & | \end{bmatrix}$$

El qual serà un operador vàlid si i només si la seva matriu és unitària.

Per exemple, si desitgéssim trobar la matriu corresponent a l'operador X, tan sols caldria saber el seu efecte sobre els estats bàsics $|0\rangle$ i $|1\rangle$:

$$X|0\rangle = |1\rangle$$
$$X|1\rangle = |0\rangle$$

Aquestes expressions es poden escriure de manera equivalent en forma vectorial:

$$X\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix}$$
$$X\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}1\\0\end{bmatrix}$$

Permetent definir la matriu de l'operador X:

$$X = \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$$

La qual té l'efecte esperat sobre l'estat arbitrari:

$$a|0\rangle + b|1\rangle \longrightarrow \boxed{X} \longrightarrow b|0\rangle + a|1\rangle$$

2.4 Sistemes de múltiples qubits

De la mateixa manera que un sol bit clàssic no ens permet realitzar cap càlcul important, cal usar sistemes de múltiples qubits que interactuïn entre ells per a dissenyar qualsevol circuit quàntic d'interès. De la mateixa manera que diem que un qubit es troba en un cert estat, un sistema de n qubits es trobarà en un cert estat $|\psi\rangle$, determinat a partir dels seus qubits $|\psi_i\rangle$ de la manera següent:

$$|\psi\rangle = |\psi_1\rangle\,|\psi_2\rangle \ldots |\psi_n\rangle$$

Assumint que cada $|\psi_i\rangle$ és expressat en la base canònica com a $|\psi_i\rangle = a_i|0\rangle + b_i|1\rangle$, aquest estat del sistema es pot reescriure:

$$\begin{aligned}
|\psi\rangle &= |\psi_0\rangle\,|\psi_1\rangle \ldots |\psi_{n-1}\rangle \\
&= (a_0|0\rangle + b_0|1\rangle) \cdot (a_1|0\rangle + b_1|1\rangle) \ldots (a_{2^n-1}|0\rangle + b_{2^n-1}|1\rangle) \\
&= a_0a_1 \ldots a_{2^n-1}|0\rangle\,|0\rangle \ldots |0\rangle + a_0a_1 \ldots b_{2^n-1}|0\rangle\,|0\rangle \ldots |1\rangle + \cdots + b_0b_1 \ldots b_{2^n-1}|1\rangle\,|1\rangle \ldots |1\rangle \\
&= a_0a_1 \ldots a_{n-1}|00\ldots0\rangle_n + a_0a_1 \ldots b_{2^n-1}|00\ldots1\rangle_n + \cdots + b_0b_1 \ldots b_{n-1}|11\ldots1\rangle_n \\
&= a_0a_1 \ldots a_{n-1}|0_2\rangle_n + a_0a_1 \ldots b_{2^n-1}|1_2\rangle_n + \cdots + b_0b_1 \ldots b_{2^n-1}|(2^n-1)_2\rangle_n
\end{aligned}$$

De manera que podem descriure qualsevol sistema de n qubits com una combinació lineal de les 2^n combinacions possibles de n estats bàsics d'un qubit, el qual ens permet definir la base d'un sistema de n qubits com el conjunt $\{|0_2\rangle_n, |1_2\rangle_n \ldots |(2^n-1)_2\rangle_n\}$. Per exemple, la base d'un sistema de tres qubits seria el conjunt:

$$\{|0_2\rangle_3, |1_2\rangle_3, |2_2\rangle_3, |3_2\rangle_3, |4_2\rangle_3, |5_2\rangle_3, |6_2\rangle_3, |7_2\rangle_3\}$$

O equivalentment:

$$\{|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle\}$$

Aquí és quan un podria començar a entendre el benefici d'aquest model de computació. Un sistema clàssic de n bits pos estar en qualsevol dels 2^n estats possibles, però un sistema quàntic de n qubits pot estar en qualsevol combinació d'aquests 2^n estats bàsics. Tal i com s'ha esmentat a la introducció, operar amb aquesta superposició és equivalent a operar amb cadascun dels estats individuals, el qual ens permet treballar amb un nombre exponencialment més gran d'informació que amb un ordinador clàssic.

De la mateixa manera que podem representar un qubit $|\psi\rangle = a\,|0\rangle + b\,|1\rangle$ en la seva forma vectorial $\begin{bmatrix} a \\ b \end{bmatrix}$, un sistema de n qubits es pot escriure mitjançant un vector format pels coeficients dels estats bàsics del sistema:

$$|\psi\rangle = a_0\,|0_2\rangle_n + a_1\,|1_2\rangle_n + \cdots + a_{2^n-1}\,|(2^n-1)_2\rangle_n$$

$$|\psi\rangle = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{2^n-1} \end{bmatrix}$$

Els quals es poden trobar a partir dels vectors que representen els estats individuals mitjançant el producte tensorial de Kronecker:

$$\psi = \begin{bmatrix} a_0 \\ b_0 \end{bmatrix} \otimes \begin{bmatrix} a_1 \\ b_1 \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} a_{2^n-1} \\ b_{2^n-1} \end{bmatrix} \otimes = \begin{bmatrix} a_0 a_1 \ldots a_{2^n-1} \\ a_0 a_1 \ldots b_{2^n-1} \\ \vdots \\ b_0 b_1 \ldots b_{2^n-1} \end{bmatrix}$$

El significat d'aquests coeficients és anàleg al cas d'un sol qubit, és a dir, $|a_i|^2$ és la probabilitat d'obtenir l'estat $|i\rangle_n$ si es mesuren **tots** els qubits en la base canònica, o equivalentment, es mesura el sistema en la base canònica. Cal recordar, però, que aquest estat és una combinació dels estats individuals dels qubits, de manera que encara podem realitzar mesures individuals sense col·lapsar els estats dels qubits no mesurats.

Per exemple, si tenim un sistema de dos qubits en l'estat:

$$|\psi\rangle = \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$$

I en mesurem el primer, obtindrem que aquest es troba en l'estat $|0\rangle$ o l'estat $|1\rangle$ amb probabilitats:

$$\begin{aligned} p(\psi_0 = |0\rangle) &= p(\psi_0 = |0\rangle\,|\psi = |00\rangle) + p(\psi_0 = |0\rangle\,|\psi = |01\rangle) + p(\psi_0 = |0\rangle\,|\psi = |10\rangle) + p(\psi_0 = |0\rangle\,|\psi = |11\rangle) \\ &= (\frac{1}{2})^2 + (\frac{-1}{2})^2 + 0^0 + 0^2 \\ &= \frac{1}{2} \end{aligned}$$

$$\begin{aligned} p(\psi_0 = |1\rangle) &= p(\psi_0 = |1\rangle\,|\psi = |00\rangle) + p(\psi_0 = |1\rangle\,|\psi = |01\rangle) + p(\psi_0 = |1\rangle\,|\psi = |10\rangle) + p(\psi_0 = |1\rangle\,|\psi = |11\rangle) \\ &= 0^0 + 0^2 + (\frac{1}{2})^2 + (\frac{-1}{2})^2 \\ &= \frac{1}{2} \end{aligned}$$

En el cas que la mesura resulti $|0\rangle$ el nou estat del sistema serà:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |01\rangle)$$

I en cas que el primer qubit col·lapsés a $|1\rangle$, el nou estat del sistema seria:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|10\rangle - |11\rangle)$$

En els dos casos la mesura del primer qubit no ha afectat l'estat del segon, ja que si l'haguéssim mesurat a l'estat inicial hauríem obtingut $|0\rangle$ o $|1\rangle$ amb probabilitat:

$$\begin{aligned} p(\psi_1 = |0\rangle) &= p(\psi_1 = |0\rangle\,|\psi = |00\rangle) + p(\psi_1 = |0\rangle\,|\psi = |01\rangle) + p(\psi_1 = |0\rangle\,|\psi = |10\rangle) + p(\psi_1 = |0\rangle\,|\psi = |11\rangle) \\ &= (\frac{1}{2})^2 + (\frac{-1}{2})^2 + 0^0 + 0^2 \\ &= \frac{1}{2} \end{aligned}$$

$$\begin{aligned} p(\psi_1 = |1\rangle) &= p(\psi_1 = |1\rangle\,|\psi = |00\rangle) + p(\psi_1 = |1\rangle\,|\psi = |01\rangle) + p(\psi_1 = |1\rangle\,|\psi = |10\rangle) + p(\psi_1 = |1\rangle\,|\psi = |11\rangle) \\ &= 0^0 + 0^2 + (\frac{1}{2})^2 + (\frac{-1}{2})^2 \\ &= \frac{1}{2} \end{aligned}$$

Les quals son iguals a les probabilitats de mesura del segon qubit en els estats posteriors a la mesura del primer, independentment del resultat. Això és conseqüència del fet que l'estat inicial ψ és factoritzable en els dos estats dels qubits que formen el sistema ψ_0 i ψ_1:

$$\begin{gathered} \psi = \psi_0 \cdot \psi_1 \\ \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \cdot \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{gathered}$$

Tot i que aquesta separabilitat d'estats sembli òbvia, considerant que cada qubit és un sistema quàntic ïndependent", no és pas el cas per tots els estats de múltiples qubits. Per exemple, si intentem factoritzar un sistema de dos qubits en l'estat:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = (a_0\,|0\rangle + b_0\,|1\rangle)(a_1\,|0\rangle + b_1\,|1\rangle)$$

$$\left.\begin{array}{l} a_0 a_1 = 1 \\ a_0 b_1 = 0 \\ b_0 a_1 = 0 \\ b_0 b_1 = 1 \end{array}\right\} \implies \begin{array}{l} \frac{b_1}{a_1} = 0 \\ \frac{a_1}{b_1} = 0 \end{array} \implies a_0 a_1 \neq 1 \wedge b_0 b_1 \neq 1$$

Aquest fenomen s'anomena **entrellaçament**, i és de gran importància en computació quàntica, permetent per exemple conèixer l'estat d'un qubit amb la mesura del seu parell entrellaçat:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \begin{cases} |\psi_0\rangle = |0\rangle \implies |\psi_1\rangle = |0\rangle \\ |\psi_0\rangle = |1\rangle \implies |\psi_1\rangle = |1\rangle \end{cases}$$

2.5 Operadors quàntics de múltiples qubits

Gràcies a la definició anterior de base d'un sistema de múltiples qubits, podem definir operadors que actuïn sobre aquest sistema de manera anàloga als operadors d'un qubit. Assumim que U és un operador que actua sobre un sistema de n qubits en l'estat $|\psi\rangle_n$:

$$\begin{aligned} U\left|\psi\right\rangle_n &= U(a_0\left|0_2\right\rangle_n + a_1\left|1_2\right\rangle_n + a_2\left|2_2\right\rangle_n + \cdots + a_{2^n-1}\left|(2^n-1)_2\right\rangle_n) \\ &= a_0U\left|0_2\right\rangle_n + a_1U\left|1_2\right\rangle_n + a_2U\left|2_2\right\rangle_n + \cdots + a_{2^n-1}U\left|(2^n-1)_2\right\rangle_n \\ &= \begin{bmatrix} | & | & \cdots & | \\ U\left|0_2\right\rangle_n & U\left|1_2\right\rangle_n & \cdots & U\left|(2^n-1)_2\right\rangle_n \\ | & | & \cdots & | \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{2^n-1} \end{bmatrix} \end{aligned}$$

És a dir, tan sols cal conèixer l'efecte de l'operador sobre els estats bàsics del sistema per a poder definir-lo completament.

L'operador quàntic de dos qubits més important és la porta CNOT, o *conditional NOT*, la qual aplica una porta NOT a un qubit condicionada a l'estat d'un altre. Més concretament, l'efecte d'aquesta porta sobre els quatre estats bàsics del sistema de dos qubits sobre el qual s'aplica és:

$$\begin{aligned} CNOT\left|00\right\rangle &= \left|00\right\rangle \\ CNOT\left|01\right\rangle &= \left|01\right\rangle \\ CNOT\left|10\right\rangle &= \left|11\right\rangle \\ CNOT\left|11\right\rangle &= \left|10\right\rangle \end{aligned}$$

Com abans, podem reescriure aquestes expressions en forma vectorial:

$$CNOT\begin{bmatrix}1\\0\\0\\0\end{bmatrix} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix} \quad CNOT\begin{bmatrix}0\\1\\0\\0\end{bmatrix} = \begin{bmatrix}0\\1\\0\\0\end{bmatrix} \quad CNOT\begin{bmatrix}0\\0\\1\\0\end{bmatrix} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix} \quad CNOT\begin{bmatrix}0\\0\\0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$$

El qual ens permet definir la matriu de l'operador CNOT:

$$CNOT = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$$

En general, qualsevol operador que actuï sobre n qubits es pot descriure mitjançant una matriu de complexos de 2^n files per 2^n columnes.

Altres operadors d'interès són la porta **Toffoli**, la qual és equivalent a una **CNOT** condicionada a un qubit addicional:

$$Toffoli\left|a\right\rangle\left|b\right\rangle\left|c\right\rangle = \left|a\right\rangle\left|b\right\rangle\left|a \wedge b \oplus c\right\rangle$$

I la porta **SWAP**, la qual intercanvia els valors de dos qubits:

$$SWAP\left|a\right\rangle\left|b\right\rangle = \left|b\right\rangle\left|a\right\rangle$$

Solem escriure estats de sistemes quàntics en la seva forma més expandida, ja que com s'ha mencionat abans no sempre es poden factoritzar sistemes en estats completament independents. És per això que és important determinar l'efecte d'aplicar un operador a un subconjunt dels qubits a l'estat general del sistema. De la mateixa manera que podem obtenir el vector d'estat del sistema a partir dels vectors individuals dels estats, podem obtenir la matriu que determina el canvi en el sistema en funció de les matrius aplicades a cada subconjunt dels qubits.

Per exemple, si tenim un sistema de 4 qubits i apliquem la porta H al primer, cap porta al segon i la porta CNOT al terçer i quart, la matriu que descriu l'efecte de totes aquestes operacions a l'estat del sistema s'obté:

$$U = H \otimes I \otimes CNOT$$

$$U = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$U = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix}$$

El resultat anterior serveix també per a demostrar la necessitat de trobar maneres d'efectuar les operacions mitjançant portes que actuïn sobre un conjunt reduït de qubits, ja que la complexitat de les matrius corresponents és exponencial respecte el nombre de qubits del sistema sobre els que actuen.

Qualsevol matriu que actuï sobre un espai de d dimensions pot ser descomposta en $O(d^2)$ matrius que actuïn de manera no trivial sobre dues de les dimensions, o equivalentment, qualsevol operador que actuï sobre un estat de n qubits pot ser decompost en 2^{2n} operadors que actuïn sobre 2 qubits. I qualsevol operador que actuï sobre dos dels n qubits pot ser implementat amb $O(n^2)$ portes d'un qubit i operadors CNOT, de manera que una màquina capaç de realitzar les operacions del conjunt $\{H, T, CNOT\}$ pot aplicar qualsevol operador[1].

Les proves formals d'aquests postulats són fora de l'àmbit del treball, i no són gaire importants ja que sovint treballarem amb el conjunt no universal $\{\mathbf{X}, \mathbf{CNOT}, \mathbf{Toffoli}\}$. Tot i no poder implementar qualsevol operador quàntic, aquest conjunt és universal per a computació clàssica reversible, de manera que sovint tractarem de determinar circuits per a les operacions desitjades basats en aquest conjunt d'operadors.

2.6 Model de computador quàntic

Abans de començar a discutir algorismes i circuits per a implementar les operacions desitjades, cal definir exactament quines propietats i capacitats té el nostre model teòric de computador quàntic:

[1] Aquest conjunt $\{H, T\}$ permet aproximar qualsevol porta d'un qubit amb precisió arbitrària. De nou, la justificació d'aquest fet és fora de l'abast d'aquest treball, però ens permet assumir que podem aplicar qualsevol porta d'un qubit la matriu de la qual puguem definir.

- **Recursos quàntics**: El model conté un sistema amb un nombre suficient de qubits, on les bases de cada qubit i del sistema són les canòniques, $(|0\rangle, |1\rangle)$ i $(|0_2\rangle_n, |1_2\rangle_n \dots |(2^n-1)_2\rangle_n)$ respectivament.
- **Aplicació d'operadors arbitraris**: El model és capaç d'aplicar totes les operacions d'un conjunt universal sobre qualsevol subconjunt dels seus qubits. Això garanteix la capacitat de poder implementar qualsevol operador quàntic, tot i que la complexitat i el nombre de portes dependrà del conjunt universal en qüestió.
- **Preparació d'estats arbitraris**: El model és capaç de preparar tot estat $|\psi_i\rangle$ de qualsevol qubit individual del sistema. Aquesta propietat és conseqüència de l'anterior, en poder implementar operadors que transformin l'estat inicial $|0\rangle$ en qualsevol altre $|\psi\rangle$, però simplifica les explicacions permetent l' omissió d'aquest pas inicial.
- **Mesura de qubits**: El model permet mesurar qualsevol dels qubits en la base canònica $(|0\rangle, |1\rangle)$.
- **Recursos clàssics**: El model consta també d'una part clàssica, capaç d'executar qualsevol algorisme clàssic i de control·lar l'aplicació de les portes quàntiques al sistema. Aquesta part clàssica pot també accedir als valors de les mesures dels qubits en la base canònica, les quals s'assignen als valors 0 i 1.

2.7 Exemples d'ús de computació clàssica en algorismes quàntics

Com s'ha mencionat anteriorment, l'importància de la recerca en el camp de la computació quàntica implica que hi ha una bona probabilitat que eventualment siguem capaços de crear computadors quàntics amb aplicacions més enllà de la recerca, tal com ha passat amb gran quantitat de tecnologies anteriorment, fent necessària aquesta recerca.

En el cas que el lector desitgi conèixer les aplicacions actuals d'aquest projecte, presentem en aquesta secció dos algorismes quàntics capaços de resoldre problemes típics, alguns intractables en ordinadors clàssics. La lectura d'aquesta secció és recomanada per a entendre la importància de la nostra recerca, però és completament optativa.

2.7.1 Algorisme de Deutsch-Jozsa generalitzat

Assumim que tenim el següent problema:

Donada una funció desconeguda de la forma:

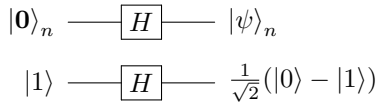
$$f : \{0,1\}^n \to \{0,1\}$$

tal que és constant (val 0 o 1 per totes les entrades) o bé balancejada (val 0 per la meitat d'entrades i 1 per l'altra meitat), quantes crides calen per discernir si la funció és constant o bé balancejada?

En un ordinador clàssic hauríem de calcular la funció per a més de la meitat de les 2^n entrades possibles, ja que si més de la meitat dels valors triats valen el mateix sabem que és constant, el qual implica una complexitat temporal exponencial en n de $O(2^n C)$, on C és la complexitat de la funció.

L'algorisme de Deutsch-Jozsa generalitzat usa un ordinador quàntic de $n+1$ qubits per a trobar la solució a aquest problema en $O(C)$.

Partint de l'estat $|\mathbf{0}\rangle_n |1\rangle$, s'aplica una $\mathbf{H}$ a cada qubit :

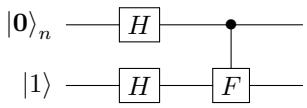


On ψ és igual a:

$$\begin{aligned}\bigotimes_{i=0}^{n-1} H\,|0\rangle &= \bigotimes_{i=0}^{n-1} \frac{|0\rangle + |1\rangle}{\sqrt{2}} \\ &= \frac{1}{\sqrt{2^n}} \bigotimes_{i=0}^{n-1} |0\rangle + |1\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |\mathbf{i}\rangle_n\end{aligned}$$

És a dir, l'estat és igual a la superposició de tots els estats possibles. Aquest és un pas comú en molts algorismes quàntics ja que se sol voler treballar amb tots aquests estats possibles per a aprofitar el paral·lelisme quàntic.

Seguidament s'aplica la funció desitjada per a transformar cada estat $|\mathbf{x}\rangle_n |y\rangle$ a $|\mathbf{x}\rangle_n |f(x) \oplus y\rangle$:



L'estat ara és, doncs:

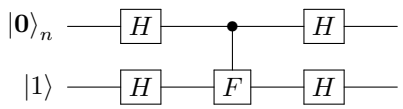
$$F \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |\mathbf{i}\rangle_n \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

$$\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} F\,|\mathbf{i}\rangle_n \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

$$\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |\mathbf{i}\rangle_n \frac{1}{\sqrt{2}}(|f(i)\rangle - |\neg f(i)\rangle)$$

$$\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |\mathbf{i}\rangle_n \frac{(-1)^{f(i)}}{\sqrt{2}}(|0\rangle - |1\rangle)$$

L'últim pas de l'algorisme és l'aplicació d'una altra porta **H** als qubits d'entrada, tal i com s'ha fet al primer pas:



Un podria pensar que no té sentit modificar els qubits d'entrada un altre cop, ja que no han estat modificats entre mig, però cal recordar que l'aplicació de qualsevol operador a un subconjunt dels qubits es pot veure com una matriu que els afecta a tots. L'estat final és:

$$\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} (-1)^{f(i)} H\,|\mathbf{i}\rangle_n H \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

$$\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} (-1)^{f(i)} \bigotimes_{j=0}^{n-1} (H \left|i_j\right\rangle) \left|1\right\rangle$$

$$\frac{1}{2^n} \sum_{i=0}^{2^n-1} (-1)^{f(i)} \bigotimes_{j=0}^{n-1} (\left|0\right\rangle + (-1)^{i_j} \left|1\right\rangle) \left|1\right\rangle$$

$$\frac{1}{2^n} \sum_{i=0}^{2^n-1} (-1)^{f(i)} \sum_{j=0}^{2^n-1} (-1)^{\bigoplus_{k=0}^{n-1} i_k j_k} \left|\mathbf{j}\right\rangle_n \left|1\right\rangle$$

$$\frac{1}{2^n} \sum_{i=0}^{2^n-1} \sum_{j=0}^{2^n-1} (-1)^{f(i) \bigoplus_{k=0}^{n-1} i_k j_k} \left|\mathbf{j}\right\rangle_n \left|1\right\rangle$$

Podem aïllar el coeficient de cada estat $\left|\mathbf{j}\right\rangle_n \left|1\right\rangle$:

$$\frac{1}{2^n} \sum_{i=0}^{2^n-1} (-1)^{f(i) \bigoplus_{k=0}^{n-1} i_k j_k}$$

Podem, doncs, calcular el coeficient de l'estat $\left|\mathbf{0}\right\rangle_n \left|1\right\rangle$:

$$\frac{1}{2^n} \sum_{i=0}^{2^n-1} (-1)^{f(i)}$$

El qual té valor absolut igual a 1 quan la funció f és constant. Com que sabem que els coeficients estan normalitzats, si la funció és constant i mesurem el sistema, obtindrem sempre $\left|\mathbf{0}\right\rangle_n \left|1\right\rangle$, i si és balancejada, la suma és 0, de manera que mai es mesurarà $\left|\mathbf{0}\right\rangle_n \left|1\right\rangle$. Hem trobat, per tant, si la funció original és constant o balancejada.

Com es pot veure, l'única part que no hem mostrat, ni se sol fer en la literatura, és com realitzar la transformació de $\left|\mathbf{x}\right\rangle_n \left|y\right\rangle$ a $\left|\mathbf{x}\right\rangle_n \left|f(x) \oplus y\right\rangle$, el qual és en el que se centra la part principal del document.

2.7.2 Algorisme de Grover

L'algorisme de Groves és un dels més versàtils coneguts, capaç d'invertir funcions, cercar elements d'un conjunt que satisfacin una certa condició i fins i tot resoldre problemes NP-HARD, tot i que aquest no ofereix un increment exponencial de rendiment.

Assumim que tenim el problema:

Donada una funció:

$$f : \{0,1\}^n \to \{0,1\}$$

Volem trobar un $x \in \{0,1\}^n$ tal que $f(x) = 1$.

De nou, en un ordinador clàssic hauríem de calcular la funció per totes les 2^n entrades possibles per a veure quins valors donen 1, el qual implica una complexitat temporal exponencial en n de $O(2^n C)$, in C és la complexitat de la funció.

L'algorisme de Grover permet obtenir aquest resultat en $O(\sqrt{2^n} C)$.

Com abans, tenim un sistema de $n+1$ qubits inicialment en l'estat $|\mathbf{0}\rangle_n |1\rangle$ i apliquem una **H** per a obtenir una superposició de tots els estats possibles, anomenarem a l'estat dels primers n qubits $|\psi\rangle_n$:

$$|\mathbf{0}\rangle_n \text{ ——}\boxed{H}\text{—— } |\psi\rangle_n$$
$$|1\rangle \text{ ——}\boxed{H}\text{—— } \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |\mathbf{i}\rangle_n$$

Assumim que i_0 és el valor de i que busquem, el qual compleix $f(i_0) = 1$. L'estat serà igual a:

$$\frac{1}{\sqrt{2^n}}(|i_0\rangle - |i_0\rangle + \sum_{i=0}^{2^n-1} |\mathbf{i}\rangle_n)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

$$\frac{1}{\sqrt{2^n}}(|i_0\rangle - |i_0\rangle + \sqrt{2^n}\,|\psi\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

$$(\frac{1}{\sqrt{2^n}}\,|i_0\rangle + \frac{1}{\sqrt{2^n}}(\sqrt{2^n}\,|\psi\rangle - |i_0\rangle))\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

Veiem que l'estat dels primers n qubits del sistema es pot expressar en termes dels vectors $\psi - |i_0\rangle$ i $|i_0\rangle$, els quals són òbviament ortogonals ja que el seu producte escalar és de 0. De fet, els podriem usar com a base per a expressar aquest mateix estat dels primers n qubits com a:

$$(\frac{1}{\sqrt{2^n}}, \frac{1}{\sqrt{2^n}})$$

Ens interessa, però, poder expressar aquest estat com a un vector normalitzat, de manera que reescriurem el coeficient del segon vector bàsic com a:

$$(\frac{1}{\sqrt{2^n}}\,|i_0\rangle + \frac{1}{\sqrt{2^n}}\frac{\sqrt{2^n-1}}{\sqrt{2^n-1}}(\sqrt{2^n}\,|\psi\rangle - |i_0\rangle))\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

$$(\frac{1}{\sqrt{2^n}}\,|i_0\rangle + \frac{\sqrt{2^n-1}}{\sqrt{2^n}}\frac{\sqrt{2^n}\,|\psi\rangle - |i_0\rangle}{\sqrt{2^n-1}})\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

De manera que podem representar aquest estat mitjançant la base formada pels vectors $|\phi\rangle = \frac{\sqrt{2^n}\psi - |i_0\rangle}{\sqrt{2^n-1}}$ i $|i_0\rangle$, el qual és el vector unitari:

$$(\frac{\sqrt{2^n-1}}{\sqrt{2^n}}, \frac{1}{\sqrt{2^n}})$$

Aplicar l'operador F que calcula la funció sobre el qubit auxiliar té el següent efecte sobre aquest estat:

$$F(\frac{1}{\sqrt{2^n}}\,|i_0\rangle + \frac{\sqrt{2^n-1}}{\sqrt{2^n}}\,|\phi\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

$$F(\frac{1}{\sqrt{2^n}}\,|i_0\rangle + \frac{\sqrt{2^n-1}}{\sqrt{2^n}}\,|\phi\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

$$\frac{1}{\sqrt{2^n}} F\left|i_0\right\rangle \frac{1}{\sqrt{2}}(\left|0\right\rangle - \left|1\right\rangle) + \frac{\sqrt{2^n-1}}{\sqrt{2^n}} F\left|\phi\right\rangle \frac{1}{\sqrt{2}}(\left|0\right\rangle - \left|1\right\rangle)$$

$$\frac{1}{\sqrt{2^n}} - \left|i_0\right\rangle \frac{1}{\sqrt{2}}(\left|0\right\rangle - \left|1\right\rangle) + \frac{\sqrt{2^n-1}}{\sqrt{2^n}} \left|\phi\right\rangle \frac{1}{\sqrt{2}}(\left|0\right\rangle - \left|1\right\rangle)$$

$$(\frac{1}{\sqrt{2^n}} - \left|i_0\right\rangle + \frac{\sqrt{2^n-1}}{\sqrt{2^n}} \left|\phi\right\rangle)\frac{1}{\sqrt{2}}(\left|0\right\rangle - \left|1\right\rangle)$$

Ja que la funció val 1 per l'estat i_0 i val 0 per tots els estats que formen l'altre vector bàsic.

El qubit del resultat no és modifica, de manera que l'efecte de l'operador **F** sobre els vectors bàsics de l'estat dels primers n qubits és:

$$F\left|\phi\right\rangle = \left|\phi\right\rangle$$
$$F\left|i_0\right\rangle = -\left|i_0\right\rangle$$

O en forma de matriu en aquesta base:

$$F = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Sobre aquesta base també es defineix un altre operador, anomenat Inversion Around Mean, o **IAM**, definit en notació de producte extern com $IAM = 2\left|\psi\right\rangle\left\langle\psi\right| - 1$. Podem calcular l'efecte en la base $\left|i_0\right\rangle$, $\left|\phi\right\rangle$:

$$IAM\left|\phi\right\rangle = 2(\frac{\sqrt{2^n-1}}{\sqrt{n}}\left|\phi\right\rangle + \frac{1}{\sqrt{n}}\left|i_0\right\rangle)(\frac{\sqrt{2^n-1}}{\sqrt{n}}\left\langle\phi\right| + \frac{1}{\sqrt{n}}\left\langle i_0\right|)\left|\phi\right\rangle - \left|\phi\right\rangle = (2\frac{2^n-1}{2^n} - 1)\left|\phi\right\rangle + 2\frac{\sqrt{2^n-1}}{2^n}\left|i_0\right\rangle$$
$$IAM\left|i_0\right\rangle = 2(\frac{\sqrt{2^n-1}}{\sqrt{n}}\left|\phi\right\rangle + \frac{1}{\sqrt{n}}\left|i_0\right\rangle)(\frac{\sqrt{2^n-1}}{\sqrt{n}}\left\langle\phi\right| + \frac{1}{\sqrt{n}}\left\langle i_0\right|)\left|i_0\right\rangle - \left|i_0\right\rangle = 2\frac{\sqrt{2^n-1}}{2^n}\left|\phi\right\rangle + (2\frac{1}{2^n} - 1)\left|i_0\right\rangle$$

O en forma de matriu:

$$IAM = \begin{bmatrix} 1-\frac{2}{2^n} & 2\frac{\sqrt{2^n-1}}{2^n} \\ 2\frac{\sqrt{2^n-1}}{2^n} & -1+\frac{2}{2^n} \end{bmatrix}$$

Si apliquem aquests dos operadors seguits al sistema format pels primers n qubits, obtenim la transformació:

$$G = \begin{bmatrix} 1-\frac{2}{2^n} & 2\frac{\sqrt{2^n-1}}{2^n} \\ 2\frac{\sqrt{2^n-1}}{2^n} & -1+\frac{2}{2^n} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 1-\frac{2}{2^n} & -2\frac{\sqrt{2^n-1}}{2^n} \\ 2\frac{\sqrt{2^n-1}}{2^n} & 1-\frac{2}{2^n} \end{bmatrix}$$

El qual és equivalent a una matriu de rotació en dues dimensions:

$$IAM = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}$$

On θ és un valor fixe que depèn de n.

Just després d'aplicar les **H**, podíem representar l'estat en la nostra base com a:

$$(\frac{\sqrt{2^n-1}}{\sqrt{2^n}}, \frac{1}{\sqrt{2^n}})$$

El qual forma un angle amb l'eix del primer vector bàsic, el cosinus del qual és:

$$\frac{\sqrt{2^n-1}}{\sqrt{2^n}} = \sqrt{\frac{2\cdot 2^n-2}{2\cdot 2^n}} = \sqrt{\frac{1+\frac{2^n-2}{2^n}}{2}} = \sqrt{\frac{1+cos(\theta)}{2}} = cos(\frac{\theta}{2})$$

És a dir, comencem formant un angle de $\frac{\theta}{2}$, i cada aplicació de **F** i **IAM** l'augmenta en θ. La probabilitat de mesurar els n qubits i trobar $|i_0\rangle$ després d'aplicar aquest parell de portes k vegades és:

$$sin^2(\frac{\theta}{2}+k\theta)$$

Volem maximitzar aquesta probabilitat, de manera que podem calcular el nombre òptim d'iteracions:

$$sin^2(\frac{\theta}{2}+k\theta) = 1 \implies k \approx \frac{\pi-\theta}{2\theta}$$

La probabilitat de mesurar el valor desitjat tendeix a 1 a mesura que n augmenta.

Com abans, l'algorisme tan sols requereix informació sobre com implementar la funció desitjada, el qual es mostrarà a la secció principal del treball. Aquesta funció, per exemple, pot calcular si una assignació de variables és una solució d'un problema SAT, si una permutació de nodes té cost menor a un cert valor pel Traveling Salesman Problem, o simplement si el resultat és un cert valor. Malauradament, requereix de $\sqrt{2^n}$ iteracions per a convergir, el qual no és un increment exponencial de rendiment, però substancial de totes maneres.

3 Operadors quàntics de càlculs clàssics

Un cop finalitzada la base teòrica, podem començar a detallar les diferents operacions a implementar en ordinadors quàntics i comparar els diversos circuits o dissenys que permeten efectuar aquests còmputs. Aquesta secció tracta de servir com a guia per a implementar qualsevol càlcul clàssic en un ordinador quàntic de la manera més eficient possible. Hi ha diversos estudis realitzats sobre temes relacionats, generalment en el marc dels algorismes de Shor de factorització i logaritme discret que van propulsar la recerca en aquest camp, amb diverses complexitats tant temporals com espacials ja que, com en la computació clàssica, la majoria d'operacions es poden accelerar usant més memòria.

En computació quàntica, hi ha tres mètriques que ens interessa discutir a l'hora de dissenyar circuits:

- **Nombre de portes:** Òbviament, el nombre de portes que cal per a formar un circuit és una magnitud de vital importància de qualsevol circuit. No té tanta relació amb l'espai que ocupa el circuit ja que s'assumeix que la seqüència de portes a aplicar s'emmagatzema a la part clàssica del computador quàntic, i es van executant de la mateixa manera que un ordinador executa instruccions en el seu processador. Tot i que és cert que aquest nombre de portes tindria un efecte sobre el temps que triga el circuit a finalitzar la seva execució, sovint s'assumeix que diversos operadors es poden "apilar"en un sol pas sempre i quan no hi hagi intersecció entre els qubits sobre els quals actuen.
- **Profunditat:** La profunditat és la millor mesura de la complexitat temporal d'un circuit ja que és igual al nombre de passos que calen per a executar totes les portes d'aquest. Aquesta profunditat és mesurada com la distància mínima des del principi del circuit fins a l'últim operador executat, tenint en compte que l'ordre relatiu d'aquests és fixe.
- **Nombre de qubits addicionals:** Generalment cal un espai major que els que ocupen els operands per a realitzar l'operació. Aquesta magnitud mesura la quantitat d'aquests qubits temporals, sovint inicialitzats a $|0\rangle$, calen per a efectuar el càlcul.

El nombre de portes no sol ser objecte de minimització ja que és la part clàssica la que se n'encarrega de l'aplicació d'aquestes, de manera que segons l'arquitectura que assumim té un ordinador quàntic el nombre de portes en sí no té cap correlació amb la complexitat temporal o espacial.

La profunditat dels circuits, en canvi, sol ser la mètrica més sovint optimitzada ja que té una relació directa amb la complexitat temporal de l'operació, en ser equivalent al nombre de passos (ídem a les instruccions en processadors clàssics).

Actualment, però, el nombre de qubits addicional també és sovint minimitzat per a facilitar la recerca del camp, permetent implementar i testejar els algorismes i circuits en els ordinadors quàntics actuals, els quals són extremadament limitats en el seu nombre de qubits.

Durant la presentació de circuits d'aquesta secció tractarem de mostrar els dissenys publicats amb millors complexitats tant de profunditat com d'espai, per a donar al lector totes les eines necessàries per a implementar les operacions desitjades ja sigui per a ser executades en un ordinador quàntic actual o per a optimitzar al màxim els càlculs per a possibles arquitectures futures de major capacitat.

Cal destacar que, com que la majoria de circuits i dissenys han estat desenvolupats per a ser usats en els algorismes de Shor, molts dels circuits presentats són modificacions dels publicats o directament nous dissenys basats en les idees principals en que es basen. S'ha parat especial atenció en donar el crèdit corresponent a tots els estudis referenciats o usats, o aquells que tot i no ser inclosos en aquest document han estat importants en el desenvolupament del camp de recerca de la computació quàntica.

Per a entendre aquest secció és imperatiu tenir els coneixements bàsics de les representacions de nombres en bits i dels algorismes bàsics per a operar amb aquests mostrats a l'apèndix B.

3.1 Lògica booleana

La lògica booleana és la base de la computació clàssica, permetent la implementació de qualsevol funció sobre bits amb un petit conjunt de portes lògiques bàsiques. Com que el nostre objectiu final és la implementació d'aquestes funcions arbitràries, sembla justificada la decisió de començar investigant la possibilitat de l'existència de circuits quàntics equivalents a aquests portes.

Tot i que les tres portes lògiques més comunes són les **NOT**,**AND** i **OR**, combinacions d'aquestes com **XOR** solen ser implementades independentment per a obtenir dissenys més eficients, donada la seva importància en la majoria de circuits.

Com que l'objectiu del treball és donar la possibilitat de realitzar qualsevol càlcul clàssic, és imperatiu poder realitzar també aquest conjunt d'operacions, de manera que encara que basar la resta de dissenys en aquesta lògica booleana sigui poc pràctic o directament infactible, els circuits seran útils per a una gran quantitat d'algorismes.

A la taula 1 es mostren les taules de la veritat de totes les portes booleanes estudiades en aquesta secció.

a	**NOT**
0	1
1	0

a	**b**	**AND**	**OR**	**XOR**
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Taula 1: *Taules de la veritat de les portes lògiques bàsiques analitzades en aquesta secció.* **Font**: Elaboració pròpia.

A continuació s'estudia la possibilitat d'implementar cadascuna d'aquestes portes mitjançant operadors quàntics reversibles i se'n tracta de donar circuits basats en el conjunt $\{\mathbf{X}, \mathbf{CNOT}, \mathbf{Toffoli}\}$. Cal destacar que aquesta secció fa èmfasi en el procés a seguir per a obtenir operadors a partir de qualsevol funció booleana, de manera que val la pena llegir-la encara que el lector no requereixi circuits de les operacions booleanes bàsiques.

3.1.1 Porta NOT

Ja s'ha comentat que la versió quàntica de la porta lògica NOT és l'operador X, però per completesa donem una petita explicació de com s'obtindria la versió quàntica d'aquesta porta de zero.

Primerament tractarem de dissenyar una versió de la porta que actuï sobre un únic qubit, tal i com ho fa la porta NOT booleana:

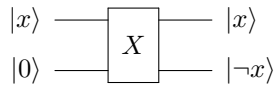
$$|x\rangle \longrightarrow \boxed{X} \longrightarrow |\neg x\rangle$$

Tal i com s'ha comentat anteriorment, coneixem l'efecte d'aquesta porta sobre tots els estats bàsics del sistema d'un qubit sobre el que actua, el qual ens permet definir l'operador:

$$X|0\rangle = |1\rangle$$
$$X|1\rangle = |0\rangle$$

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Ara bé, si pensem en el nostre objectiu d'implementar operacions aritmètiques tal i com ho fan els ordinadors clàssics veurem que un estat ha de poder ser usat per a realitzar diversos còmputs. El mètode general usat en la literatura actual per a solucionar aquest problema es basa en dissenyar operadors que no modifiquin els qubits sobre els que actuen i emmagatzemin el resultat en un qubit addicional. El diagrama d'aquesta porta X sería:



El qual ens permetria mantenir tant el nostre estat inicial com el resultat per a futures operacions. Malauradament, definir l'operador no és tant simple com en el cas anterior, ja que no hem definit l'acció d'aquest obre tots els estats inicials possibles:

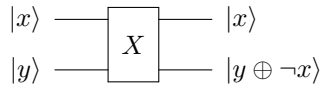
$$X\,|yx\rangle = |y'x'\rangle$$

$$\begin{aligned} X\,|00\rangle &= |10\rangle \\ X\,|01\rangle &= |11\rangle \\ X\,|10\rangle &= ? \\ X\,|11\rangle &= ? \end{aligned}$$

El qual pot ser fàcilment resolt de forma general de dues maneres:

- Assignar qualsevol estat final a les entrades no definides, assegurant que la matriu sigui unitària, i assumir que l'entrada sempre serà 0.
- Extendre la definició de l'operador per a actuar de manera coneguda sobre tots els estats possibles. Aquesta sol ser l'opció preferible.

En el nostre cas extendrem la definició de la porta a:



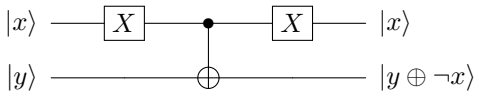
On $\oplus$ és la XOR lògica. Ara sabem l'efecte sobre tots els estats bàsics i podem definir la matriu de l'operador:

$$X\,|yx\rangle = |y'x'\rangle$$

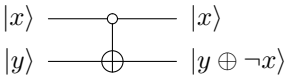
$$\begin{aligned} X\,|00\rangle &= |10\rangle \\ X\,|01\rangle &= |11\rangle \\ X\,|10\rangle &= |00\rangle \\ X\,|11\rangle &= |01\rangle \end{aligned}$$

$$X = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

El qual és fàcilment implementable mitjançant una **CNOT**:



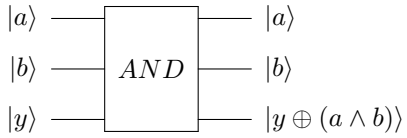
O equivalentment:



El qual s'anomena **CNOT0**, i és equivalent a una **CNOT** condicionada a l'estat $|0\rangle$ i no a $|1\rangle$.

3.1.2 Porta AND

És impossible implementar aquesta operació sobre dos qubits ja que l'operador ha de ser reversible, de manera que tractarem de dissenyar la versió amb el qubit addicional pel resultat directament:



Com sempre, coneixem el seu efecte sobre tots els estats bàsics:

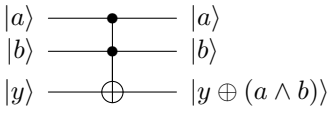
$$AND\,|yba\rangle = |y'b'a'\rangle$$

$$\begin{aligned}
AND\,|000\rangle &= |000\rangle \\
AND\,|001\rangle &= |001\rangle \\
AND\,|010\rangle &= |010\rangle \\
AND\,|011\rangle &= |111\rangle \\
AND\,|100\rangle &= |100\rangle \\
AND\,|101\rangle &= |101\rangle \\
AND\,|110\rangle &= |110\rangle \\
AND\,|111\rangle &= |011\rangle
\end{aligned}$$

El qual ens permet definir la matriu:

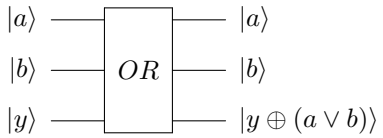
$$AND = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{bmatrix}$$

Aquest operador és també conegut amb el nom de porta *Toffoli*, i és equivalent a una NOT condicionada a dos qubits:



3.1.3 Porta OR

Igual que en el cas anterior, implementacions sobre dos qubits no poden ser reversibles així que trobarem l'operador corresponent al següent circuit:



El qual té un efecte conegut sobre cada estat bàsic:

$$\begin{aligned} OR\,|yba\rangle &= |y'b'a'\rangle \\ \\ OR\,|000\rangle &= |000\rangle \\ OR\,|001\rangle &= |101\rangle \\ OR\,|010\rangle &= |110\rangle \\ OR\,|011\rangle &= |111\rangle \\ OR\,|100\rangle &= |100\rangle \\ OR\,|101\rangle &= |001\rangle \\ OR\,|110\rangle &= |010\rangle \\ OR\,|111\rangle &= |011\rangle \end{aligned}$$

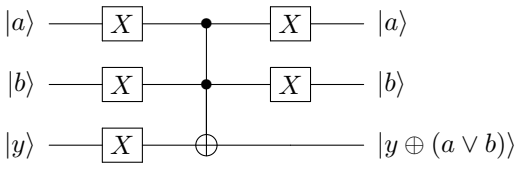
Amb matriu derivada:

$$OR = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

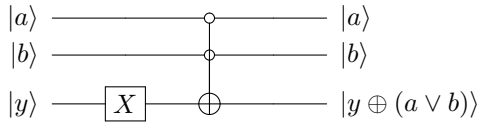
Com que el conjunt bàsic {**X**,**CNOT**,**Toffoli**} és bastant comú en la literatura, val la pena obtenir una implementació d'aquest operador mitjançant aquestes portes. Podem reescriure la funció gràcies a les propietats de la lògica booleana:

$$a \vee b = \neg(\neg a \wedge \neg b)$$

El qual permet implementar l'operador mitjançant aquest conjunt:



O equivalentment:



El resultat és justificat veient els estats després de cada pas:

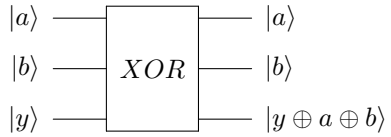
$$|y\rangle\,|b\rangle\,|a\rangle$$
$$|\neg y\rangle\,|\neg b\rangle\,|\neg a\rangle$$
$$|\neg y \oplus (\neg b \wedge \neg a)\rangle\,|\neg b\rangle\,|\neg a\rangle$$

On podem transformar el valor del registre y, sabent que $x \oplus y \equiv \neg x \oplus \neg y$:

$$\neg y \oplus (\neg b \wedge \neg a)$$
$$y \oplus \neg(\neg b \wedge \neg a)$$
$$y \oplus (b \vee a)$$

3.1.4 Porta XOR

El procediment és anàleg als anteriors:



El qual té un efecte conegut sobre cada estat bàsic:

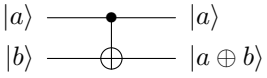
$$XOR\,|yba\rangle = |y'b'a'\rangle$$

$$\begin{aligned}
XOR\,|000\rangle &= |000\rangle \\
XOR\,|001\rangle &= |101\rangle \\
XOR\,|010\rangle &= |110\rangle \\
XOR\,|011\rangle &= |011\rangle \\
XOR\,|100\rangle &= |100\rangle \\
XOR\,|101\rangle &= |001\rangle \\
XOR\,|110\rangle &= |010\rangle \\
XOR\,|111\rangle &= |111\rangle
\end{aligned}$$

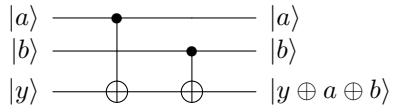
Amb matriu derivada:

$$OR = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

Aquesta porta pot ser implementada sobre dos qubits, ja que el seu funcionament és equivalent a la negació condicionada de la CNOT:



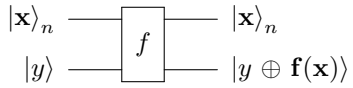
El qual permet també implementar-la mitjançant portes del conjunt {**X**,**CNOT**,**Toffoli**}:



3.1.5 Portes booleanes arbitràries

Qualsevol funció booleana de la forma $f : \{0,1\}^n \to \{0,1\}^m$ pot ser representada mitjançant m funcions de la forma $f : \{0,1\}^n \to \{0,1\}$, on cadascuna calcula un dels bits de la sortida.

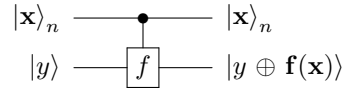
Qualsevol funció booleana de la forma $\{0,1\}^n \to \{0,1\}$ pot implementar-se sobre un ordinador quàntic mitjançant un circuit de la forma:



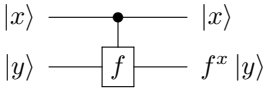
Això és cert ja que la funció f es pot entendre com una permutació dels elements del conjunt: $f = \{(\mathbf{x}, y))|\mathbf{x} \in \{0,1\}^n \wedge y \in \{0,1\}\}$ de manera que es compleixi:

$$\begin{array}{lcl}(\mathbf{x},0) & \to & (\mathbf{x}, f(\mathbf{x})) \\ (\mathbf{x},1) & \to & (\mathbf{x}, \neg f(\mathbf{x}))\end{array}$$

Aquesta funció també es pot entendre també com una negació de l'estat de $|y\rangle$ condicionada al valor de $f(\mathbf{x})$, el qual és equivalent a una **CNOT** quan f és la identitat. Aquesta semblança se sol usar per a representar aquestes funcions que apliquen una porta **X** condicionada a una funció sobre uns estats sense modificar de la forma:



Cal destacar que aquesta nomenclatura és equivalent a la de l'aplicació d'una porta condicionada a un o diversos qubits:



On f^x indica l'aplicació de la porta f sobre l'estat $|y\rangle$ només quan $|x\rangle = |1\rangle$[2]. Tot i usar la mateixa nomenclatura, l'opció usada serà sempre obvia donat el context o és detallada explícitament en cas contrari.

Usant aquesta nomenclatura, els diagrames dels prototips de les tres portes **AND**, **OR**, **XOR** es mostren a la figura 7.

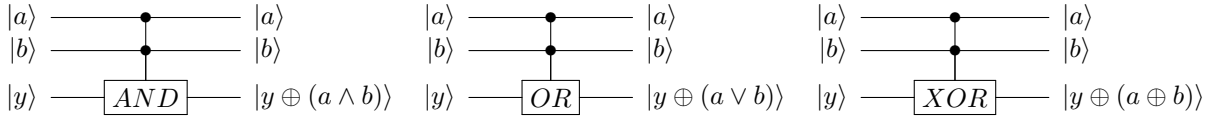


Figura 7: *Circuits quàntics de les portes booleanes bàsiques **AND**,**OR** i la composta **XOR** mitjançant la nomenclatura de les CNOT.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

[2]La notació $f^n(x)$ indica l'aplicació de la funció f n vegades, de la forma $f(f(\dots f(x)\dots))$. En aquest cas si $x = 0$ no s'aplica la funció cap vegada, i si $x = 1$ sí que s'aplica, el qual és equivalent a condicionar l'execució de la funció a l'estat d'un qubit.

3.2 Suma de naturals i enters

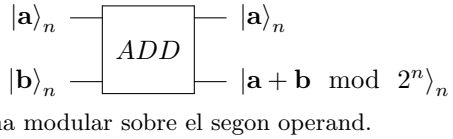
Representarem enters mitjançant la codificació *complement a 2*, de manera que el circuit per a sumar enters és el mateix que permet sumar naturals.

L'estudi d'aquests circuits va començar a finals del segle passat amb la publicació dels algorismes quàntics de factorització i logaritme discret al 1994 ([6]), per a tractar d'obtenir circuits capaços d'implementar els passos d'aquest algorisme que requereixen de l'aplicació de funcions numèriques. Les primeres publicacions detallant circuits per a aquestes operacions aritmètiques van sortir dos any després ([7], [8]), amb dissenys basats en les portes **X**, **CNOT** i **Toffoli** i amb profunditats lineals respecte el nombre de qubits dels operands.

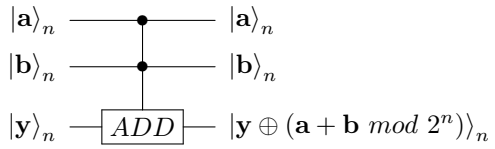
Documents posteriors proveeixen dissenys amb millor complexitat de profunditat, a cost de l'ús de qubits addicionals ([9], [10], [11], [12]), o circuits sense cap qubit addicional amb profunditat lineal ([13]), o fins i tot dissenys que empren una operació coneguda com a **Transformada de Fourier Quàntica** per a realitzar aquesta suma ([4],[14], [15], [16]).

De la mateixa manera que ens pot fer falta una porta **NOT** que o bé actuï sobre un sol qubit o bé dipositi el resultat en un altre qubit, o una **XOR** que modifiqui una de les entrades o un qubit auxiliar, pot ser que la operació de suma per la qual volem dissenyar un circuit actuï sobre un dels operands o sobre un registre apart.

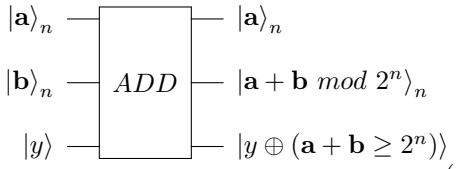
És per això que en les seccions següents començarem presentant els diversos prototips de les versions de l'operació en particular a dissenyar. Per exemple, en el cas de la suma de dos registres de n qubits, tractarem de proporcionar dissenys capaços de realitzar les operacions mostrades en la figura 8.



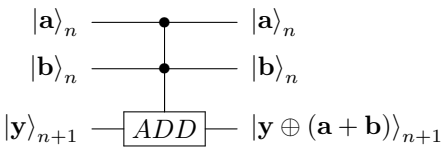
(a) Suma modular sobre el segon operand.



(b) Suma modular sobre un registre addicional.



(c) Suma no modular sobre el segon operand.



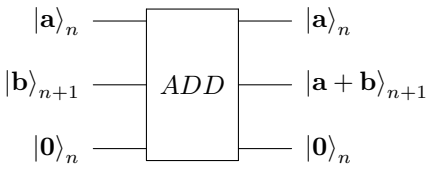
(d) Suma no modular sobre un registre addicional.

Figura 8: *Prototips de l'operador ADD per a modificar el segon operand amb el resultat (a i c) o no modificar els operands i emmagatzemar el resultat en un registre addicional (b i d) per a sumes modulars i no modulars.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Els documents que presenten implementacions d'aquests prototips solen representar els diagrames directament en funció de les operacions del conjunt bàsic de portes, sovint {**X**,**CNOT**,**Toffoli**}. Com que l'objectiu d'aquest document és la divulgació d'aquests dissenys de manera que el lector els sigui capaç d'entendre i modificar, els construirem de manera jeràrquica mitjançant operadors de complexitat progressiva.

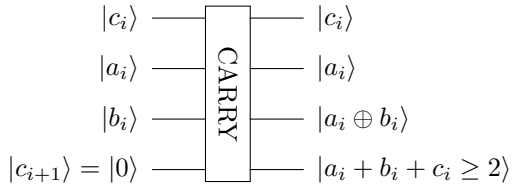
3.2.1 Circuit bàsic basat en Ripple-Carry

El primer circuit per a sumar dos registres quàntics va ser publicat el 1996 ([7]), i el seu prototip té la forma:



És a dir, implementa la versió de suma no modular sobre un dels operands i usa un registre temporal de n qubits el qual pot ser reutilitzat.

El seu funcionament es basa en l'aplicació iterativa de modificacions del bloc Full Adder típic de l'aritmètica en computadors clàssics, el qual es centra en el càlcul del ròssec de cada operació:



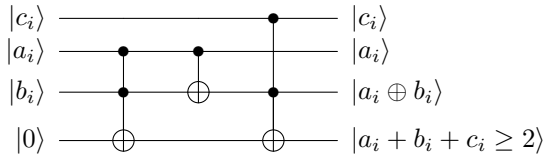
El qual realitza la suma de tres qubits c_i, a_i i b_i i emmagatzema el bit de major pes del resultat en un qubit addicional c_{i+1}. El significat de cada qubit és:

- $\mathbf{a_i}$ Qubit i d'un dels registres a sumar.
- $\mathbf{b_i}$ Qubit i d'un dels registres a sumar.
- $\mathbf{c_i}$ Ròssec de la suma de l'anterior parell de qubits.
- $\mathbf{c_{i+1}}$ Ròssec de la suma de l'actual parell de qubits amb el ròssec anterior, inicialitzat a 0.

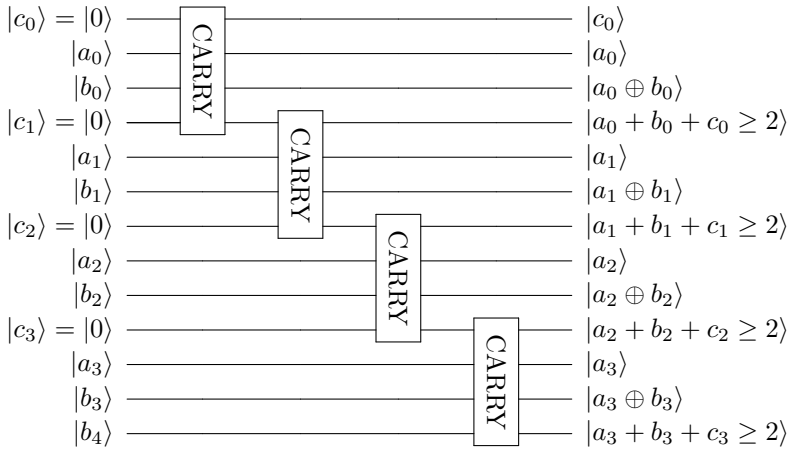
El càlcul de $a_i + b_i + c_i \geq 2$ es pot definir mitjançant lògica booleana:

$$\begin{aligned} a_i + b_i + c_i \geq 2 &\equiv a_i \wedge b_i \vee a_i \wedge c_i \vee b_i \wedge c_i \\ &\equiv a_i \wedge b_i \oplus (c_i \wedge (a_i \oplus b_i)) \end{aligned}$$

Permetent definir la porta **CARRY** amb tan sols dues portes **Toffoli** i una **CNOT**:



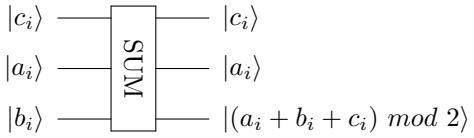
Gràcies a aquesta porta podem construir un circuit capaç de calcular els ròssecs de les sumes de cada terna c_i, a_i i b_i, el qual té la següent forma per nombres de 4 qubits:



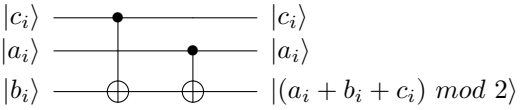
El qual actua sobre dos registres de 4 i $4+1$ qubits $|\mathbf{a}\rangle_4$ i $|\mathbf{b}\rangle_5$ que contenen enters o naturals de 4 bits i usa un registre addicional per als ròssecs $|\mathbf{c}\rangle_4$ inicialitzat a $|\mathbf{0}\rangle_4$.

Veiem que els valors d'aquest registre de ròssecs són correctes, de manera que tan sols cal trobar alguna forma d'usar aquests resultats per a transformar els operands en a i $a+b$ i netejar el registre de ròssecs a $|\mathbf{0}\rangle_n$.

El disseny usa un altre bloc per a aquesta segona part, anomenat **SUM**, el qual simplement realitza la suma de tres qubits mòdul 2:



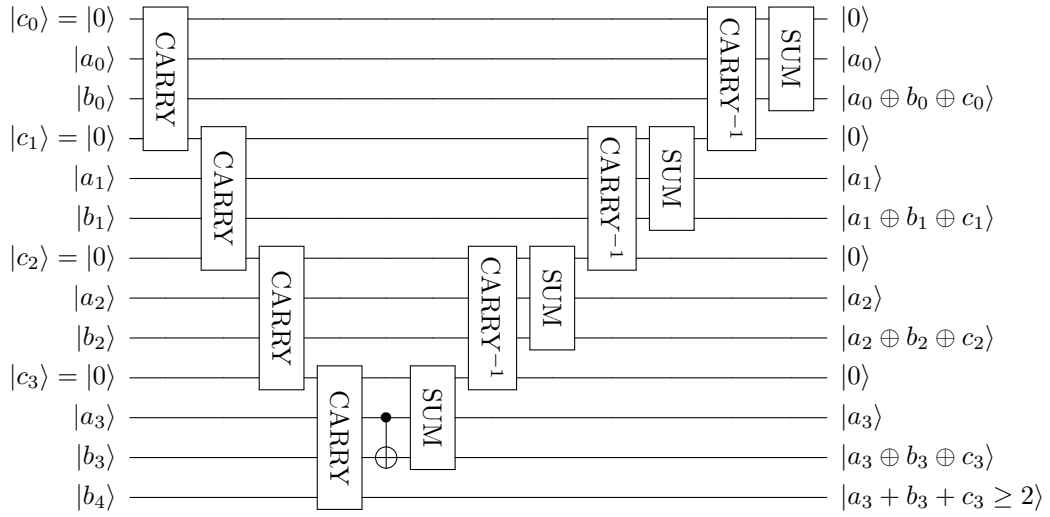
Aquesta suma es pot calcular de manera trivial mitjançant dues **CNOT**:



El funcionament de la segona part es basa en desfer tota la computació realitzada per les portes **CARRY** mitjançant les seves inverses però usar els ròssecs abans per a calcular cada valor final del segon operand. Primerament es realitza una única **CNOT** per a transformar el qubit $|a_3 \oplus b_3\rangle$ a $|b_3\rangle$, i s'aplica la porta **SUM** per a calcular el bit 3 de la suma $a+b$. Seguidament s'aplica el mateix conjunt d'operacions iterativament per cada bit anterior del resultat:

- Aplicar l'inversa de **CARRY** per a retornar els qubits b_i i c_{i+1} als estats inicials (a_i i c_i no són modificats en cap moment per la porta **CARRY** que estem desfent).
- Aplicar l'operador **SUM** per a calcular el bit de la suma, el qual es pot fer gràcies al fet que hem calculat tots els ròssecs anteriorment, mitjançant la igualtat: $(a+b)_i = a_i \oplus b_i \oplus c_i$.

El circuit pren la forma següent:



Tot i que no és mencionat en el document original, el primer bloc **SUM** desfà l'efecte de la **CNOT** prèvia, de manera que podem substituir aquest parell de portes per una sola **CNOT**. La figura 9 mostra el circuit final.

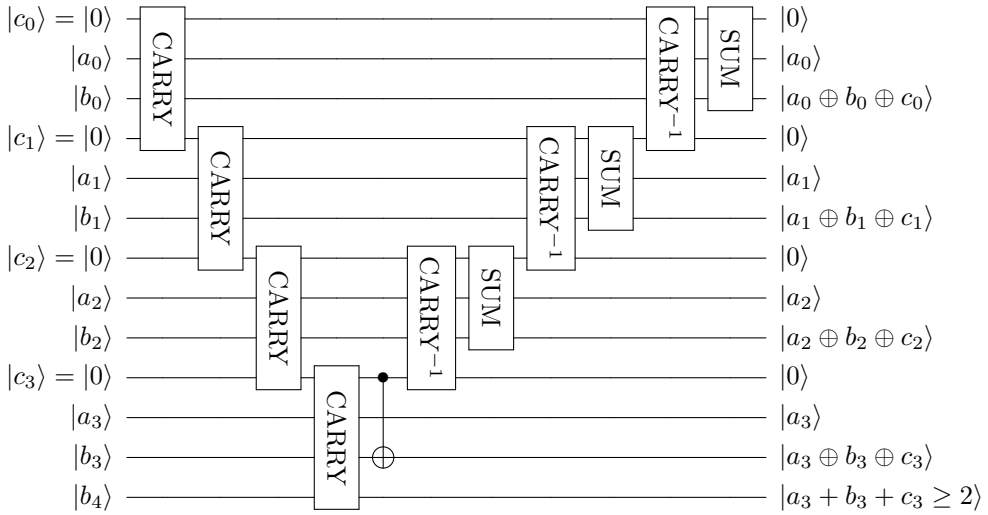


Figura 9: *Implementació bàsica de l'operador ADD per Ripple-Carry per a sumar dos registres* $|\boldsymbol{a}\rangle_n$ *i* $|\boldsymbol{b}\rangle_{n+1}$ *sobre el segon operador, usant un registre addicional pels ròssecs* $|\boldsymbol{c}\rangle_n$ **Font:** Elaboració pròpia mitjançant Q-Circuit. Basat en el disseny de [7].

La profunditat del circuit és lineal respecte el tamany n dels registres, de l'ordre $\approx 8n$.

El nombre de qubits addicionals requerits és $\mathbf{O(n)}$, de fet és exactament n.

El nombre de portes en el diagrama i la quantitat d'operadors del conjunt bàsic necessaris per a obtenir-ne una implementació completa es mostren a la taula 2.

	Circuit				**Conjunt bàsic**		
Porta	CARRY	CARRY^{-}1	SUM	CNOT	X	CNOT	Toffoli
Quantitat	n	$n-1$	n	1	0	$4n$	$4n-2$

Taula 2: *Nombre de portes de la implementació bàsica de l'operador ADD per Ripple-Carry. A l'esquerra es mostra el nombre de portes compostes presents en el diagrama per enters de n qubits, i a la dreta el nombre de portes del conjunt bàsic que calen per a construir el circuit des de zero.* **Font:** Elaboració pròpia.

Cal destacar, però, que un dels nostres objectius era obtenir la suma modular de a i b, de manera que aquest disseny no s'ajusta completament al prototip del circuit de suma descrit inicialment. El document original descriu un mètode per a obtenir un circuit de suma modular per un mòdul arbitrari, però aquest ha estat omès degut a les altes fites de profunditat i nombre de portes bàsiques requerides (unes 6 vegades major al circuit de suma no modular anterior). Si es vol obtenir la versió modular descrita en el prototip, tan sols cal eliminar l'últim qubit i la part del bloc **CARRY** que el modifica.

3.2.2 Ripple-carry modular i no modular sense qubits addicionals

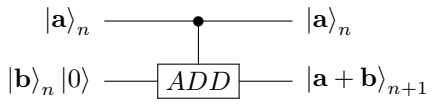
Estudis posteriors han desenvolupat el mateix concepte d'implementar variants quàntiques dels blocs Full-Adder per a realitzar la suma en ordinadors quàntics, anomenat generalment *Ripple-Carry*. Uns dels principals objectius d'aquests estudis han estat la reducció del nombre de qubits addicionals i del nombre de portes bàsiques requerides per a construir el circuit.

Documents com [11] presenten un disseny que redueix el nombre de qubits extres de n a 1, el qual va ser posteriorment optimitzat a 0 en [13] i [17] sense pèrdua en complexitat del nombre de portes o profunditat.

La reducció del nombre de portes bàsiques ha estat també estudiada en aquests documents, reduint el nombre d'operadors de $\approx 6 \cdot 8n$ ([2]), a $\approx 9n$ ([11]), $7n$ ([17]) fins a $\approx 6n$ ([12]).

Altres estudis han tractat de reduir la profunditat del circuit a detriment de la complexitat d'aquest mitjançant mètodes alternatius com *Carry-Lookahead*, de manera que en aquesta secció presentarem un circuit que minimitzi el nombre de qubits addicionals, i en properes seccions descriurem els estudis més importants en la reducció de la complexitat temporal dels circuits de suma i la quantitat d'operadors bàsics, i donarem també alternatives que minimitzin aquestes mètriques.

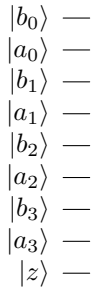
És per això que presentarem el disseny [17] en aquesta secció, en tenir una de les millors fites de complexitat i no requerir cap qubit addicional. El seu prototip té la forma:



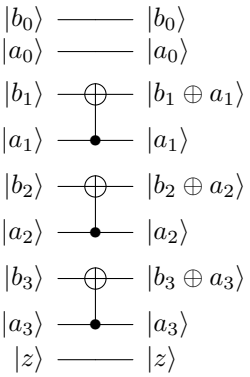
És a dir, calcula la suma no modular de a i b sobre el segon operand, emmagatzemant el ròssec de la suma en el tercer sense usar qubits addicionals.

El funcionament d'aquest circuit es basa en calcular tots els resultats intermedis sobre els qubits dels operands i propagar-los sobre tots els parells $|a_i\rangle$, $|b_i\rangle$. Aquest disseny no és tant intuïtiu com l'anterior ja que ha estat enormement optimitzat, de manera que presentarem la seqüència de resultats intermedis que es calculen en cada pas i mostrarem com, efectivament, la combinació d'aquests passos culminen en el resultat desitjat.

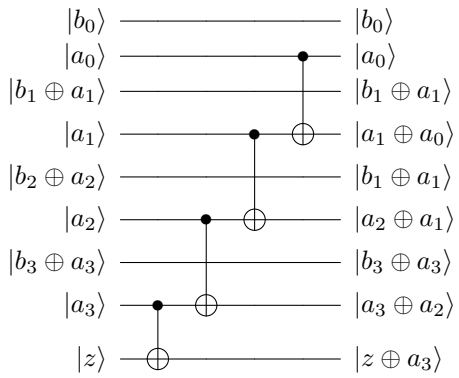
Assumim que tenim 4 qubits i el nostre sistema comença en l'estat:



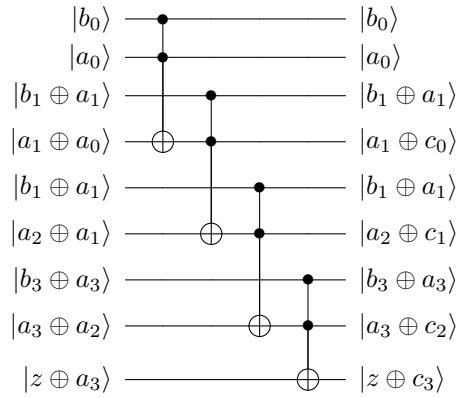
Comencem aplicant una **CNOT** sobre cada $|a_i\rangle$ i $|b_i\rangle$ per $i > 0$:



Apliquem una **CNOT** sobre cada parell $|a_i\rangle$, $|a_{i+1}\rangle$ començant per $i = n - 1$ fins a $i = 1$ ($|z\rangle$ es comporta com a_n):

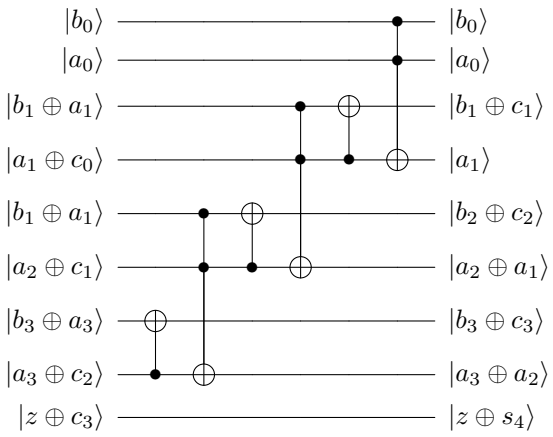


Seguidament apliquem una **Toffoli** sobre cada $|a_i\rangle$, $|b_i\rangle$ i $|a_{i+1}\rangle$ per $i = 0, \ldots, n - 1$:



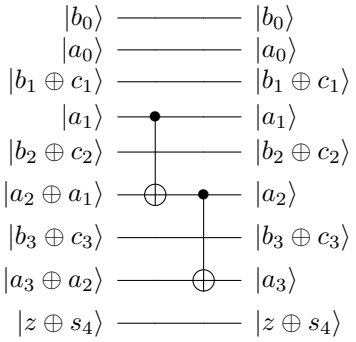
On $c_i = a_{i-1} \wedge b_{i-1}$.

Després, per $i = n-1, \ldots, 1$, aplicar una **CNOT** igual a les del primer pas; sobre cada parell $|a_i\rangle$, $|b_i\rangle$, seguida d'una **Toffoli** sobre $|a_{i-1}\rangle$, $|b_{i-1}\rangle$ i $|b_i\rangle$:



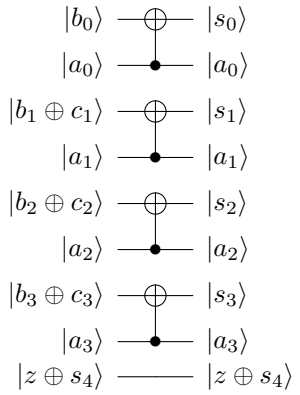
On s_i és el bit i de la suma $a + b$. Veiem que hem trobat el ròssec final de la suma.

El cinquè pas consisteix en aplicar una **CNOT** a cada parell $|a_i\rangle$, $|a_{i+1}\rangle$ per $i = 1, \ldots, n-2$:



Veiem que hem reiniciat els estats dels qubits del registre $|\mathbf{a}\rangle_n$, i que tots els registres de $|\mathbf{b}\rangle_n$ contenen la suma exclusiva dels seus elements inicials amb els ròssecs calculats en els passos anteriors.

Podem transformar aquests elements del registre $|\mathbf{b}\rangle_n$ en els resultats de la suma simplement aplicant una **CNOT** a cada parell $|a_i\rangle$ i $|b_i\rangle$ per $i = 0, \ldots, n-1$ de manera paral·lela:



El qual és el resultat que desitgem: a al registre $|\mathbf{a}\rangle_n$ i $a + b$ als registres $|\mathbf{a}\rangle_n$ i $|z\rangle$.

La figura 10 mostra el circuit complet, obtingut unint les operacions realitzades en els sis passos.

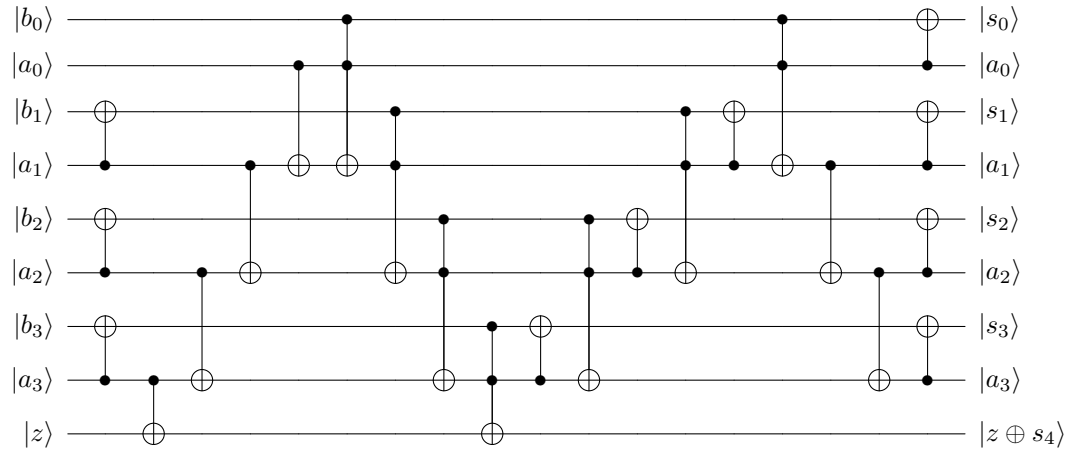


Figura 10: *Implementació Ripple-Carry de l'operador ADD per a sumar dos registres $|\boldsymbol{a}\rangle_5$ i $|\boldsymbol{b}\rangle_5$ sobre el segon operador i un qubit addicional pel ròssec.* **Font:** Elaboració pròpia mitjançant Q-Circuit. Basat en el disseny de [17].

A part de ser molt més simple que el primer disseny de suma per Ripple-Carry, podem modificar aquest circuit de manera trivial per a calcular la suma modular dels operands suprimint les portes que actuen sobre el qubit a eliminar. Aquesta modificació es mostra a la figura 11.

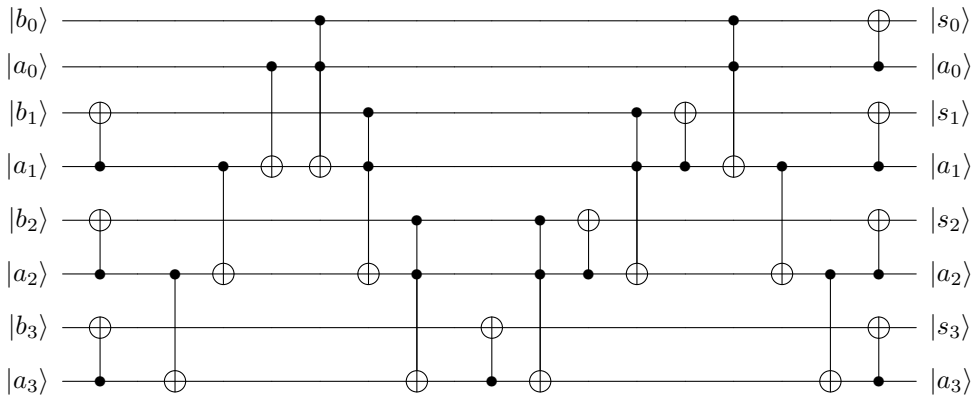


Figura 11: *Implementació Ripple-Carry de l'operador ADD per a sumar dos registres* $|\boldsymbol{a}\rangle_5$ *i* $|\boldsymbol{b}\rangle_5$ *sobre el segon operador de manera modular.* **Font:** Elaboració pròpia mitjançant Q-Circuit. Basat en el disseny de [17].

Complexitat

La profunditat del circuit és lineal respecte el tamany n dels registres, de l'ordre $\approx 5n$.

El nombre de qubits addicionals requerits és 0.

La quantitat d'operadors del conjunt bàsic necessaris per a obtenir-ne una implementació completa es mostren a la taula 3.

	Conjunt bàsic		
Porta	X	CNOT	Toffoli
Quantitat	0	$5n-2$	$2n-1$

Taula 3: *Nombre de portes de la implementació Ripple-Carry de ADD. La taula mostra el nombre de portes del conjunt bàsic que calen per a construir el circuit des de zero.* **Font:** Elaboració pròpia.

3.2.3 Carry-Lookahead modular i no modular en temps logarítmic

Carry-Lookahead és una tècnica alternativa a Ripple-Carry per a realitzar la suma d'enters basada en obtenir els ròssecs de manera paral·lela en temps logarítmic i calcular el resultat en una sola operació $s_i = a_i \oplus b_i \oplus c_i$. Aquest mètode és comú en computadors clàssics, però estudis sobre la seva implementació sobre el model quàntic van començar a ser publicats el 2004 amb [9], on s'estudia una adaptació del mètode construït a partir de portes del conjunt {**X**,**CNOT**,**Toffoli**}. Aquest document presenta un circuit de suma amb un enorme estalvi temporal respecte els seus predecessors, ja que la profunditat logarítmica del circuit es tradueix a temps de complexitat similar, a cost d'usar una quantitat lineal de qubits addicionals.

Aquest disseny de $O(log\ n)$ profunditat i $O(n)$ qubits extres va ser generalitzat en [17] amb un mètode per a construir qualsevol circuit amb profunditat $O(d(n))$ i $O(\frac{n}{d(n)})$ qubits addicionals per qualsevol $d(n) = \Omega(log\ n)$. Encara més sorprenentment, si s'assumeix l'existència de portes **fan-out**[3] sobre $O(n^\epsilon)$, per ϵ constant i petita, es pot construir un circuit de suma de profunditat $O(log^* n)$[4].

En aquesta secció, però, presentarem la versió amb profunditat $O(log\ n)$ [9] ja que el mètode generalitzat és massa complex per a explicar-se sense context, i de totes maneres es pot obtenir el circuit òptim modificant el disseny que presentarem.

Abans d'entrar més en detall en els circuits, explicarem el funcionament del mètode Carry-Lookahead. Ja hem mencionat anteriorment que si coneixem els dos operands d'una suma en base 2 a i b i els ròssecs de cadascuna de les sumes individuals c_i podem calcular cada bit del resultat final $s = a + b$ com $s_i = a_i \oplus b_i \oplus c_i$, de manera que la complexitat dels algorismes de suma és deguda únicament al mètode usat per a obtenir aquests ròssecs.

El mètode Ripple-Carry els calcula de manera iterativa tal i com es fa en l'algorisme de suma a mà: $c_{i+1} = a_i + b_i + c_1 \geq 2$, però el mètode Carry-Lookahead realitza aquest càlcul d'una manera alternativa. El seu funcionament es basa en el fet que quan $a_i = b_i$ podem conèixer el valor de c_{i+1} sense saber c_i ja que o bé $c_{i+1} \equiv a_i + b_i + c_i \geq 2 \equiv 0 + 0 + c_i \geq 2 \equiv 0$ o $c_{i+1} \equiv a_i + b_i + c_i \geq 2 \equiv 1 + 1 + c_i \geq 2 \equiv 1$.

La clau de l'algorisme és una variable anomenada estatus del ròssec, o *Carry status* en el document original. Aquesta està definida sobre cada possible interval $C[i, j]$ per $0 \leq i < j < n$, i tan sols pot prendre un dels tres valors *Kill* (k), *Propagate* (p), *Generate* (g). De manera informal, i per ajudar a la comprensió de l'explicació, es pot entendre aquest valor com a l'indicador de com *canviarà* el ròssec en l'interval:

- Si $C[i, j] = k$, independentment del valor del ròssec c_i, c_j serà 0. Es diu que l'interval mata (*kills*) el ròssec.
- Si $C[i, j] = p$, el valor de c_j serà igual al de c_i. Es diu que l'interval propaga (*propagates*) el ròssec.
- Si $C[i, j] = g$, independentment del valor del ròssec c_i, c_j serà 1. Es diu que l'interval genera (*generates*) el ròssec.

Cal notar que en aquesta explicació assumirem que $c_0 = 0$, tot i que el circuit final es podrà modificar de manera trivial per a ser més general. Podem calcular els valors de cada $C[i, i+1]$ per $0 \leq i < n-1$ de manera òbvia a partir dels valors a_i i b_i:

- Si $a_i = b_i = 0$ mai hi haurà ròssec entre i i $i + 1$, de manera que $C[i, i + 1] = k$ ja que "mata"qualsevol ròssec anterior.
- Si $a_i = b_i = 1$ sempre hi haurà ròssec entre i i $i + 1$, de manera que $C[i, i + 1] = g$ ja que "genera"sempre un ròssec.
- Si $a_i \neq b_i$ un és 1 i l'altre 0 de manera que el ròssec es propaga: $c_{i+1} \equiv a_i + b_i + c_i \geq 2 \equiv 1 + c_i \geq 2 \equiv c_i$. $C[i, i + 1] = p$ ja que es "propaga"el ròssec anterior a l'interval.

A partir d'aquesta base podem definir un interval arbitrari mitjançant una partició d'aquest $[i, j] = [i, k] \cup [k + 1, j]$:

[3] Una porta fan-out (F_t) clàssica és simplement una unió dels t cables per a que tots tinguin el mateix valor. En literatura quàntica s'assumeix que aquest operador actua sobre un estat de la manera següent: $F_t\,|y\rangle \bigotimes_{i=1}^{t} |x_i\rangle = |y\rangle \bigotimes_{i=1}^{t} |x_i \oplus y\rangle$.

[4] $log^* n$ s'anomena logaritme iteratiu de n, i és el nombre de vegades que cal aplicar el logaritme per a obtenir un valor igual o inferior a 1. Aquesta funció creix extremadament lent, prenent per exemple base 2 i valors de $n \leq 2^{65536}$(major al nombre d'atoms de l'univers) el valor de la funció no supera 5, fent la seva complexitat constant a la pràctica.

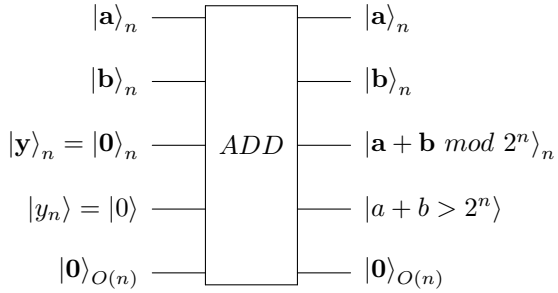
$C[i,j]$		$C[k,j]$		
		k	p	g
$C[i,k]$	k	k	k	g
	p	k	p	g
	g	k	g	g

Podem veure que la definició anterior ajuda a entendre la taula, ja que quan el segon subinterval és k, el valor per l'interval també ho és ja que es "mata", quan és p es copia el del primer, i quan és g no importa el valor de l'anterior, ja que el segon crea el nou ròssec per si mateix.

El valor de k no afecta al càlcul, però escollir intervals el doble de grans cada vegada permet construir la taula en temps logarítmic en n, el qual és el motiu del millor rendiment d'aquest algorisme.

Un cop calculat tot l'interval, podem obtenir els ròssecs a partir de la relació $c_i \equiv C[0,i] = g$, ja que com que hem assumit un ròssec inicial de 0, el valor del ròssec c_i serà 1 si i només si en algun moment s'ha generat el ròssec en l'interval $[0,i]$.

Aquest és l'algorisme el qual ens permetrà calcular la suma de dos registres en un ordinador quàntic. Específicament, el prototip que implementarem és el següent:



És a dir, mostrarem un circuit per a realitzar la suma no modular de dos registres de n qubits sobre un registre addicional $|\mathbf{y}\rangle_n |y_n\rangle$ de $n+1$ qubits a 0 i usant-ne $O(n)$ d'auxiliars. El valor exacte d'aquest nombre de qubits addicionals es detallarà posteriorment en l'anàlisi del circuit.

Per a aplicar aquest algorisme per a realitzar la suma en un ordinador quàntic cal primer modificar-lo una mica per a que s'ajusti a les restriccions del model. Primerament cal veure que no podem emmagatzemar la variable C en qubits ja que té tres valors possibles, i tan sols en podem representar dos. Per a assolir això simplement usarem dos qubits per a codificar el valor de $C[i,j]$, un anomenat $p[i,j]$ (per *propagates*) i un per $g[i,j]$ (per *generates*). La relació entre la variable $C[i,j]$ original i les noves és:

$C[i,j]$	$p[i,j]$	$g[i,j]$
k	0	0
g	0	1
p	1	0

Podem definir aquestes de manera recursiva de manera anàloga a l' original. Un interval propaga el ròssec si i només si els seus dos subintervals també ho fan:

$$p[i,j] = p[i,k] \wedge p[k,j]$$

Un interval genera un ròssec si i només si o bé el segon subinterval el genera, o si el primer subinterval el genera i el segon el propaga:

$$\begin{aligned} g[i,j] &= (g[i,k] \wedge p[k,j]) \vee g[k,j] \\ &= (g[i,k] \wedge p[k,j]) \oplus g[k,j] \end{aligned}$$

Podem canviar el símbol $\vee$ per $\oplus$ ja que els dos termes mai seran certs alhora.

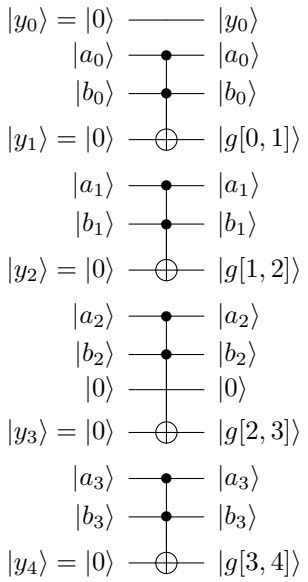
La implementació de l'algorisme es divideix en una sèrie de passos, els quals anirem aplicant seqüencialment a mesura que detallem el seu objectiu i funcionament. Assumim que volem realitzar la suma de dos registres de 4 qubits i comencem en l'estat:

$$\begin{aligned} |y_0\rangle = |0\rangle &\ — \\ |a_0\rangle &\ — \\ |b_0\rangle &\ — \\ |y_1\rangle = |0\rangle &\ — \\ |a_1\rangle &\ — \\ |b_1\rangle &\ — \\ |y_2\rangle = |0\rangle &\ — \\ |a_2\rangle &\ — \\ |b_2\rangle &\ — \\ |0\rangle &\ — \\ |y_3\rangle = |0\rangle &\ — \\ |a_3\rangle &\ — \\ |b_3\rangle &\ — \\ |y_4\rangle = |0\rangle &\ — \end{aligned}$$

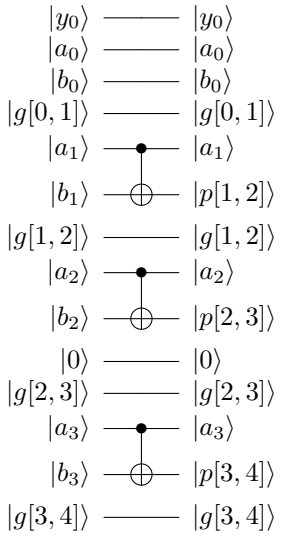
Cal notar el qubit addicional entre $|b_2\rangle$ i $|y_3\rangle$. En el cas on n és quatre el tamany del registre addicional és de 1, però el valor exacte per a un tamany arbitrari dels registres és $n - w(n) - \lfloor log\ n \rfloor$[5]. El motiu es detallarà en el passos següents.

Primerament apliquem una **Toffoli** a cada $|a_i\rangle$, $|b_i\rangle$, $|y_{i+1}\rangle$ per $0 \geq i < n$ amb l'objectiu d'emmagatzemar $a_i \wedge b_i$ sobre y_{i+1}, el qual assumirem que és igual a $|0\rangle$ inicialment. El resultat serà que per tot $0 \geq i < n$ es compleix $y_{i+1} = a_i \wedge b_i = g[i, i+1]$.

[5] $w(n)$ és la suma dels bits a 1 de la representació en base 2 de n. Aquesta funció compleix $n = w(n) + \sum_{i=1}^{\inf} \lfloor \frac{n}{2^i} \rfloor$.



Després apliquem una **CNOT** a cada parell $|a_i\rangle$, $|b_i\rangle$ per $0 < i < n$ amb l'objectiu de garantir que, per a aquests mateixos índexs, $|b_i\rangle = a_i \oplus b_i = p[i, i+1]$:



El nostre objectiu és poder obtenir el valor de cada interval $C[0,i]$ per a trobar els ròssecs, el qual es fa mitjançant una modificació de l'estructura de dades coneguda com a arbres binaris indexats (*Fenwick trees*). Aquestes estructures de dades permeten reconstruir informació de qualsevol subconjunt en temps logarítmic i espai lineal fent que cada element de la llista i emmagatzemi informació dels primers $i \wedge (-i)$ elements[6].

En el nostre cas, usarem aquest arbre per a emmagatzemar informació de tots els valors de $g[i,j]$ en els n qubits del resultat $|\mathbf{y}\rangle_n$ inicialitzats a 0. La informació que emmagatzema cadascun d'aquests índexs es correspon al subconjunt mostrat a la figura 12.

[6] $i \wedge (-i)$ calcula el bit de menor pes del nombre binari i. Per exemple si tenim el nombre $12 = 01100$, el seu negat en complement a 2 és $-12 = \neg 12 + 1 = 10011 + 1 = 10100$, de manera que el resultat és $01100 \wedge 10100 = 00100 = 4$. És a dir, és igual al valor del bit de menor pes de i.

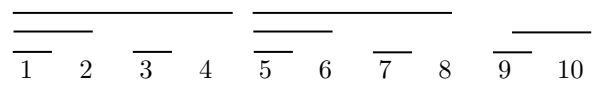


Figura 12: *Informació de cada node d'un Fenwick Tree. La línea directament a sobre de cada índex indica el subconjunt sobre el qual aquest node emmagatzema informació. Per exemple el node* 4 *emmagatzema informació dels nodes* 1 *a* 4. **Font**: Elaboració pròpia mitjançant Tikz.

Recordem que els índexs de q comencen per 0, de manera que caldria sumar 1 per a fer l'anàlogia amb l'arbre, però el funcionament és el mateix. Si tenim tots els valors $q[i,j]$ per i i j mostrats en la figura 12, podem obtenir el valor de qualsevol interval $q[0,i]$. Per exemple, si tinguéssim $n = 10$ i volguéssim trobar $q[0,6]$, sabent que la informació es troba en l'interval $[1,7]$ de l'arbre de Fenwick, el qual és la unió dels intervals $[1,4]$, $[5,6]$ i $[7,7]$, podem reconstruir el valor usant la formula d'unió esmentada anteriorment: $q[0,6] = q[0,3] * q[3,5] * q[5,6]$ on $*$ indica l'operació de combinació de conjunts també definida anteriorment. Si recordem, però, per a realitzar aquesta unió d'intervals de q, calia coneixement de certs valors de p. Més concretament, calia saber el valor de p dels intervals no inicials, ja que cal saber si aquests propaguen ròssecs o no, de manera que per a generar el Fenwick tree de q primer cal tenir-ne un per p.

A la pràctica no cal calcular tots els n valors de l'arbre de Fenwick de p, si no que tan sols aquells requerits per a calcular el de g. Com que aquest càlcul depèn únicament del nombre n de qubits del sistema, es pot computar amb un petit script, o de manera analítica: $2n - w(n) - \lfloor log\ n \rfloor$. Podem emmagatzemar n d'aquests valors de l'estructura de dades dins dels registre de l'operador $|\mathbf{b}\rangle_n$, de manera que el nombre de qubits addicionals per a emmagatzemar l'arbre és de $n - w(n) - \lfloor log\ n \rfloor$, tal i com s'havia mencionat abans.

En resum, hem de generar un arbre de fenwick de g, i per a fer això n'hem de generar un altre per p. Per $n = 4$, els intervals de g a emmagatzemar són:

$$g[0,1] \quad g[0,2] \quad g[2,3] \quad g[0,4]$$

Els quals es podem obtenir a partir dels que ja hem calculat de la manera:

$$\begin{aligned}
g[0,1] &= g[0,1] \\
g[0,2] &= g[0,1] * g[1,2] = (g[0,1] \wedge p[1,2]) \oplus g[1,2] \\
g[2,3] &= g[2,3] \\
g[0,4] &= g[0,2] * g[2,4] = (g[0,2] \wedge p[2,4]) \oplus g[2,4] \\
&= (g[0,2] \wedge p[2,4]) \oplus (g[2,3] \wedge p[3,4]) \oplus g[3,4]
\end{aligned}$$

I a partir d'aquests podrem reconstruir tots els ròssecs:

$$\begin{aligned}
g[0,1] &= g[0,1] \\
g[0,2] &= g[0,2] \\
g[0,3] &= g[0,2] * g[2,3] = (g[0,2] \wedge p[2,3]) \oplus g[2,3] \\
g[0,4] &= g[0,4]
\end{aligned}$$

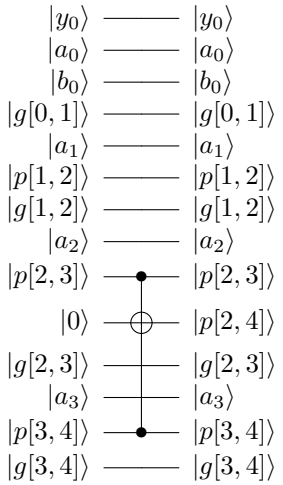
És a dir, per a trobar els ròssecs primer hem de calcular:

$$p[1,2] \quad p[2,3] \quad p[2,4] \quad p[3,4]$$

On $p[1,2]$, $p[2,3]$ i $p[3,4]$ ja han estat calculats sobre el registre de $|\mathbf{b}\rangle_n$, de manera que tan sols ens fa falta un únic qubit addicional a on calcularem $p[2,4]$, de nou, seguint la formula d'unió de p:

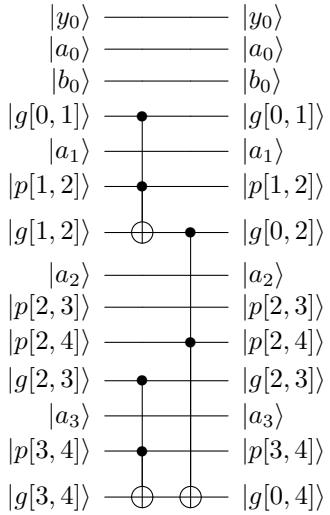
$$p[2,4] = p[2,3] * p[3,4] = p[2,3] \wedge p[3,4]$$

En aquest pas, doncs, calcularem tots aquests valors de p que ens calen, que en el cas de $n = 4$ tan sols és un:

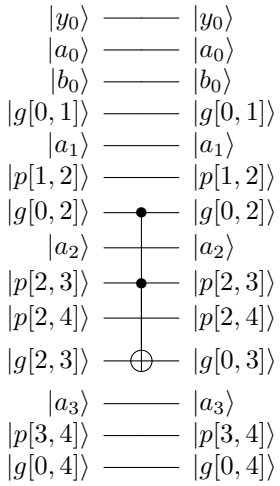


De manera general, els valors de p es calculen mitjançant el seu propi arbre de Fenwick. El nombre de qubits addicionals vé directament determinat pel nombre de termes de p addicionals als $p[i, i+1]$ que cal calcular i emmagatzemar per a calcular l'arbre de g.

Ara que tenim aquests valors de p, podem calcular els valors de g:

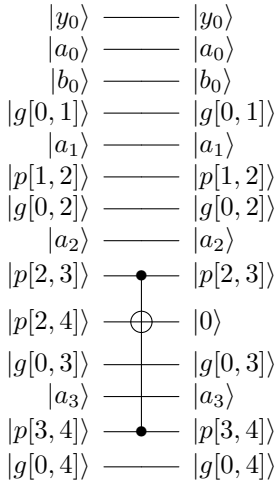


Un cop tenim construït l'arbre de Fenwick de g, podem obtenir tots els ròssecs tal i com s'ha mencionat abans. En aquest cas ja tenim la majoria de valors calculats, així que tan sols cal calculat $g[0,3] = g[0,2] * g[2,3] = (g[0,2] \wedge p[2,3]) \oplus g[2,3]$:

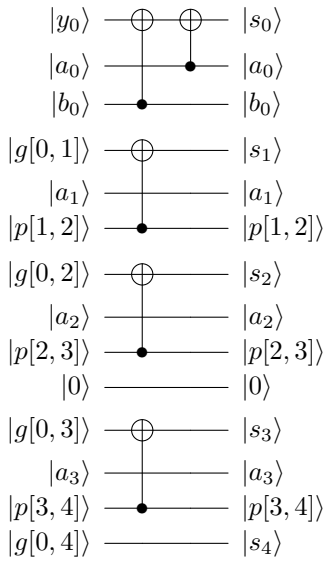


Ara ja tenim els ròssecs a cada qubit del registre de resultat $y_{i+1} = c_{i+1} = g[0, i+1]$ per $0 \geq i < n$, de manera que tan sols cal reiniciar els registres $|\mathbf{b}\rangle_n$ i els qubits temporals.

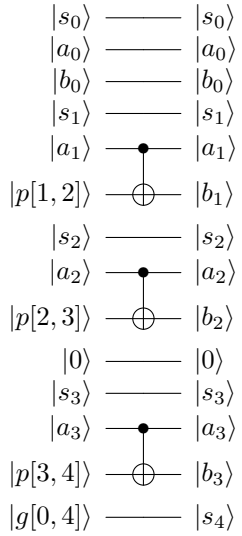
Començem esborrant els qubits auxiliars simplement aplicant les portes del tercer pas a la inversa, ja que aquestes són les úniques que els modifiquen:



Com que $p[i, i+1] = a_i \oplus b_i$, $g[0, i] = c_i$ i $s_i = a_i \oplus b_i \oplus c_i$, podem calcular cada bit de la suma com $s_i = p[i, i+1] \oplus g[0, i]$ amb una **CNOT** sobre cada parell $|b_i\rangle$, $|y_i\rangle$ per $0 < i < n$. També podem calcular $s_0 = a_0 \oplus b_0$ ja que hem assumit $c_0 = 0$:



I finalment desfem els canvis al registre $|\mathbf{b}\rangle_n$ realitzant la mateixa operació del segon pas a la inversa:



El qual ens dona el resultat esperat, ja que no modifica els operadors, deixa el resultat al registres correctes i no modifica els qubits auxiliars.

El disseny complet del circuit es mostra a la figura 13, obtingut unint els passos anteriors.

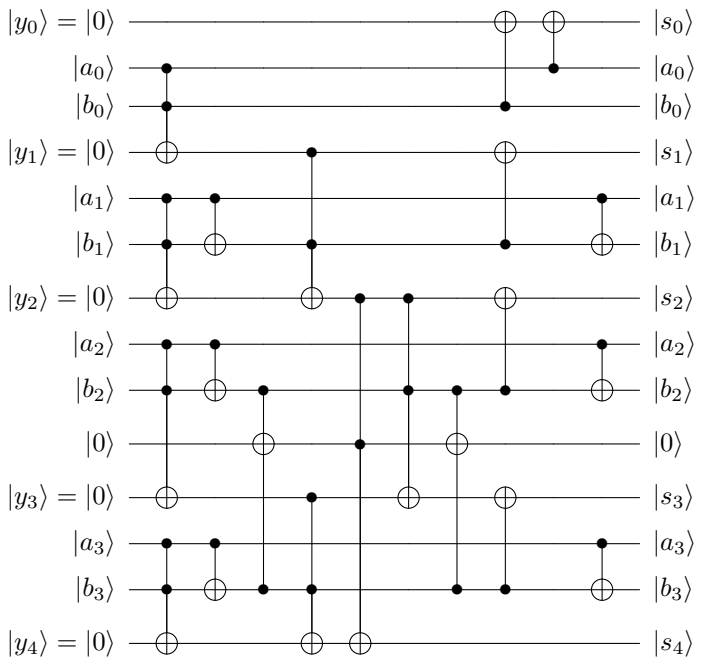


Figura 13: *Implementació Carry-Lookahead de l'operador ADD per a sumar dos registres* $|\boldsymbol{a}\rangle_4$ *i* $|\boldsymbol{b}\rangle_4$ *sobre un registre pel resultat* $|\boldsymbol{y}\rangle_4 |z\rangle$ *usant un registre temporal de* $O(n)$ *qubits.* **Font:** Elaboració pròpia mitjançant Q-Circuit. Basat en el disseny de [9].

Tal i com hem fet amb el disseny anterior, podem modificar trivialment el circuit per a calcular la suma modular, ja que en cap moment es fa servir el qubit addicional per a emmagatzemar un resultat intermedi. Aquesta modificació es mostra a la figura 14.

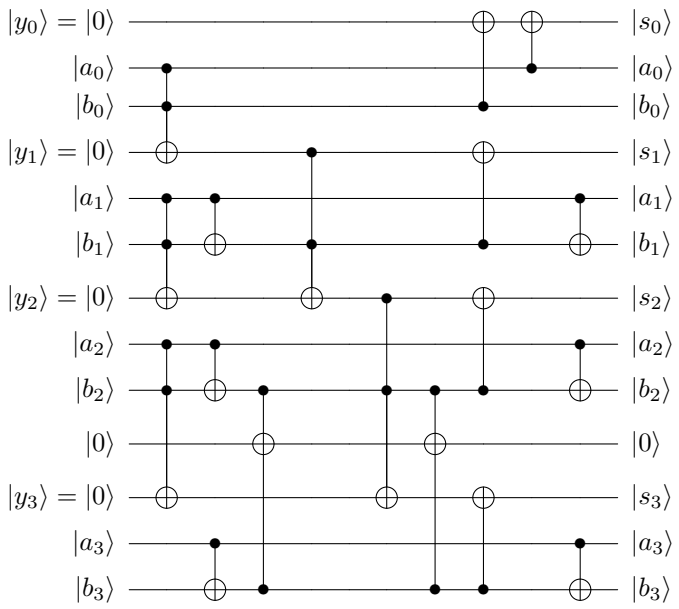
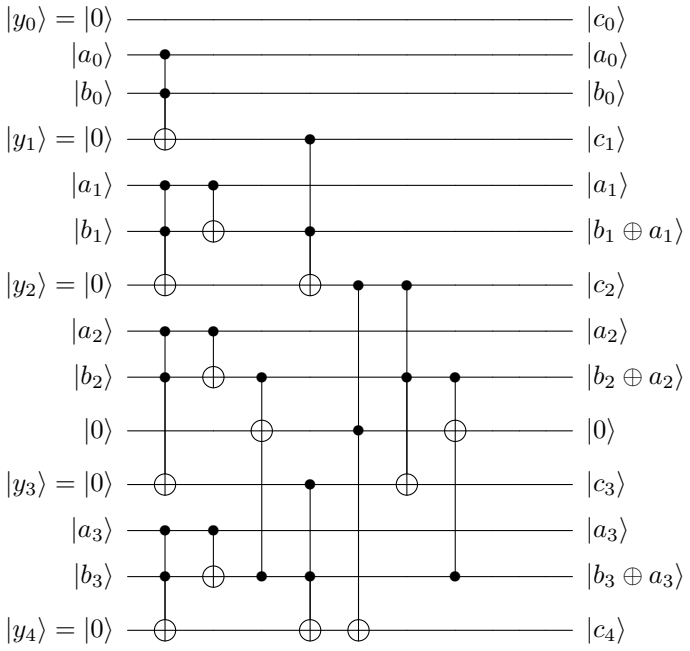


Figura 14: *Implementació Carry-Lookahead de l'operador ADD per a sumar dos registres* $|\boldsymbol{a}\rangle_4$ *i* $|\boldsymbol{b}\rangle_4$ *sobre un registre pel resultat* $|\boldsymbol{y}\rangle_4$ *usant un registre temporal de* $O(n)$ *qubits.* **Font:** Elaboració pròpia mitjançant Q-Circuit. Basat en el disseny de [9].

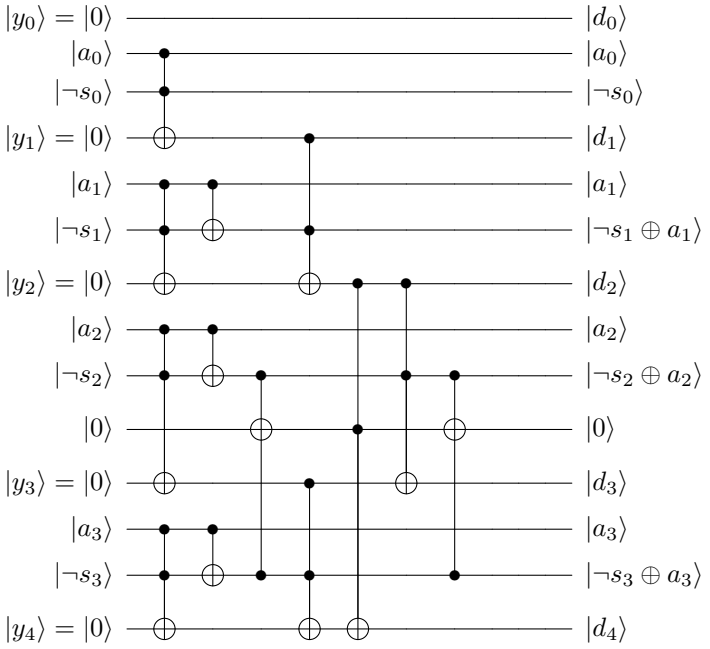
Aquests circuits permeten realitzar la suma sobre un registre addicional, però si es vol que aquesta suma es calculi sobre un dels operands cal expandir el circuit per a reiniciar els estats de n qubits addicionals. Partint del circuit de suma no modular sobre un registre extra mostrat a la figura 13, eliminem les portes **CNOT** que calculen els bits de la suma sobre els ròssecs emmagatzemats en el registre addicional i les que retornen $|\mathbf{b}\rangle_{n+1}$ al seu estat original:



(1)

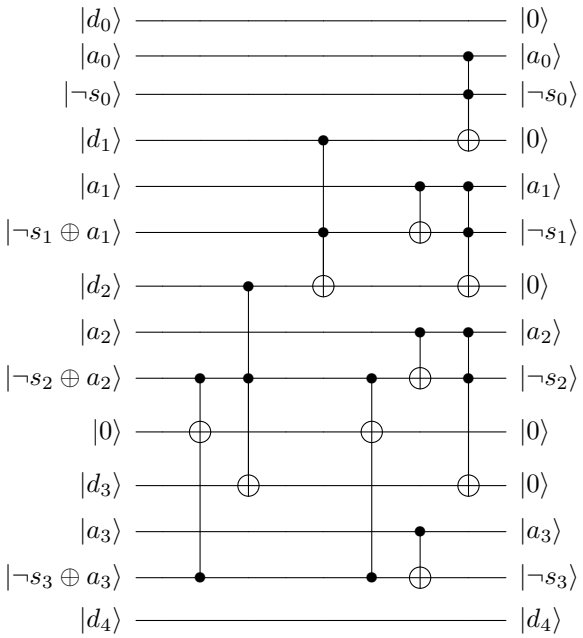
En aquest estat, cada terna de qubits inicials $|a_i\rangle$,$|b_i\rangle$,$|y_i\rangle$ conté $|a_i\rangle$,$|b_i \oplus a_i\rangle$,$|c_i\rangle$ respectivament. Sabem que podríem calcular la suma sobre el segon operand simplement amb l'aplicació d'una **CNOT** que transformi $|b_i \oplus a_i\rangle$ en $|b_i \oplus a_i \oplus c_i\rangle = |s_i\rangle$, però en realitzar aquesta acció ens queden els valors del registre $|\mathbf{y}\rangle_n$, els quals s'usaven abans pel resultat però ara han de ser reiniciats a $|0\rangle$.

Assumim que volem calcular la suma $a + \neg s$ on $s = a + b$. Amb el nostre circuit, obtindríem un conjunt de n ternes amb els estats $|a_i\rangle$,$|s_i \oplus a_i\rangle$,$|d_i\rangle$, on d és el vector de ròssecs de la suma, els quals compleixen $a + \neg s \equiv a \oplus \neg s \oplus d$. Aquesta suma pot semblar arbitrària en aquest moment, però serà vital per a obtenir el circuit de suma en temps logarítmic sobre un operand. Per context, el circuit i els valors inicials i finals serien:



Poder realitzar aquesta operació no és pas gaire útil ja que requereix tenir el valor de la suma, però gràcies a la reversibilitat dels computadors quàntics podem invertir aquest circuit. El diagrama anterior transforma l'estat $|\mathbf{a}\rangle_n |\neg\mathbf{s}\rangle_n |\mathbf{0}\rangle_n$, a $|\mathbf{a}\rangle_n |\neg\mathbf{s} \oplus \mathbf{a}\rangle_n |\mathbf{d}\rangle_n$, de manera que si aconseguim arribar a aquest segon estat, podrem aplicar el circuit invers per a calcular la suma i reiniciar el registre addicional.

Aquest circuit no és idèntic a l'anterior ja que no volem eliminar els valor de $|y_4\rangle$, doncs aquest conté el valor del bit de ròssec addicional de la suma no modular, de manera que el circuit a aplicar serà l'invers del de suma de registres de $n - 1$ qubits:



(2)

No s'han eliminat totes les portes possibles per a mostrar millor com es veuria el disseny per a registres majors, però com que aquest és el circuit invers de suma de 3 qubits, no cal realitzar pràcticament cap porta per a calcular l'arbre de Fenwick.

Gràcies al circuit parcial 1 podem transformar cada terna inicial $|a_i\rangle$,$|b_i\rangle$,$|0\rangle$ a $|a_i\rangle$,$|b_i \oplus a_i\rangle$,$|c_i\rangle$, i amb 2 podem transformar cada terna $|a_i\rangle$,$|\neg s_i \oplus a_i\rangle$, $|d_i\rangle$ a $|a_i\rangle$,$|\neg s_i\rangle$,$|0\rangle$, el qual és pràcticament el resultat desitjat.

Cal, doncs, trobar una manera de transformar cada $|a_i\rangle$,$|b_i \oplus a_i\rangle$,$|c_i\rangle$ a $|a_i\rangle$,$|\neg s_i \oplus a_i\rangle$, $|d_i\rangle$. El primer qubit és el mateix, de manera que no cal realitzar cap transformació. El valor desitjat del segon qubit pot ser reescrit:

$$\begin{gathered}\neg s_i \oplus a_i \\ s_i \oplus 1 \oplus a_i \\ a_i \oplus b_i \oplus c_i \oplus a_i \oplus 1\end{gathered}$$

De manera que tan sols cal afegir els termes $\oplus c_i \oplus a_i \oplus 1$, els quals són calculats fàcilment mitjançant dues **CNOT** i una $\mathbf{X}$[7].

Pel que fa al tercer qubit, pot semblar impossible transformar $|c_i\rangle$ a $|d_i\rangle$ ja que no hi ha cap relació aparent, però la podem inferir a partir de les definicions dels vectors de ròssecs c i d.

Sabem que $x + \neg x + 1 \equiv 0 \ mod\ 2^n$, de manera que:

$$a + \neg s \equiv a - s - 1 \equiv a - a - b - 1 \equiv b - 1 \equiv \neg b$$

A partir d'aquesta igualtat, i recordant que c i d són els ròssecs de les sumes $a + b$ i $a + \neg s$ respec-

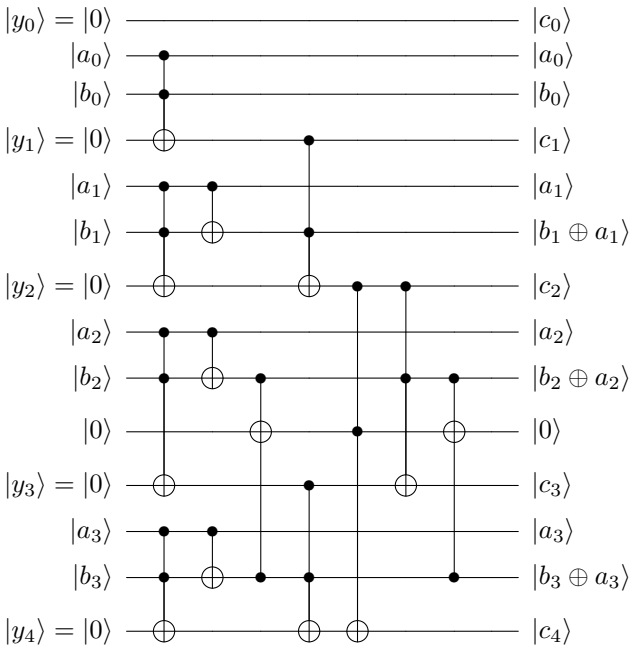
[7] També es pot unir $\oplus 1$ amb qualsevol dels altres dos per a estalviar-se la $\mathbf{X}$ i canviar una **CNOT** per una **CNOT0**

tivament, tenim:

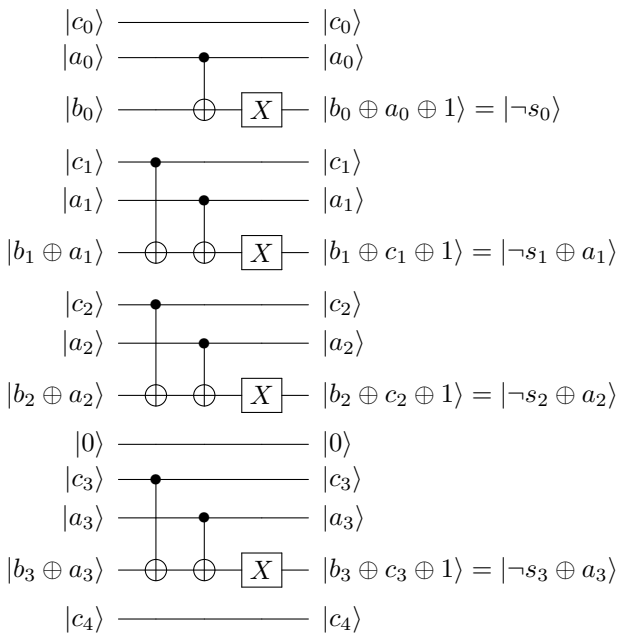
$$\begin{aligned} a + \neg s &\equiv \neg b \\ a \oplus \neg s \oplus d &\equiv \neg b \\ a \oplus s \oplus (-1) \oplus d &\equiv b \oplus (-1) \\ a \oplus (a \oplus b \oplus c) \oplus (-1) \oplus d &\equiv b \oplus (-1) \\ c \oplus d &\equiv 0 \\ c &\equiv d \end{aligned}$$

És a dir, no cal realitzar cap transformació.

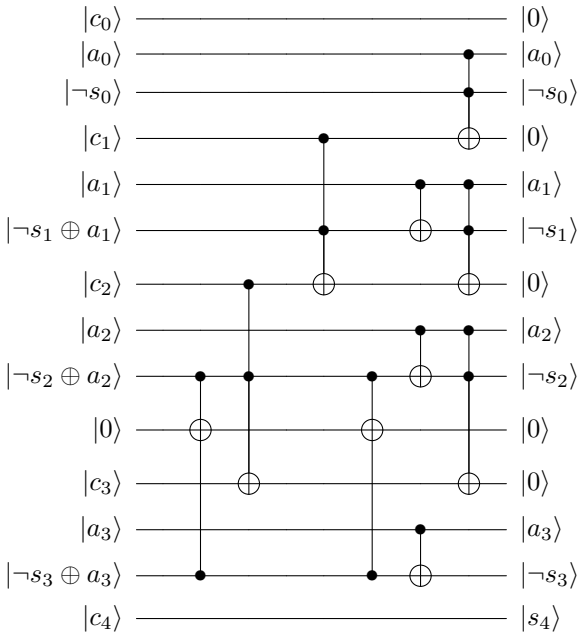
Amb aquesta informació, podem construir el circuit de suma sobre un operand. Partint del circuit parcial 1 que transforma $|a_i\rangle$,$|b_i\rangle$,$|0\rangle$ a $|a_i\rangle$,$|b_i \oplus a_i\rangle$,$|c_i\rangle$:



Transformem cada terna per a poder usar el segon circuit parcial. Com s'ha mencionat abans, no cal modificar ni $|a_i\rangle$ ni $|c_i\rangle$, de manera que tan sols apliquem una **CNOT** sobre $|c_i\rangle$ i $|b_i\rangle$, una altra **CNOT** sobre $|a_i\rangle$ i $|b_i\rangle$ i finalment una **X** sobre $|b_i\rangle$:



I finalment apliquem el circuit parcial 2 per a transformar cada $|a_i\rangle$,$|\neg s_i \oplus a_i\rangle$, $|d_i\rangle$ a $|a_i\rangle$,$|\neg s_i\rangle$,$|0\rangle$:



Podem negar cada qubit del resultat per a obtenir la suma desitjada. El circuit complet, obtingut unint les tres parts anteriors, es mostra a la figura 15.

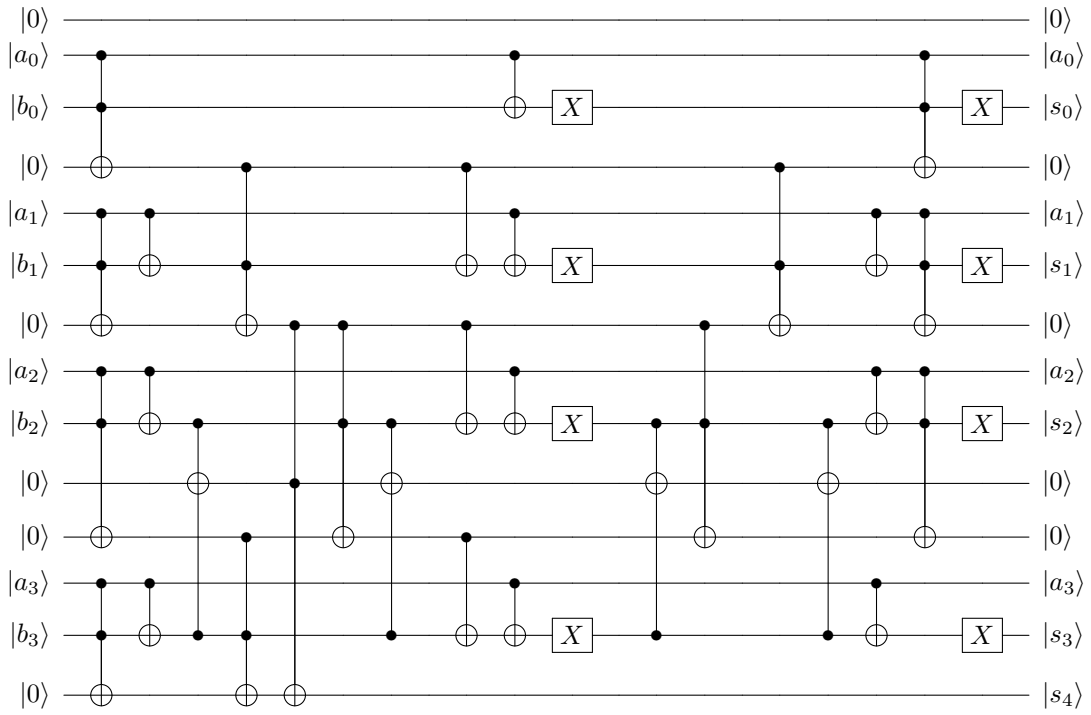


Figura 15: *Implementació Carry-Lookahead de l'operador ADD no modular per a sumar dos registres* $|\boldsymbol{a}\rangle_4$ *i* $|\boldsymbol{b}\rangle_4$ *sobre aquest segon operand amb un qubit addicional pel ròssec usant un registre temporal de* $O(n)$ *qubits.* **Font:** Elaboració pròpia mitjançant Q-Circuit. Basat en el disseny de [9].

Tal i com s'ha fet amb la versió que actua sobre un registre addicional, podem modificar aquest circuit de manera trivial per a calcular la suma modular sobre el segon operand ja que el qubit del ròssec mai s'usa per a resultats intermedis. Aquesta modificació es mostra a la figura 16.

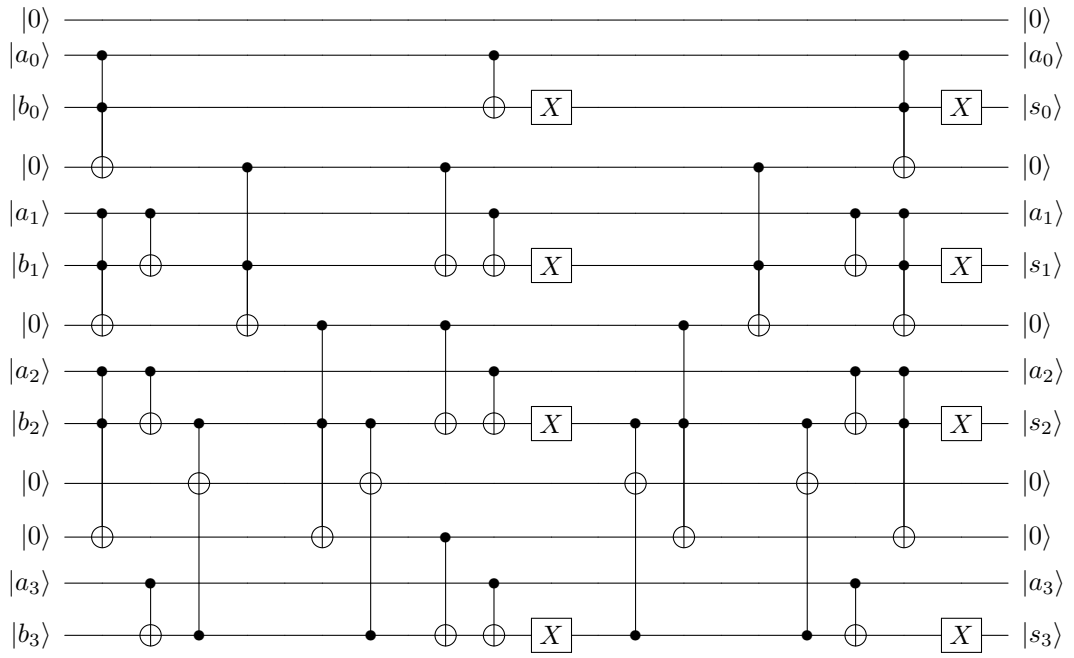


Figura 16: *Implementació Carry-Lookahead de l'operador ADD modular per a sumar dos registres* $|\boldsymbol{a}\rangle_4$ *i* $|\boldsymbol{b}\rangle_4$ *sobre aquest segon operand usant un registre temporal de* $O(n)$ *qubits.* **Font:** Elaboració pròpia mitjançant Q-Circuit. Basat en el disseny de [9].

Complexitat

La profunditat del circuit de suma no modular sobre un registre addicional és logarítmica respecte el tamany n dels registres, de $\lfloor log(n) \rfloor + \lfloor log(\frac{n}{3}) \rfloor + 7$ per $n > 3$ i menor en cas contrari ja que no cal computar res per $p[i, j]$. Aquest valor és aproximadament el doble per a la suma sobre el segon operand, però la profunditat logarítmica es manté.

El nombre de qubits addicionals requerits és $n - w(n) - \lfloor log\ n \rfloor$, més un pel ròssec en les versions no modulars.

La quantitat d'operadors del conjunt bàsic necessaris per a obtenir-ne una implementació completa es mostren a la taula 4.

	Conjunt bàsic		
	X	CNOT	Toffoli
Registre addicional	0	$3n - 1$	$5n - 3w(n) - 3\lfloor log(n) \rfloor - 1$
Segon operand	$2n$	$4n - 1$	$9n - 6w(n) - 6\lfloor log(n) \rfloor - 2$

Taula 4: *Nombre de portes de la implementació Carry-Lookeahead de ADD. La taula mostra el nombre de portes del conjunt bàsic que calen per a construir el circuit de suma modular i no modular sobre un registre addicional o sobre un operand des de zero.* **Font:** Elaboració pròpia.

3.2.4 Suma modular amb la Transformada de Fourier Quàntica

De manera informal, el terme Transformada de Fourier es refereix a un conjunt de transformacions matemàtiques que associen funcions o conjunts amb les seves contrapartides en l'espai de freqüències. El llibre [18] en dona una explicació extensiva, però en aquesta secció en donarem una petita introducció ja que és necessari per a poder entendre el funcionament de la seva versió quàntica.

La transformada de Fourier discreta transforma un conjunt de nombres complexos $x_0, x_1, \ldots, x_{n-1}$ en un altre $y_0, y_1, \ldots, y_{n-1}$ de manera que per tot k es compleixi:

$$y_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j e^{2\pi i \frac{j}{n} k}$$

L'inversa d'aquesta transformació és similar a la funció, de manera que es pot entendre alternativament com a una transformació per a obtenir els coeficients y_k que compleixin:

$$x_j = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} y_k e^{-2\pi i \frac{k}{n} j}$$

La transformada de Fourier quàntica és un operador que actua sobre un sistema de n qubits i realitza la transformació següent:

$$\sum_{j=0}^{n-1} x_j \left|\mathbf{j}\right\rangle_n \rightarrow \sum_{k=0}^{n-1} y_k \left|\mathbf{k}\right\rangle_n$$

On els coeficients del resultat y_k són el producte de la transformada de Fourier discreta dels coeficients de l'estat inicial x_j.

Una definició alternativa però sovint més útil és la següent, on el resultat ha estat factoritzat en els estats individuals, facilitant l'enteniment:[8]

$$\left|x_1, x_2 \ldots x_n\right\rangle \rightarrow \frac{(\left|0\right\rangle + e^{2\pi i 0.j_n} \left|1\right\rangle)(\left|0\right\rangle + e^{2\pi i 0.j_{n-1} j_n} \left|1\right\rangle) \ldots (\left|0\right\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n} \left|1\right\rangle)}{2^{\frac{n}{2}}}$$

És a dir, la transformada quàntica de Fourier transforma cada estat en una superposició on el qubit k té una fase relativa igual a 2π per la part fraccionària de $\frac{n}{2^k}$.

Implementacions d'aquest operador encara són estudiats actualment, però presentarem el disseny mostrat en el llibre *Quantum Computation and Quantum Information* [19].

Assumirem la capacitat de poder aplicar portes de la forma:

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

Per $k \in \mathbb{N}$. Com s'ha explicat a la recció d'introducció a la computació quàntica, podem implementar qualsevol operador condicionat mitjançant **CNOT** i portes d'un qubit[9], de manera que assumirem també la capacitat d'aplicar aquestes portes R_k condicionades a un altre qubit. La figura 17 mostra una implementació de la transformada de Fourier quàntica sobre n qubits.

La transformada de Fourier quàntica per a n qubits consta de $O(n^2)$ portes del conjunt $\{R_k, H\}$ i té una profunditat de $O(n)$.

[8] $0.x_1 x_2 \ldots x_n$ és igual a la fracció en base dos equivalent a $\sum_{i=1}^{n} \frac{x_i}{2^i}$, on $x_1 x_2 \ldots x_n$ és un natural en base 2.
[9] Desenvolupat més en profunditat a [20] i [21]

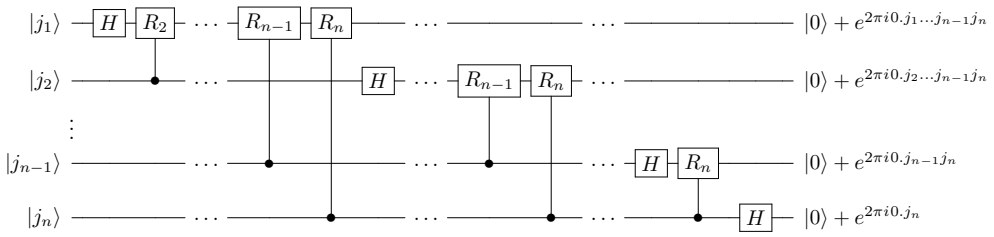


Figura 17: *Implementació de la transformada de Fourier quàntica. En aquest disseny els bits estan indexats de major a menor pes: $a = a_1 a_2 \cdots a_n$.* **Font:** Elaboració pròpia mitjançant Q-Circuit. Basat en el disseny de [19].

L'estudi de l'aplicació d'aquest operador en el càlcul aritmètic sobre qubits va començar amb la publicació [4], on es presenta un circuit per a realitzar la suma modular sobre el segon operador. Estudis posteriors com [14] o [15] expandeixen els mateixos conceptes i generalitzen el circuit per a realitzar càlculs més complexos amb menor cost espacial (qubits addicionals) o temporal (profunditat o nombre de portes).

Les implementacions de l'operació de suma d'enters són pràcticament idèntiques en tots els documents, però farem més referència al circuit de la publicació [15] ja que és capaç d'operar sobre enters i naturals i de computar sumes modulars i no modulars amb petites modificacions.

El funcionament del circuit es basa en el fet que aplicar una porta R_k a l'estat $|0\rangle + e^{2\pi i x}|1\rangle$ és equivalent a sumar $2\pi\frac{1}{2^k}$ a la fase.

$$\begin{aligned} R_k(|0\rangle + e^{2\pi i x}) &= R_k|0\rangle + e^{2\pi i x} R_k|1\rangle \\ &= |0\rangle + e^{2\pi i x} e^{2\pi i/2^k}|1\rangle \\ &= |0\rangle + e^{2\pi i (x+\frac{1}{2^k})}|1\rangle \end{aligned}$$

Si s'entén x com un decimal en base 2 de la forma $0.x_1x_2\dots x_n = \frac{x_1x_2\dots x_n}{2^n}$, sumar $\frac{1}{2^k}$ a x és igual a $\frac{x_1x_2\dots x_n + 2^{n-k}}{2^n}$, o equivalentment, és igual a sumar 1 al bit k començant per l'esquerra del numerador $x_1x_2\dots x_n$, ignorant la part entera. Aquesta suma d'un bit és equivalent a la que realitza el circuit de suma de naturals o enters normal, de manera que podem aprofitar aquesta propietat per a operar sobre bits de nombres en base dos sempre i quan els tinguem en la seva forma transformada per l'operador **QFT**.

Si apliquem aquesta porta a tots els qubits de la transformada de Fourier d'una superposició $R_k\Psi(a)$, com s'ha dit abans, obtenim la superposició corresponent a $|\Phi(a + 2^{n-k})\rangle$. Podem aplicar n operacions d'aquest tipus sobre $\Phi(a)$ condicionades als bits de $b = b_n b_{n-1}\dots b_1$ per a obtenir $|\Phi(a + \sum_{i=1}^{n} b_i 2^i)\rangle = |\Phi(a + b \ mod \ 2^n)\rangle$.

El circuit que realitza aquest conjunt d'operacions és equivalent al que calcula la transformada de Fourier tan sols usant els qubits de l'altre operand com a control, el qual es mostra a la figura 18.

El disseny complet, doncs, s'obté afegint una porta **QFT** sobre el registre $|\mathbf{a}\rangle_n$ abans del circuit 18 que transformi $|\mathbf{a}\rangle_n$ en $|\mathbf{\Phi(a)}\rangle_n$ i posteriorment la seva inversa per a transformar $|\mathbf{\Phi(a+b)}\rangle_n$ en $|\mathbf{a+b}\rangle_n$.

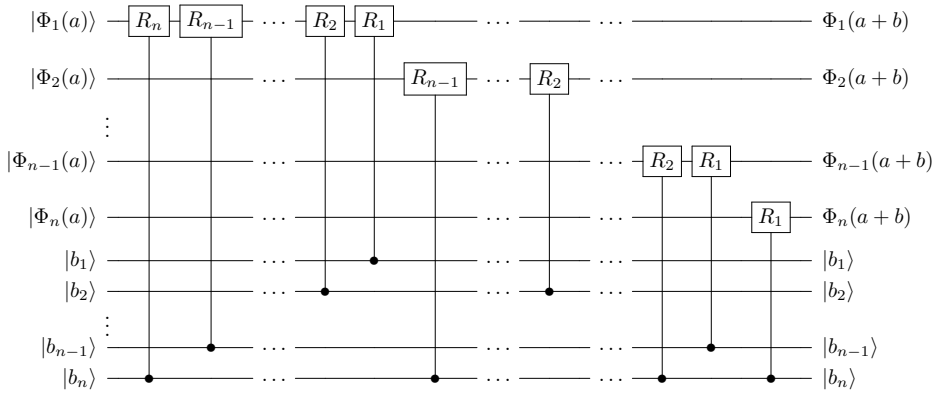


Figura 18: *Implementació de l'operador ADD modular sobre un dels operands mitjançant la transformada de Fourier quàntica. En aquest disseny els bits estan indexats de major a menor pes:* $a = a_1 a_2 \cdots a_n$. **Font:** Elaboració pròpia mitjançant Q-Circuit. Basat en el disseny de [15].

Aquest circuit calcula la suma mòdul 2^n ja que tan sols es té en compte la part fraccionària de les fases, però es pot modificar de manera trivial per a calcular la suma no modular, és a dir, realitzar el càlcul del ròssec c_{n+1}. Tan sols cal augmentar el tamany del registre resultat amb un nou qubit inicialitzat a $|0\rangle$ per a suma de naturals, el qual serà transformat per la **QFT** a $|0\rangle + e^{2\pi i 0.0 a_1 a_2 \ldots a_n}$. El conjunt de portes R_k a aplicar sobre aquest nou qubit és fàcilment deduïble a partir d'aquesta representació, el qual serà igual al conjunt aplicat a $|\Phi_1(a)\rangle$ augmentant en 1 totes les k, tal i com s mostra a la figura 19.

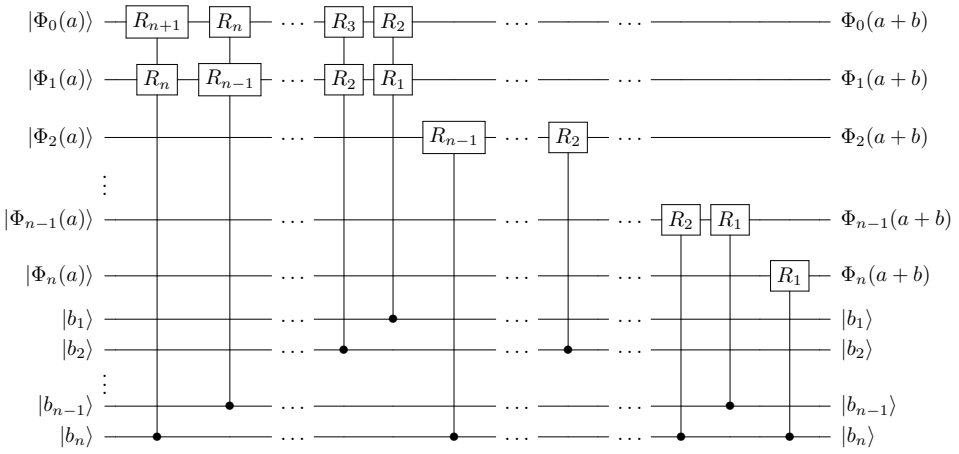


Figura 19: *Implementació de l'operador ADD no modular sobre un dels operands mitjançant la transformada de Fourier quàntica. En aquest disseny els bits estan indexats de major a menor pes:* $a = a_1 a_2 \cdots a_n$. **Font:** Elaboració pròpia mitjançant Q-Circuit. Basat en el disseny de [15].

La profunditat del circuit és lineal respecte el tamany n dels registres, de l'ordre de $\approx 3n$ ja que cal aplicar la **QFT**, el còmput de la suma sobre aquesta transformació, i la inversa, totes de profunditat lineal.

El nombre de qubits addicionals requerits és 0, més un pel ròssec.

La quantitat d'operadors del conjunt bàsic (i R_k) necessaris per a obtenir-ne una implementació completa es mostren a la taula 5.

	Conjunt bàsic			
Porta	X	CNOT	Toffoli	R_k
Quantitat	0	0	0	$3(n + \frac{n(n+1)}{2})$

Taula 5: *Nombre de portes de la implementació Carry-Lookeahead de ADD. La taula mostra el nombre de portes del conjunt bàsic que calen per a construir el circuit des de zero.* **Font:** Elaboració pròpia.

Tot i que el nombre elevat de portes pugui semblar indicar que aquest circuit de suma és inferior a tots els anteriors, la realitat és que treballar amb les transformades de Fourier dels operadors sols ser convenient en altres situacions, de manera que en certs casos es podria dir que el nombre de portes real per a realitzar la suma és de $n + \frac{n(n+1)}{2}$, ja que no cal aplicar els operadors **QFT** i $\mathbf{QFT}^{-1}$.

Un altre avantatge és el fet de que en cap moment es modifiquen els qubits del segon operand, de manera que aquest disseny és ideal per a sumar o restar valors constants sense usar qubits addicionals.

3.3 Negació d'enters

Tot i semblar trivial, realitzar una negació de nombres codificats en bits no sol ser tan simple com "afegir un signe -". Diverses representacions de nombres tenen diferents maneres de realitzar aquesta operació, així que de moment ens limitarem a tractar d'obtenir una implementació de l'operador **NEG** per a enters en Ca2 de n qubits, amb prototip mostrat a la figura 20.

Figura 20: *Prototips de l'operador NEG per a negar un enter codificat en n qubits.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Per a nombres enters representats en complement a 2, el negat d'un nombre x de n bits és un nombre x' tal que:

$$x + x' \equiv 0 \ mod \ 2^n$$

El qual, per x i x' representats amb bits, és equivalent a:

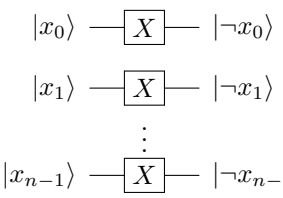
$$x + x' = 2^n$$

$$x + x' = x + \neg x + 1$$

$$x' = \neg x + 1$$

On $\neg x$ denota la negació bit a bt de x.

L'operador per a negar un enter és, doncs, format per una negació bit a bit de l'operand seguida de la suma d'una unitat. El primer pas per a calcular x' és ben senzill, tan sols cal aplicar una **X** a tots els qubits de $|\mathbf{x}\rangle_n$:

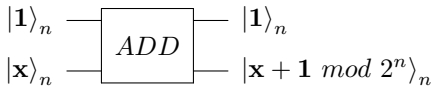


El segon pas consisteix en l'addició d'una unitat al registre, tasca que serà efectuada per l'operador **ADD1**, amb prototip mostrat a la figura 21.

Figura 21: *Prototip de l'operador ADD1 per a sumar 1 a un registre de n qubits.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

És obvi, doncs, que qualsevol implementació d'aquest operador **ADD1** permet la negació d'enters, però la complexitat d'aquesta negació dependrà en la seva majoria de la complexitat de la suma d'una unitat.

Aquesta suma es pot realitzar trivialment mitjançant qualsevol operador **ADD** estudiat anteriorment, on un dels operands pren el valor de $|\mathbf{1}\rangle_n$:



Però això sembla més complex del que s'esperaria de la suma d'una unitat, en requerir n qubits addicionals. Solucions més eficients es poden obtenir mitjançant modificacions dels blocs **ADD** estudiats anteriorment.

3.3.1 Suma d'una unitat amb Ripple-Carry

El mètode de Ripple-Carry es basa en realitzar la suma tradicional emmagatzemant els resultats intermitjos sobre els qubits dels operands. En el nostre cas, però, no podem emmagatzemar aquests resultats ja que tan sols disposem de la meitat de qubits, però si assumim que l'altra meitat pren un valor constant de $000\ldots1$, podem calcular cada ròssec en funció de l'anterior com a:

$$c_{i+1} = a_i \wedge c_i$$

Amb el case base $c_0 = 1$.

I cada qubit del resultat com:

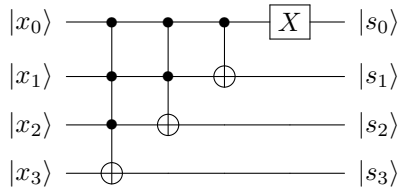
$$s_i = a_i \oplus c_i$$

Amb el cas base $s_0 = \neg a_0$.

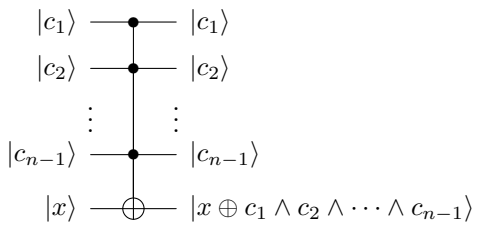
Gràcies a aquestes, podem obtenir un expressió per s_i en termes a:

$$s_i = a_i \oplus (a_{i-1} \wedge c_{i-1}) = a_i \oplus (a_{i-1} \wedge (a_{i-3} \wedge c_{i-3})) = \bigwedge_{i=0}^{n} a_{n-1} \wedge c_0 = \bigwedge_{i=0}^{n-1} a_i$$

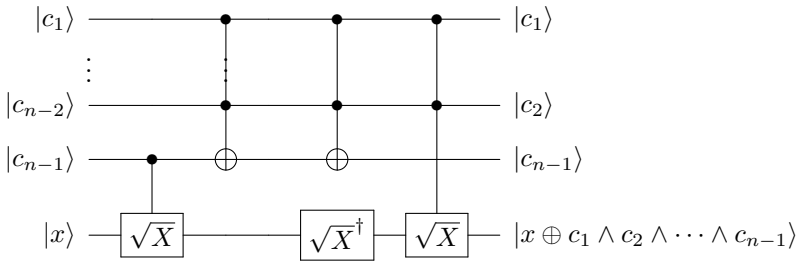
És a dir, es pot realitzar l'operació amb n portes **X** condicionades de 1 a n qubits:



Aquestes portes d'anomenen **Toffoli-n**, i apliquen una **X** condicionada a $n-1$ qubits:

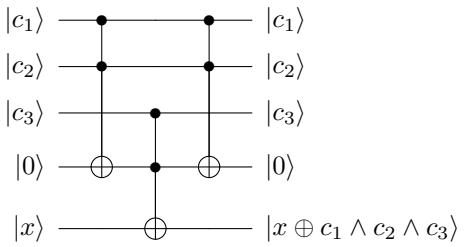


Quan s'usa el mètode Ripple-Carry sovint es desitja optimitzar el nombre de qubits addicionals, de manera que idealment es podrien implementar aquestes **Toffoli-n** sense usar-ne cap. Estudis com [22] en donen una definició recursiva en funció de portes **CNOT**, $\mathbf{X}^{2^{-k}}$[10], i **Toffoli-n-1**:

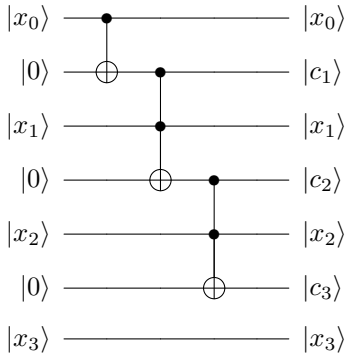


Una implementació alternativa més irregular es detalla a [23], però aquesta té una millor profunditat de $O(n)$ també sense qubits auxiliars.

Amb qualsevol d'aquestes dues implementacions de la porta **Toffoli-n** podem obtenir circuits de suma d'una unitat sense qubits auxiliars, però si ens podem permetre $O(n)$ qubits auxiliars, podem usar un mètode alternatiu per a dissenyar aquests operadors:

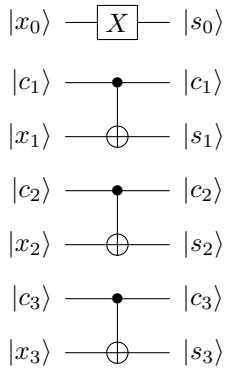


Gràcies al fet que podem emmagatzemar resultats entremitjos, ja no cal realitzar les n **Toffoli-k** per separat, de manera que podem usar un esquema molt més semblant al de suma de dos operands amb Ripple-Carry. Primerament podem calcular tots els ròssecs $c_{i+1} = c_i \wedge x_i$ mitjançant portes **Toffoli**:

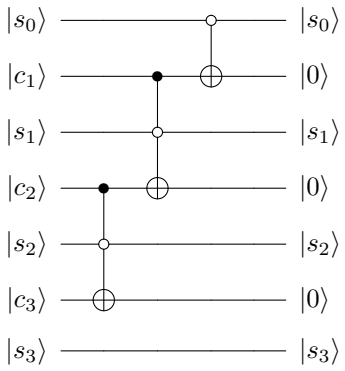


Seguidament calculem cada valor de la suma sobre l'operand $s_i = x_i \oplus c_i$ mitjançant portes **CNOT** i una **X** pel primer $s_0 = x_i \oplus 1 = \neg x_i$:

[10] Per qualsevol operador **Q**, $\sqrt{\mathbf{Q}}$ és definida com aquella porta tal que aplicada dues vegades és igual a **Q**. Anàlogament, la porta $\mathbf{X}^{2^{-k}}$ és defineix com un operador tal que aplicar-lo 2^k vegades és equivalent a aplicar **X**.



Tan sols queda eliminar els c_i residuals dels registres temporals. Podem assolir això definint els ròssecs en funció dels anteriors i dels resultats: $c_i = x_{i-1} \wedge c_{i-1} = (s_{i-1} \oplus c_{i-1}) \wedge c_{i-1} = c_{i-1} \wedge \neg s_{i-1}$:



En aquest disseny s'han marcat alguns qubits de control amb el punt buit, indicant que l'estat d'quest per a activar la porta ha de ser $|0\rangle$ em comptes de $|1\rangle$. És equivalent a col·locar una **X** abans i després d'un control normal, però generalment s'assumeix que si un computador quàntic pot implementar una operació control·lada a un qubit en l'estat $|1\rangle$, també ho pot fer per l'oposat.

El disseny complet de suma d'una unitat per Ripple-Carry es mostra a la figura 22, obtingui unint les tres parts detallades en aquesta secció.

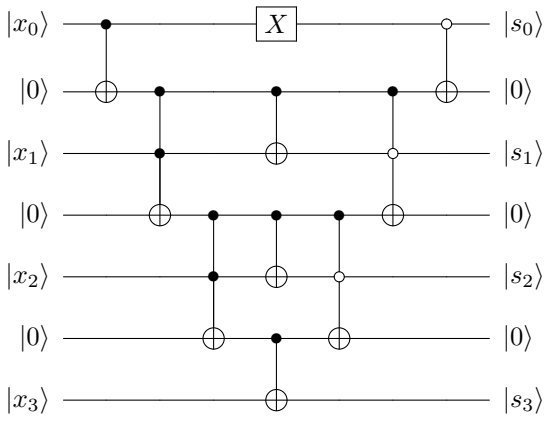


Figura 22: *Implementació Ripple-Carry de l'operador ADD1 per a sumar una unitat a enters de 4 qubits.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Complexitat

La profunditat del circuit és lineal respecte el tamany n dels registres, de l'ordre de $\approx 2n$ per al circuit donat.

El nombre de qubits addicionals requerits és $\approx n$ pel circuit donat, o 0 si s'usen les implementacions de **Toffoli-k** adients.

La quantitat d'operadors del conjunt bàsic necessaris per a obtenir-ne una implementació completa es mostren a la taula 6.

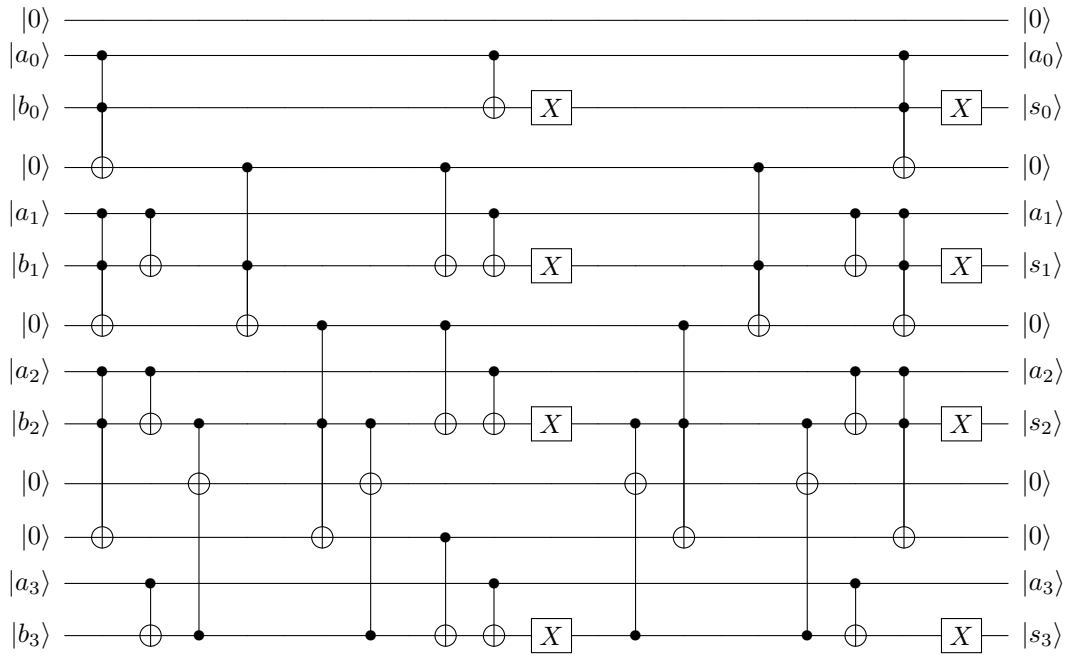
	Conjunt bàsic		
	X	CNOT	Toffoli
Amb qubits	1	$n+1$	$2(n-2)$

Taula 6: *Nombre de portes de la implementació Ripple-Carry amb qubits temporals de ADD1. La taula mostra el nombre de portes del conjunt bàsic que calen per a construir el circuit des de zero.* **Font:** Elaboració pròpia.

3.3.2 Suma d'una unitat amb Carry-Lookahead

Si la profunditat del circuit fos la mètrica principal a optimitzar, i ens pugéssim permetre usar qubits addicionals, es podria modificar el circuit de suma mitjançant Carry-Lookahead detallat anteriorment per a realitzar aquesta operació.

Recordem que el circuit que realitza aquesta operació, per registres de 4 qubits, modular, i sobre un operand, pren la forma:



Aquest circuit no tan sols manté el valor del operand a la sortida, sinó que mai en modifica el seu estat, de manera que tan sols és usat com a control en diverses operacions. Aquesta propietat ens permet redissenyar el circuit modificant tots els operadors condicionats per aquests qubits de manera que assumeixin un valor constant d'aquests.

En general, una **CNOT** realitza una negació condicionada a un qubit:

$$CNOT\,|a\rangle\,|b\rangle = |a\rangle\,|a \oplus b\rangle$$

Si el valor del qubit de condició és conegut, doncs, podem reescriure les operacions:

$$CNOT\,|0\rangle\,|b\rangle = |0\rangle\,|0 \oplus b\rangle = |0\rangle\,|b\rangle$$
$$CNOT\,|1\rangle\,|b\rangle = |1\rangle\,|1 \oplus b\rangle = |0\rangle\,X\,|b\rangle$$

Veiem que el qubit de control conegut no es fa servir, de manera que es pot eliminar sempre i quan se sàpiga el seu valor.

El mateix passa amb les **Toffoli** les quals recordem realitzen l'operació:

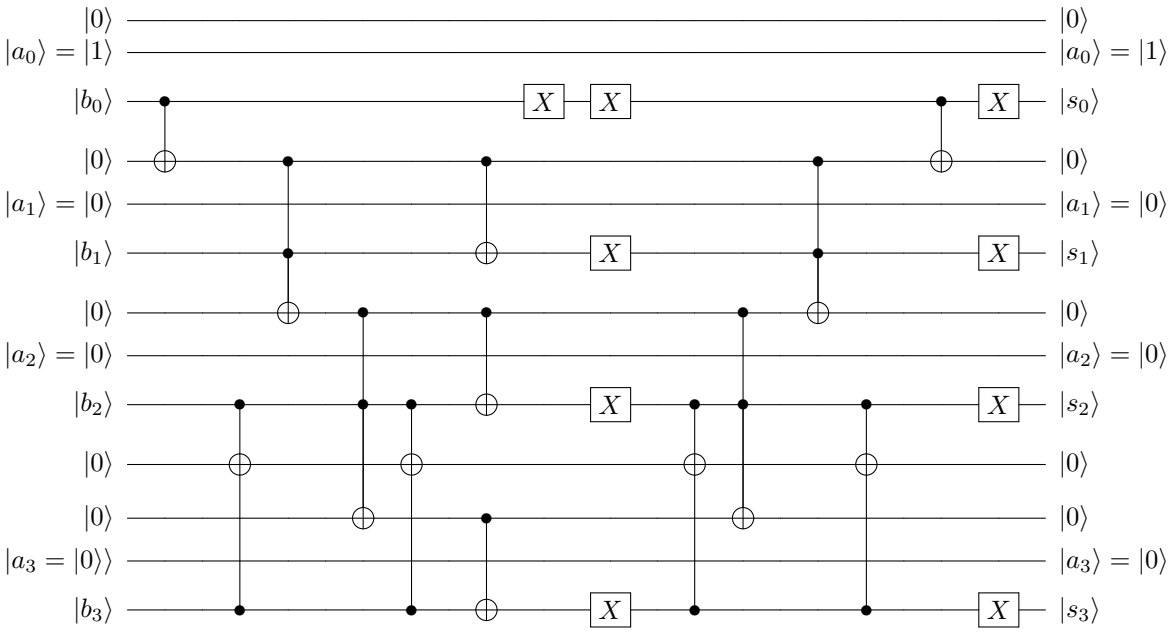
$$\text{Toffoli}\,|a\rangle\,|b\rangle\,|c\rangle = |a\rangle\,|b\rangle\,|a \wedge b \oplus c\rangle$$

Quan es coneix el valor d'un dels qubits de control, doncs:

$$\text{Toffoli}\,|0\rangle\,|b\rangle\,|c\rangle = |0\rangle\,|b\rangle\,|0 \wedge b \oplus c\rangle = |0\rangle\,|b\rangle\,|c\rangle$$
$$\text{Toffoli}\,|1\rangle\,|b\rangle\,|c\rangle = |1\rangle\,|b\rangle\,|1 \wedge b \oplus c\rangle = |1\rangle\,CNOT\,|b\rangle\,|c\rangle$$

On de nou, els qubits coneguts son completament redundants i ens els podem estalviar.

Si assumim que el registre $|\mathbf{a}\rangle_n$ és igual a $|000\ldots01\rangle$, podem simplificar totes les **CNOT** i **Toffoli** controlades pels qubits d'aquest registre:



Tot i que aquest mètode és completament vàlid per a obtenir el circuit, considerem convenient explicar l'efecte d'aquestes modificacions sobre totes les parts del circuit.

Primerament el circuit calcula $g[i, i+1] = a_i \wedge b_i$ sobre uns qubits ara temporals (aquells usats pel resultat en la versió alternativa del circuit), el qual indica si es genera un nou ròssec entre i i $i+1$. Com que tan sols a_0 és 1, tots aquests valors valdran 0, a excepció del primer que serà igual a l'estat de b_0. En el circuit de suma d'una unitat, aquesta modificació es tradueix en l'eliminació de totes les **Toffoli** que calculaven aquests valors sobre els qubits auxiliars, i la transformació de la primera en una **CNOT**, que calculi $g[0, 1] = b_i$.

Després el circuit original calcula $p[i, i+1]$, el qual indica si es propaga elròssec i a $i+1$, i és igual a $a_i \oplus b_i$. En el nostre cas aquest valor serà sempre igual a b_i menys el primer, el qual seria $\neg b_i$ però no es calcula ja que recordem que mai cal aquest valor per a cap operació d'unió $g[i,j] = (g[i,k] \wedge p[k+1,j]) \oplus g[k+1,j]$. En el circuit de suma d'una unitat, el valor constant del primer operand es tradueix en l'eliminació de totes les portes **CNOT** que realitzaven aquest càlcul.

Seguidament s'usaven aquests valors inicials de p per a construir l'arbre de Fenwick d'aquesta taula sobre els qubits addicionals inicialitzats a $|0\rangle$. Aquest pas no canvia ja que aquests valors depenen de l'altre operand, de manera que en el circuit **ADD1** encara hi trobarem totes aquestes portes. Per exemple, en el circuit donat per $n = 4$, veiem que hi roman la **Toffoli** que computava $p[2,4] = p[2,3] \wedge p[3,4]$ sobre un dels qubis temporals.

Amb els valors de p, el circuit original calcula l'arbre de Fenwick de g sobre els valors ja emmagatzemats de $g[i, i+1]$. De nou, aquesta estructura de dades és la raó de la velocitat d'aquest disseny i conseqüentment no pot ser obviada ni simplificada. En el circuit d'aquest apartat, per tant, hi trobarem les mateixes portes **Toffoli** que calculen $g[i,j] = (g[i,k] \wedge p[k+1,j]) \oplus g[k+1,j]$ al circuit original. Per 4 qubits sense ròssec, veiem que hi ha les que calculen $g[0,2]$ i $g[0,3]$.

El circuit descrit fins ara és aplicat de nou de manera invertida al final, i posteriorment s'apliquen n portes **X** per a obtenir els qubits de la suma final. Entre aquestes dues execucions d'aquest subcircuit, el circuit original transforma cada $s_i \oplus a_i$ de la sortida del primer en $\neg s_i \oplus b_i$ mitjançant 1 **X** i 2 **CNOT** per $i > 1$ i una per $i = 0$. En aquest apartat aquest càlcul es pot realitzar amb una sola **CNOT** en comptes de dues, i transformant la de la primera terna en una **X** la que el qubit de condició és $|1\rangle$.

En realitza aquests canvis obtenim dues portes **X** seguides, de manera que les podem eliminar. El circuit final d'addició d'una unitat en temps logarítmic es mostra a la figura 23.

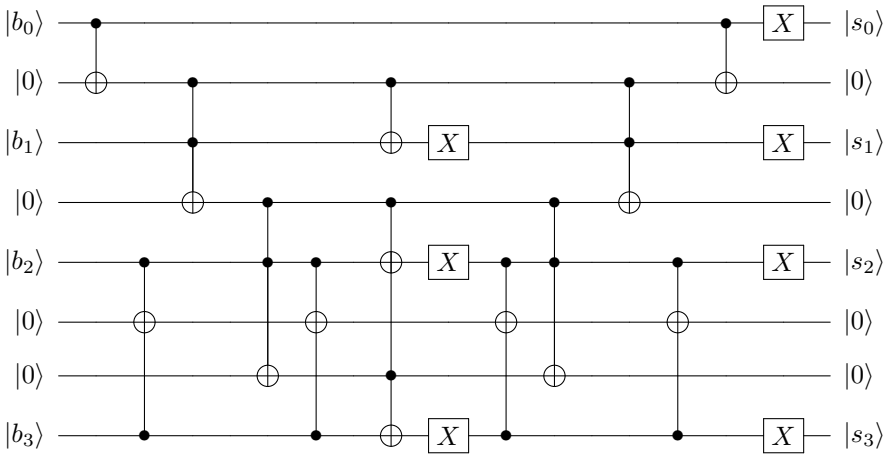


Figura 23: *Implementació Carry-Lookahead de l'operador ADD1 per a sumar una unitat a enters de* 4 *qubits.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Complexitat

La profunditat del circuit és logarítmica respecte el tamany n dels registres, de l'ordre de $\approx 2log\ n$.

El nombre de qubits addicionals requerits és lineal en n.

La quantitat d'operadors del conjunt bàsic necessaris per a obtenir-ne una implementació completa es mostren a la taula 7.

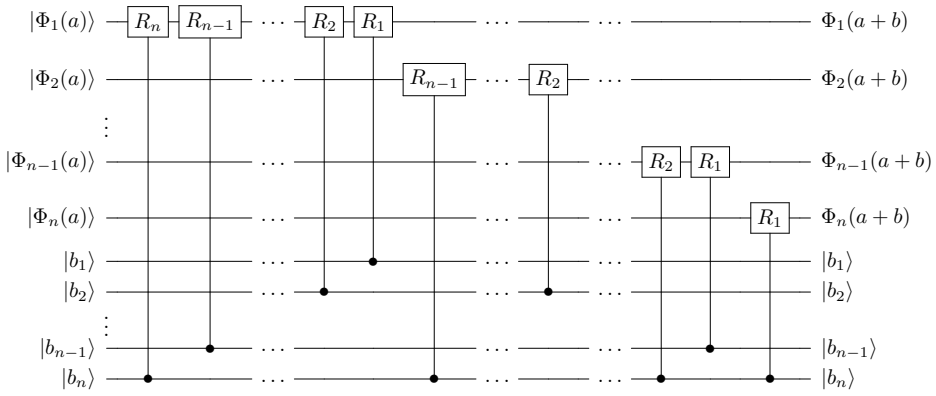
	Conjunt bàsic		
Porta	X	CNOT	Toffoli
Quantitat	$2n-1$	3	$\approx 4n$

Taula 7: *Nombre de portes de la implementació Carry-Lookahead de ADD1.* **Font:** Elaboració pròpia.

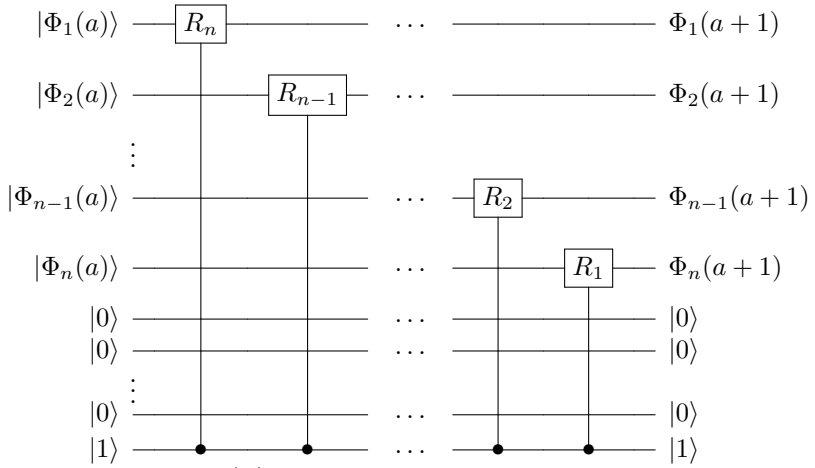
3.3.3 Suma d'una unitat amb QFT

La transformada de Fourier quàntica ens permet transformar un registre $|\mathbf{a}\rangle_n$ per a poder operar sobre els seus bits individuals mitjançant portes de fase R_k, les quals sumen 2^{-k} a la fase. Amb una aplicació correcta d'aquestes a cascun dels qubits del registre transformat $|\Phi(\mathbf{a})\rangle_n$ és equivalent a la suma d'un valor constant un cop desfeta la transformació.

D'aquesta manera, podem sumar 1 aplicant una sèrie de rotacions, les quals sumen una potència de dos a la fase de cada estat en funció de la posició que pren cada qubit en l'exponent de la fase. També es pot derivar el circuit a partir del de suma de dos operands:



Si assumim que el registre $|\mathbf{b}\rangle\,|b_1 b_2 \dots b_{n-1} b_n\rangle$ és igual a $00\dots01$, podem eliminar la gran majoria de portes, ara condicionades a qubits de valor constant $|0\rangle$:



Podem eliminar el registre $|\mathbf{b}\rangle_n$ constant, ja que totes les operacions són condicionades al qubit $|b_n\rangle$ de valor constant $|1\rangle$, el quals ens permet obtenir el circuit final de suma, mostrat a la figura 24

Complexitat

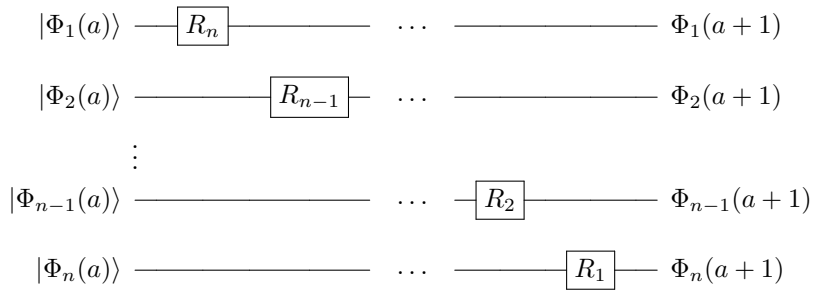


Figura 24: *Implementació de l'operador ADD1 per a sumar una unitat a enters de n qubits mitjançant la transformada de Fourier quàntica.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

La profunditat del circuit és constant si no cal realitzar les transformacions, o lineal respecte el tamany n dels registres, $\approx 2n$ si s'han de fer servir els operadors **QFT** just abans i després de la suma.

El nombre de qubits addicionals requerits és 0.

La quantitat d'operadors del conjunt bàsic necessaris per a obtenir-ne una implementació completa es mostren a la taula 8.

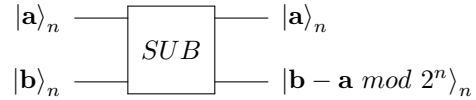
	Conjunt bàsic			
Porta	X	CNOT	Toffoli	R_k
Quantitat	0	0	0	n

Taula 8: *Nombre de portes de la implementació QFT de ADD1. La taula mostra el nombre de portes del conjunt bàsic que calen per a construir el circuit des de zero, assumint que ja es té el registre transformat. En cas contrari cal afegir $O(n^2)$ portes per als operadors **QFT**.* **Font:** Elaboració pròpia.

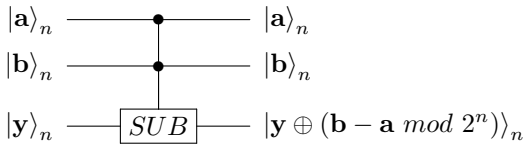
3.4 Resta de naturals i enters

Una possible solució per al problema de la resta d'enters seria el disseny d'una implementació quàntica del circuit de resta de naturals booleà, o bé es podria usar la suma d'enters estudiada anteriorment precedida d'una negació del segon operand. Aquestes solucions serien bastant extenses ja que els circuits de resta d'enters i naturals no són sempre equivalents, tal i com passa amb els de suma. Per certes representacions, com el Ca2, que admeten valors positius i negatius, també es pot aprofitar el circuit de suma **ADD** i el de negació **NEG** per a negar un dels operands, efectivament realitzant la resta.

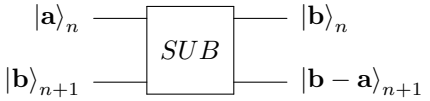
Tal i com hem fet amb les seccions anteriors, mostrem a la figura 25 un conjunt de prototips d'aquest operador per a fer-nos una idea de les operacions que hauran de realitzar els circuits a dissenyar.



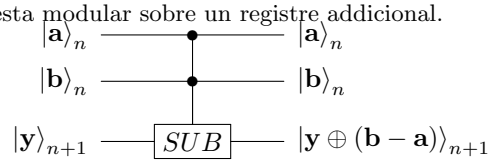
(a) Resta modular sobre un segon operand.



(b) Resta modular sobre un registre addicional.



(c) Resta no modular sobre el segon operand.



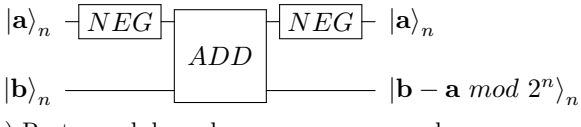
(d) Resta no modular sobre un registre addicional.

Figura 25: *Prototips de l'operador SUB per a modificar el segon operand amb el resultat (a i c) o no modificar els operands i emmagatzemar el resultat en un registre addicional (b i d) per a restes modulars i no modulars.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

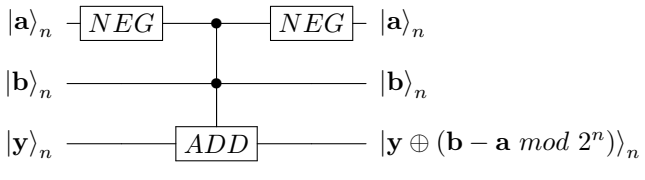
La literatura específica de circuits de resta no és gaire extensa per motius que seran obvis a l'última part d'aquesta secció, però de totes maneres presentarem les idees bàsiques per a dissenyar circuits quàntics capaços de restar naturals i enters.

3.4.1 Resta d'enters per negació

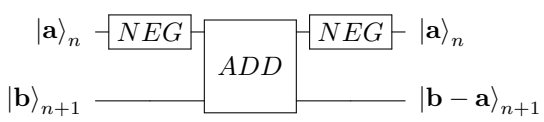
Si treballem amb enters, o un altre conjunt sobre el qual es pugui definir un bloc **NEG** que negui qualsevol element, podem implementar fàcilment els prototips anteriors negant un dels operands. A la figura 26 es mostren els circuits que implementarien cadascun dels quatre prototips proposats per operadors de resta.



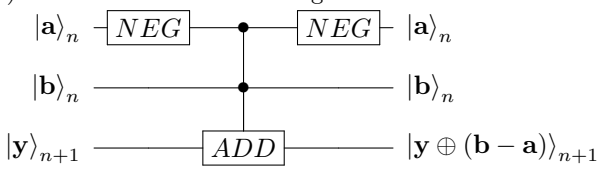
(a) Resta modular sobre un segon operand.



(b) Resta modular sobre un registre addicional.



(c) Resta no modular sobre el segon operand.



(d) Resta no modular sobre un registre addicional.

Figura 26: *Implementacions de l'operador SUB mitjançant negació d'un dels operands.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

3.4.2 Resta d'enters per inversió de circuits

Sorprenentment, hi ha un mètode trivial per a obtenir operadors capaços de realitzar aquesta operació, ja sigui modular, no modular, sobre un registre addicional, o qualsevol altra variant.

Sabem que és possible implementar l'operador **ADD** de qualsevol de les maneres detallades anteriorment per a sumar dos naturals o enters:

$$ADD\,|\mathbf{a}\rangle_n\,|\mathbf{b}\rangle_n\,|0\rangle = |\mathbf{a}\rangle_n\,|\mathbf{a}+\mathbf{b}\rangle_{n+1}$$

I sabem que aquest és reversible:

$$ADD^{-1}\,|\mathbf{a}\rangle_n\,|\mathbf{a}+\mathbf{b}\rangle_{n+1} = |\mathbf{a}\rangle_n\,|\mathbf{b}\rangle_n\,|0\rangle$$

Podem realitzar un canvi de variable $a' = a$, $b' = a + b$, de manera que l'efecte d'aquesta inversa és:

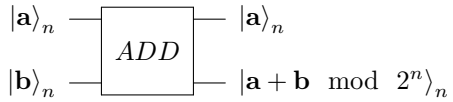
$$ADD^{-1}\,|\mathbf{a}'\rangle_n\,|\mathbf{b}'\rangle_{n+1} = |\mathbf{a}'\rangle_n\,|\mathbf{b}'-\mathbf{a}'\rangle_n\,|0\rangle$$

És a dir, a partir dels prototips de **ADD** de la secció anterior (8) podem obtenir diverses implementacions dels prototips de **SUB** mostrats a la figura 26.

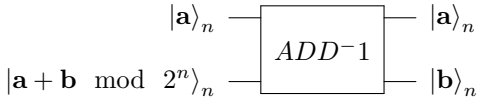
Els circuits de resta mai seran més eficients que els de suma[11], de manera que els podem definir directament a partir d'aquests. A continuació mostrem com obtenir circuits per a realitzar qualsevol dels prototips de **SUB** a partir de blocs *ADD*.

Resta modular sobre un operand

Podem obtenir una implementació simple d'aquest operador tal i com s'ha detallant anteriorment mitjançant la inversió d'algun circuit reversible que realitzi la suma modular de dos enters o naturals de la forma:



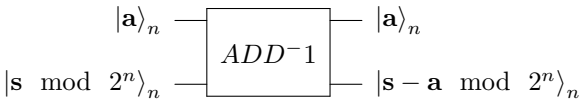
El seu invers doncs, és:



És a dir, si tenim a, i $a + b \mod 2^n$, el circuit calcula el valor a afegir al primer operand per a obtenir el segon, el qual és una resta. El benefici d'usar aquesta definició per la resta és que el bloc **SUB** derivate de **ADD** operarà sobre el mateix que aquest, de manera que podem definir restes sobre naturals, enters o decimals de qualsevol rang a partir de la porta de suma sobre el mateix conjunt.

Reorganitzant el circuit i aplicant la substitució $s \equiv a + b\ mod\ 2^n$ obtenim:

[11] En cas contrari podriem dissenyar operadors de suma a partir dels de resta, de complexitat igual gràcies a la reversibilitat, resultant en contradicció.



El qual és equivalent al prototip si assumim que s és representable amb n qubits. La figura 27 mostra la implementació d'aquesta resta basada en invertir un bloc de suma.

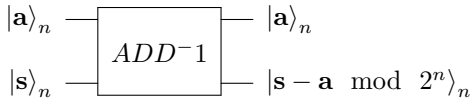
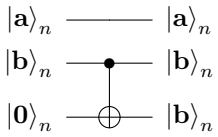


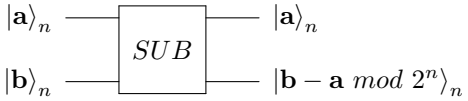
Figura 27: *Circuit que realitza la resta modular sobre el segon operand de dos registres de n qubits, derivat de la suma modular també sobre un operand.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Resta modular sobre un registre addicional

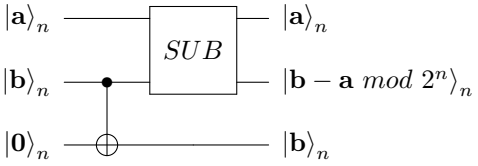
La porta **CNOT** permet copiar[12] valors de qubits de la superposició, de manera que podem crear una copia del registre $|\mathbf{s}\rangle_n$:



De manera que si assumim l'existència d'una implementació de l'operador **SUB** de la secció anterior amb prototip:



Podem calcular la resta sense perdre els operadors.



Tot i no ser imperatiu, podem reorganitzar els registres per a que el circuit s'ajusti millor al prototip mitjançant portes **SWAP**. Aquest circuit es mostra a la figura 28.

[12]Quan s'aplica sobre un registre, ens referim a aplicar la **CNOT** sobre cadascun dels elements del registre, el qual es pot realitzar en temps constant si s'apilen totes les operacions.

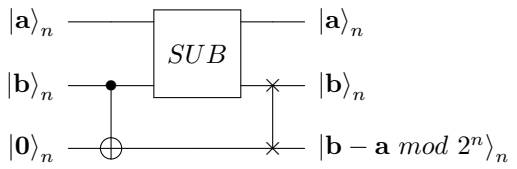
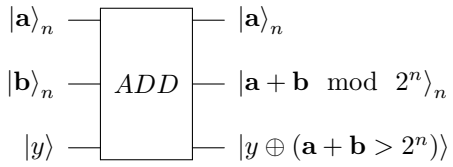


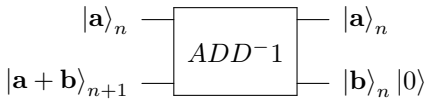
Figura 28: *Circuit que realitza la resta modular sobre un registre addicional de dos registres de n qubits, derivat de la resta modular sobre un operand.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Resta no modular sobre un operand

Podem obtenir una implementació de la resta no modular mitjançant un procés anàleg a l'usr per a dissenyar un circuit de resta modular. Assumim que existeix un circuit de suma no modular amb prototip:



El seu invers doncs, és:



Aplicant la substitució $s = a + b$ obtenim el disseny final, mostrat a la figura 29.

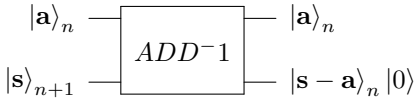


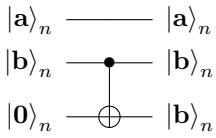
Figura 29: *Circuit que realitza la resta no modular sobre el segon operand de dos registres de n qubits, derivat de la suma no modular també sobre un operand.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Cal notar que el circuit obtingut no és completament idèntic al demanat, en deixar a $|0\rangle$ l'últim qubit de la resta ja que el tamany original del registre resultat era n qubits. Depenent del conjunt de nombres sobre els quals s'estigui realitzant la resta, es pot calcular aquest valor de manera senzilla. Per exemple si es calcula la resta de naturals ($s \geq a$) aquest serà 0, i si es calcula la resta d'enters tan sols cal copiar el bit de major pes del resultat $|\mathbf{b}\rangle_n$ amb una **CNOT**.

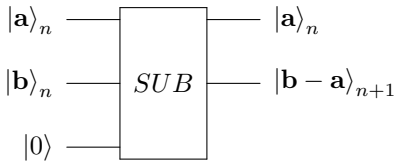
El nombre de qubits del registre $|\mathbf{b}\rangle_{n+1}$ tampoc és el mateix que en el prototip, però això no representa un problema a l'hora de dissenyar els circuits ja que, com hem vist abans, és trivial modificar el tamany dels registres dels operands quan s'entén el funcionament general de l'algorisme de suma en qüestió. El mètode més senzill seria assumir que, tal i com s'indica al prototip, el valor s és representable en n qubits, de manera que el registre $|\mathbf{s}\rangle_{n+1}$ és igual a $|\mathbf{s}\rangle_n|0\rangle$.

Resta modular sobre un registre addicional

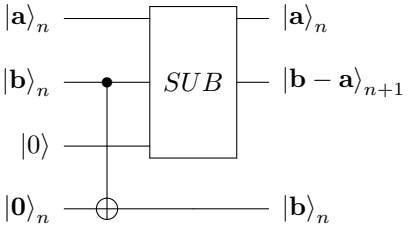
El procediment és també anàleg a l'anterior. Copiem els valors del registre $|\mathbf{s}\rangle_n$ mitjançant **CNOT**:



Fen ús de l'operador **SUB** de la secció anterior amb prototip:



El qual assumeix un b representable en n qubits i també calcula el qubit addicional de la resta.
Podem calcular la resta sense perdre els valors dels operadors:



Podem reorganitzar els resultats mitjançant portes **SWAP** tot i no ser necessari, el qual es mostra a la figura 30.

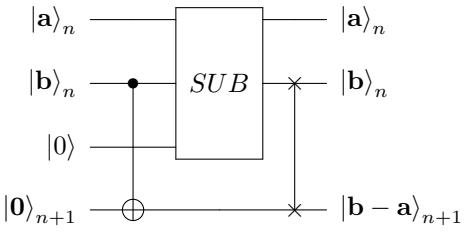


Figura 30: *Circuit que realitza la resta no modular sobre un registre addicional de dos registres de n qubits, derivat de la resta no modular sobre un operand.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

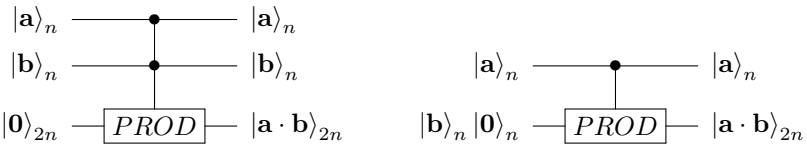
La porta **SWAP** s'ha usat amb una mica de llibertat ja que caldria que els dos registres fossin del mateix tamany per a poder realitzar l'intercanvi qubit a qubit. En la implementació real el qubit $|0\rangle$, el qual s'usa com a padding pel registre $|\mathbf{b}\rangle_n$ es passa a usar en el resultat. De totes maneres l'inercanvi dels registres es fa tan sols per a obtenir un digrama equivalent al del prototip, però si es vol un disseny correcte, un es pot referir al circuit anterior a la figura 30.

3.5 Producte de naturals i enters

Tal i com va passar amb els circuits de suma, la publicació dels algorismes de factorització i logaritme discret van resultar en un gran interès per part de la comunitat científica en el disseny i obtenció d'operadors quàntics capaços d'efectuar la suma de dos naturals emmagatzemats en registres de n qubits.

Com que la gran majoria de circuits de multiplicació de naturals han estat publicats o bé com a parts de la implementació completa dels algorismes quàntics mencionats anteriorment, o amb l'objectiu de ser usats en aquests, no hi ha gaire recerca en la multiplicació eficient d'enters. De totes maneres, donarem indicacions de les modificacions que caldria fer als circuits proposats per tal de poder operar tant amb naturals positius com enters en complement a 2.

A l'hora de dissenyar circuits de suma hem tractat d'obtenir variants modulars i no modulars, ja que certs algorismes requereixen el ròssec de la suma (o resta) i d'altres no, però tots aquests dissenys modulars poden ser modificats trivialment per a calcular l'últim bit del resultat, i la majoria dels no modulars poden ser alterats sense gaire complicació per a no usar l'últim qubit. El producte no modular no és una operació d'interés pel que fa a l'algorisme de factorització de Shor, de manera que és obviat per la gran majoria del les publicacions. Com que l'objectiu d'aquest document és proporcionar eines per a implementar qualsevol càlcul aritmètic, tractarem de mostrar modificacions dels circuits escollits que permetin realitzar aquestes operacions. Més específicament, buscarem circuits que implementin els prototips mostrat a la figura 31. En aquesta secció tan sols donarem un circuit per la versió que opera sobre un registre addicional, però en la propera secció es donarà una versió del segon prototip per a operar sobre un dels operands i estalviar-se qubits.



(a) Sobre un registre addicional. (b) Sobre un operand.

Figura 31: *Prototips de l'operador PROD per a calcular el producte de dos registres de n qubits sobre un registre addicional (a) o sobre un dels operands (b).* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Els primers operadors de producte modular van ser publicats juntament amb els de suma un any després de la publicació dels algorismes de factorització i logaritme discret ([2], [7]), basats en els algorismes de multiplicació habituals de suma del primer operand ponderat per 2^k condicionada a l'estat del bit k del segon. Com que aquestes multiplicacions es descomponen en un conjunt fix de sumes, investigacions posteriors van dissenyar circuits capaços de realitzar aquestes sumes en paral·lel, ja sigui dividint-les en blocs ([16]), mitjançant el mètode de suma *Carry-Save* ([24],[25],[26]), o l'algorisme *Booth* de multiplicació d'enters amb signe ([27]). Altres dissenys es basen en la Transformada de Fourier quàntica dels operadors per a efectuar aquestes sumes mitjançant canvis de fase ([16], [28]).

Com s'ha mencionat abans, però, tots aquests dissenys han estat creats amb l'objectiu de portar l' optimització de les operacions fins als límits que permeti el model de computació quàntica. De la mateixa manera que cal entendre el funcionament bàsic de les ALU[13] abans d'endinsar-se en el mon dels algorismes i tècniques per a optimitzar càlculs, mostrarem primerament un circuit bàsic de multiplicació d'enteniment senzill abans d'explicar el funcionament dels complexos dissenys optimitzats per a l'algorisme de Shor.

[13]Una ALU, o *Arithmetic Logic Unit*, és el cor d'un processador, i se n'encarrega de realitzar totes les operacions aritmètiques sobre els registres de bits.

3.5.1 Circuit Bàsic

Tot i que pugui semblar redundant, aquesta secció és potser de les més importants del document, ja que no tan sols mostrarà un disseny eficient de càlcul sobre qubits i n'analitzarà la seva complexitat, sinó que donarà una idea del procediment a seguir i tècniques a usar a l'hora d'adaptar un algorisme clàssic per a funcionar sobre un computador quàntic, el qual és l'objectiu principal del projecte.

L'algorisme de multiplicació típic és explicat a l'apèndix B en detall, però el seu funcionament es basa en descompondre el producte mitjançant la representació en base 2 d'un dels operands:

$$\begin{aligned} x \cdot y &= x \cdot (y_{n-1}y_{n-2}\dots y_1y_0) \\ &= x \cdot (y_{n-1}2^{n-1} + y_{n-2}2^{n-2} + \dots + y_1 2^1 + y_0 2^0) \\ &= y_{n-1}x2^{n-1} + y_{n-2}x2^{n-2} + \dots + y_1 x 2^1 + y_0 x 2^0 \end{aligned}$$

És a dir, la suma del registre x ponderada per tot 2^i només si $y_i = 1$. Com s'ha mencionat a la introducció, gràcies al fet que hem trobat dissenys capaços d'executar tota porta lògica sobre qubits, podríem realitzar una traducció literal del circuit clàssic sense gaires complicacions. El problema d'usar aquest mètode és que obtindríem un circuit que requereix d'un qubit addicional per a la majoria de portes lògiques simulades, i amb una profunditat molt més elevada del que es podria esperar. A la pràctica, tot i que és possible realitzar qualsevol càlcul clàssic, simular directament l'aritmètica lògica sol ser la pitjor opció possible per a realitzar qualsevol operació, ja que el *hardware* quàntic s'especialitza en ser capaç de realitzar un conjunt completament diferent d'operacions.

Seria interessant, doncs, estudiar si és possible dissenyar un circuit de multiplicació basat en el mètode anterior mitjançant operacions específiques del model quàntic, preferentment operadors del conjunt $\{\mathbf{X}, \mathbf{CNOT}, \mathbf{Toffoli}\}$.

Ja hem vist que és possible obtenir operadors **ADD** de diverses formes i complexitats, però per a implementar la multiplicació hem de ser capaços d'efectuar aquesta operació condicionada, la qual tindria un prototip mostrat a la figura 32.

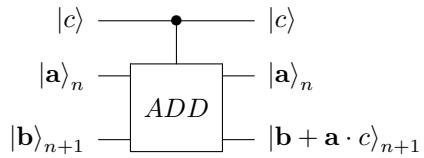


Figura 32: *Prototip de l'operador ADD no modular condicionat per a sumar dos registres de n sobre el segon només quan el qubit $|c\rangle$ sigui $|1\rangle$.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Per a l'algorisme també ens caldrà una porta capaç de realitzar el producte de x per potències de dos. En computació clàssica aquest procediment es realitza simplement movent els cables dels circuits un nombre de posicions iguals a la potència, tot afegint zeros (o uns, veure appèndix B) a les posicions buides. Aquesta operació és clarament no reversible, ja que es perd informació dels qubits dels extrems, de manera que és impossible d'implementar. Una alternativa en seria l'operador **Rotate-k** el qual realitza una rotació dels valors dels qubits de k posicions. Si el valor dels últims k qubits és de $|0\rangle$, aquesta rotació és equivalent al shiftat, de manera que caldrà tenir en compte el tamany dels registres a l'hora de definir el circuit. El prototip d'aquest operador es mostra a la figura 33.

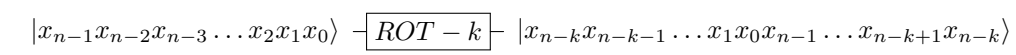


Figura 33: *Prototip de l'operador ROT-k per a realitzar una rotació dels qubits d'un registre $|\mathbf{x}\rangle_n$.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Si tenim implementacions d'aquests dos operadors, podem dissenyar un circuit capaç de calcular el producte de dos registres de qubits, de manera que els estudiarem individualment.

Suma condicionada

Òbviament podríem redissenyar circuits des de zero per a acomodar les noves necessitats, però això requeriria refer tots els dissenys, el qual sembla feina redundant per una modificació tan aparentment simple. Trobar un mètode per a transformar qualsevol circuit de suma en un operador condicionat també ens dona la capacitat d'usar qualsevol d'aquests circuits de suma, o aquells publicats en el futur, en el producte, permetent-nos una gran flexibilitat per a adaptar-nos a les restriccions espacials del sistema (**ADD** sense qubits addicionals), o temporals del problema (**ADD** en profunditat logarítmica).

La primera solució es basa en el fet que 0 és l'element nul de la suma, de manera que si poguéssim transformar el primer operand de la suma a $|\mathbf{0}\rangle_n$ de manera condicionada, no caldria modificar el circuit de suma per a obtenir l'operador condicionat.

Com que hem de restaurar el valor de l'operand, cal tenir un registre temporal $|\mathbf{0}\rangle_n$ per a emmagatzemar els valors dels qubits. Mitjançant una porta **CSWAP**[14], la qual intercanvia els valors de dos qubits de manera condicionada al valor d'un terçer, podem realitzar aquest canvi condicional del primer operand[15]. El circuit capaç de realitzar la suma condicionada mitjançant aquest mètode es mostra a la figura 34.

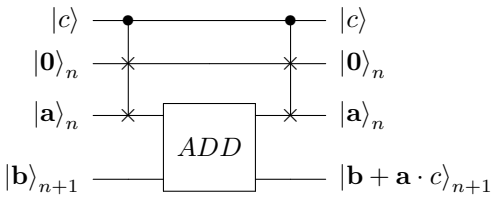


Figura 34: *Implementació de l'operador CADD mitjançant anul·lació del primer operand per a sumar dos registres de n sobre el segon només quan el qubit* $|c\rangle$ *sigui* $|1\rangle$. **Font:** Elaboració pròpia mitjançant Q-Circuit.

Alternativament, si fos possible condicionar totes i cadascuna de les portes de qualsevol circuit **ADD** a l'estat de $|c\rangle$, no caldria modificar cap dels operands ja que, si $|c\rangle$ és $|0\rangle$, cap de les condicions d'execució dels operadors de **ADD** es complirà i no es modificarà el segon operand.

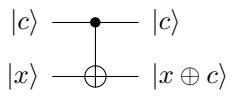
El problema és, aleshores, trobar un mètode per a condicionar qualsevol dels operadors dels circuits que implementen l'operació **ADD** a l'estat de $|c\rangle$. Ja s'ha esmentat a la introducció que tot operador unitari pot ser implementat amb portes d'un conjunt universal, però ja que estem treballant amb $\{\mathbf{X}, \mathbf{CNOT}, \mathbf{Toffoli}\}$, tractarem de realitzar aquesta tasca amb elles.

Sabem que totes les implementacions de **ADD** es basen en aquest conjunt de portes, de manera que mostrarem un mètode general per a condicionar cadascuna d'aquestes a l'estat d'un altre qubit.

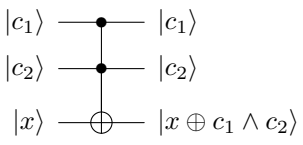
La **X** condicionada a un altre qubit $|c\rangle$ no és més que una **CNOT**:

[14] La porta **CSWAP** pot ser implementada extenent les 3 **CNOT** a **Toffoli**, on el nou qubit de condició és aquell de la **CSWAP**.

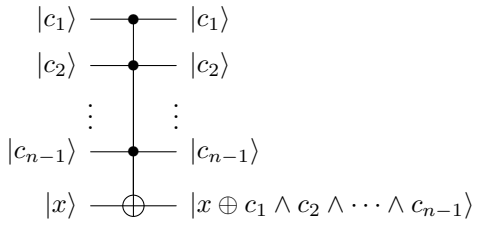
[15] Cal tenir en compte que quan mostrem una **CSWAP** actuant sobre un registre, ens referim a una porta per cada element del registre. Com que el qubit de condició és el mateix, aquestes operacions no poden ser executades paral·lelament, de manera que s'afegeria n a la profunditat del circuit. Mitjançant n qubits addicionals, però, un podria primer copiar aquest qubit de control a n qubits en temps $O(log\ n)$ de manera trivial. Alguns models de computador quàntic assumeixen l'existència de portes **FAN-OUT** explicades anteriorment, les quals permetrien realitzar aquesta operació en temps constant.



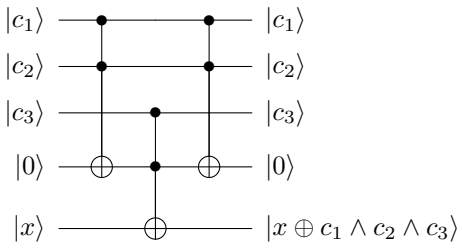
La **CNOT** condicionada a un altre qubit $|c\rangle$ no és més que una **Toffoli**:



Condicionar la **Toffoli** a un altre qubit és una mica més complex. Com s'ha explicat anteriorment, generalment anomenem **Toffoli-n** a la porta que realitza una negació d'un qubit condicionada a $n-1$ altres:



El qual pot ser implementat en $2n^2 - 6n + 5$[23] portes de 2 qubits sense usar-ne cap d'addicional i profunditat lineal en n. Aquestes complexitats no són importants ja que tan sols requerim una implementació de la **Toffoli-4**. De totes maneres, el circuit resultant és bastant complex i usa portes fora de $\{\mathbf{X}, \mathbf{CNOT}, \mathbf{Toffoli}\}$, de manera que presentarem una alternativa molt més simple, la qual usa menys portes amb la mateixa profunditat, a cost d'usar 1 qubit addicional:



El qual es pot extendre per a definir qualsevol porta **Toffoli-n**.

Si transformem tots els operadors de qualsevol implementació de **ADD** per a estar condicionats a un qubit addicional, obtenim un nou operador amb les mateixes propietats, el qual anomenarem també **CADD**. La figura 35 mostra com s'implementaria la suma condicionada amb aquest operador **ADD** modificat.

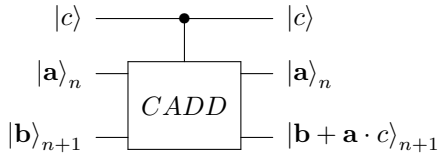


Figura 35: *Implementació de l'operador CADD mitjançant condicionament de tots els operands per a sumar dos registres de n sobre el segon només quan el qubit $|c\rangle$ sigui $|1\rangle$.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Rotació

El mètode més senzill per a realitzar una rotació és mitjançant portes **SWAP**. La figura 36 mostra la implementació de **ROT-1**:

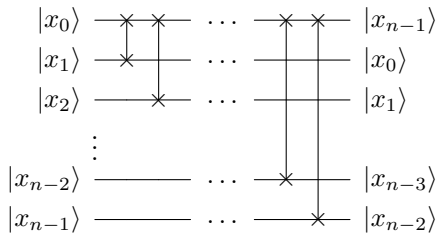


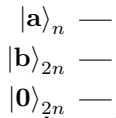
Figura 36: *Implementació de l'operador ROT-1 mitjançant SWAP per a rotar els valors dels qubits d'un registre de n.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

El cas és que aquesta operació és completament redundant ja que en el nostre model de computació quàntica hem assumit que qualsevol operador pot actuar sobre qualsevol subconjunt dels qubits del sistema, de manera que realitzar una $SWAP$ és equivalent a modificar els qubits sobre els quals operen els operadors posteriors. En la resta del document usarem aquestes portes per a facilitar la comprensió però no les tindrem en compte en l'anàlisi de complexitats ja que són completament opcionals.

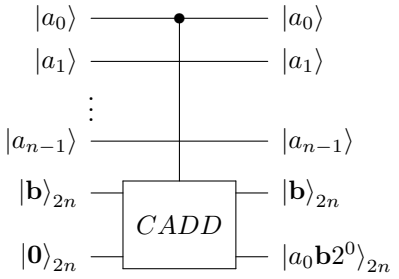
Circuit de producte

Ara que tenim blocs de suma condicionada i de rotació, podem dissenyar un circuit capaç de calcular el producte no modular de dos registres de n qubits. Per a fer el circuit el més general possible, augmentarem el tamany del registre d'entrada $|\mathbf{b}\rangle_n$ amb $|\mathbf{0}\rangle_n$, per a poder usar els blocs de suma estudiats. Cal destacar que hem proporcionat les eines per a dissenyar circuits de suma amb operadors de diferent tamany, de manera que el circuit que obtindrem és fàcilment optimitzable.

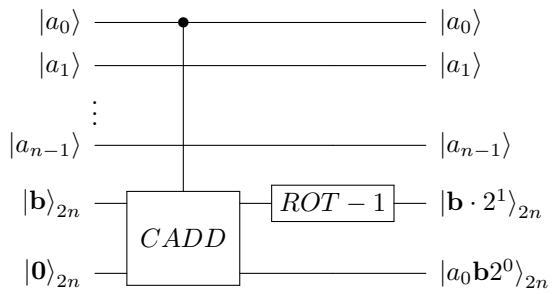
Afegint el registre resultat $|\mathbf{0}\rangle_{2n+1}$ obtenim l'estat inicial del sistema:



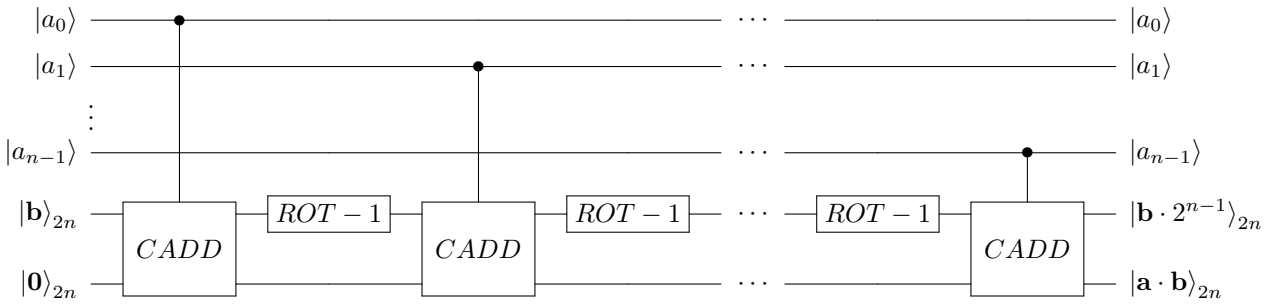
Podem realitzar la suma de $a_0 b 2^0$ sobre el resultat mitjançant una suma de $b2^0$ condicionada a l'estat de a_0:



Seguidament podem multiplicar el valor del registre $|\mathbf{b}\rangle_{2n}$ per 2 aplicant l'operador **ROT-1**, el qual afegirà un zero a l'esquerra gràcies al fet que hem inicialitzat els n qubits de major mes del registre a $|0\rangle$:



Si repetim el procediment per cada qubit del registre $|\mathbf{a}\rangle_n$ podem obtenir la suma final desitjada:



De manera que tan sols cal retornar el registre $|\mathbf{b}\rangle_n$ a l'estat inicial, el qual es pot realitzar mitjançant una rotació de $n+1$. El circuit obtingut en concatenar les n sumes condicionades amb la rotació final del segon operand es mostra a la figura 37.

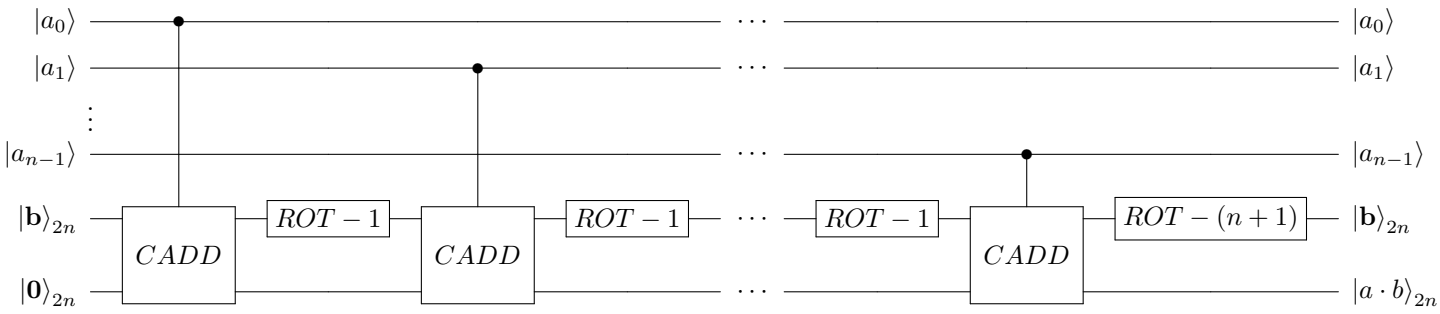
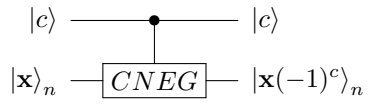


Figura 37: *Circuit que calcula el producte no modular de dos registres de n qubits sobre un registre addicional usant-ne un de temporal de n qubits.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Tot i que les portes **ADD** del circuit poden operar tant amb naturals o enters, els shiftats implementats com a rotacions no realitzen l'operació desitjada ja que els enters negatius haurien de tenir els qubits de major pes a $|1\rangle$. És per això que caldria modificar aquest circuit si es desitgés treballar amb enters. Una solució possible seria el disseny d'operadors de rotació per a realitzar el shiftat d'enters correctament, però això faria impossible ignorar aquests blocs en l'anàlisi de complexitat i possible implementació real.

La solució més senzilla i generalitzable és la negació condicionada dels operands i resultat. Podem realitzar el producte dels valors absoluts dels operands i negar el resultat en funció dels signes d'aquests. Per a poder fer això primer cal obtenir un operador de negació condicionada:



El qual es pot obtenir amb el mètode descrit anteriorment de condicionar tota porta de l'operador al nou qubit de control $|c\rangle$ sense modificar la profunditat del circuit però potencialment afegint n qubits auxiliars, depenent de si es vol minimitzar la memòria o el nombre d'operadors requerits.

De totes maneres, és completament possible obtenir aquest operador, de manera que es pot implementar el producte d'enters per negació condicionada dels operands, mostrat a la figura 38.

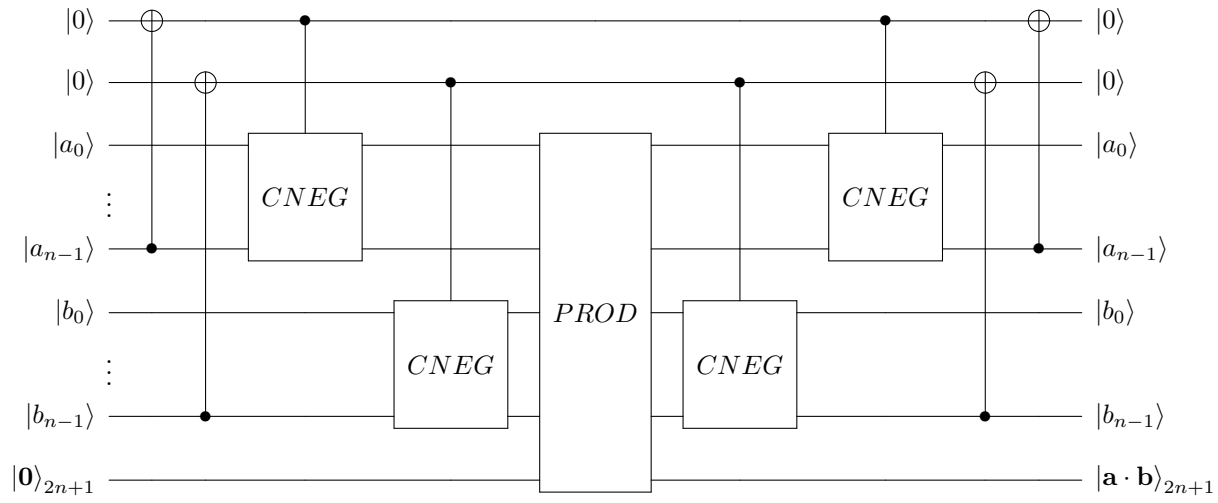


Figura 38: *Circuit que calcula el producte no modular d'enters per negació condicionada dels operands. El diagrama no inclou els qubits auxiliars requerits pels blocs individuals ja que són dependents de la implementació triada.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Complexitat

L'estudi de la complexitat és particularment complicat ja que hi ha dotzenes de possibles combinacions d'implementacions de **CADD** i **CNEG** en cas que desitgéssim el producte d'enters.

En general, la profunditat del circuit és de n blocs **ADD** seqüencials més 2 **NEG**, ja que les podem apilar en parelles. Si desitgéssim optimitzar aquesta profunditat, podriem implementar els operadors de suma mitjançant Carry-Lookahead, on cadascun té una profunditat de $O(log\ n)$[16], de manera que obtindríem una profunditat total de $O(n\ log\ n)$. En el pitjor dels casos, aquest circuit de producte de naturals requerirà $O(n)$ qubits addicionals per les implementacions de les sumes i els qubits requerits pel bloc **PROD** per a incrementar la mida del segon operand.

[16]Com s'ha mencionat a la secció corresponent, idealment es pot dissenyar un circuit amb profunditat $O(log^*\ n)$, el qual és essencialment constant, però la complexitat del disseny està per sobre de l'objectiu principal del projecte, el qual és servir com a guia per a dissenyar circuits que realitzin qualsevol càlcul o algorisme clàssic. Aquest circuit també assumeix l'existència de portes **FAN-OUT** de profunditat constant, el qual no és generalment assumit.

El nombre de qubits addicionals serà de $O(n)$ pel bloc de producte més el requerit pels blocs de suma. Com que aquests són aplicats de manera seqüencial podem reutilitzar els qubits temporals, de manera que tan sols cal reservar memòria equivalent a la necessària per a una sola suma amb la implementació de **ADD** escollida. En cas que es volgués minimitzar aquest valor, es podrien implementar els operadors **CADD** amb Ripple-Carry on les operacions han estat condicionades a un nou qubit mitjançant el mètode descrit a [23], de manera que tots els qubits addicionals que calen són els n necessaris per a augmentar el tamany del registre $|\mathbf{b}\rangle_n$ i el circuit tindria una profunditat de $O(n^2)$.

El nombre de portes del producte de naturals és, també, depenent de la implementació dels n blocs de suma, generalment $O(n^2)$.

3.5.2 Producte basat en aritmètica Carry-Save

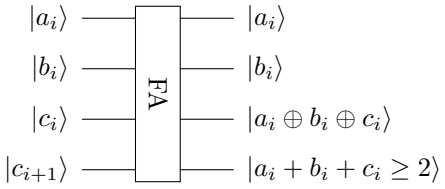
Com s'ha explicat anteriorment, la gran majoria dels circuits de producte han estat desenvolupats en l'àmbit dels algorismes de Shor de factorització i logaritme discret per a optimitzar la multiplicació modular fins als límits possibles del model, sovint a cost de generalitat de l'operador. Si bé no podem tractar aquests circuits publicats directament per a dissenyar l'operador de producte desitjat, podem aplicar les idees usades en aquests documents per a obtenir un mètode que ens permeti dissenyar circuits de multiplicació no modular amb complexitats de profunditat $O(\cdot)$ pràcticament equivalents a les dels millors operadors publicats.

Suma Carry-Save

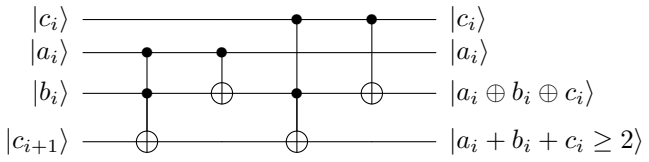
L'aritmètica Carry-Save és una tècnica per a realitzar sumes, amb importància equiparable a tècniques alternatives com Ripple-Carry o Carry-Lookahead. Aquesta tècnica no ha estat explicada a la secció de suma de naturals i enters ja que, tot i ser una tècnica de suma binària, no permet obtenir circuits de suma de dos registres.

Aquesta tècnica va se introduïda a la computació quàntica el 1998 amb la publicació *Quantum carry-save arithmetic*[24], i desenvolupada posteriorment en diversos estudis ([25], [26])) per a reduir-ne encara més la profunditat o nombre de qubits auxiliars.

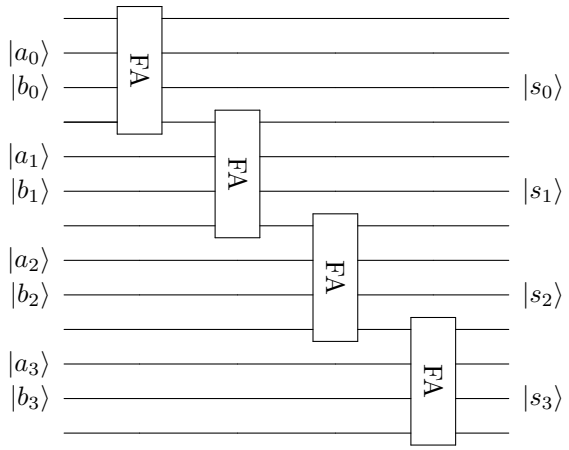
El seu funcionament es basa una modificació del bloc **Full-Adder** (o **FA**) semblant a l'usat en les implementacions més rudimentàries de Ripple-Carry:



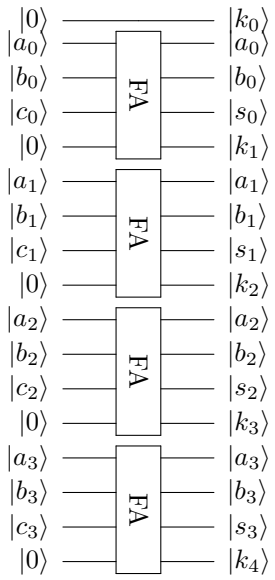
Amb implementació:



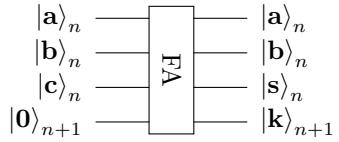
L'ús que fa l'algorisme d'aques bloc és, però, completament diferent. Ripple-Carry aplica aquesta porta seqüencialment per a trobar els ròssecs i valors finals de la suma de dos registres $|\mathbf{a}\rangle_n$ i $|\mathbf{b}\rangle_n$:



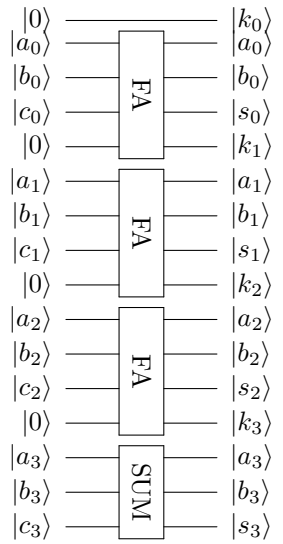
Carry-Save no troba en cap moment la suma final de dos operands, però permet transformar la suma d'un nombre arbitrari d'operands en la suma de dos. Per exemple, si appliquem aquesta tècnica a la suma de tres registres de quatre qubits, obtenim:



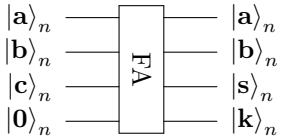
El qual representarem mitjançant una sola porta sobre els operands mitjançant una reorganització de les sortides, el qual és implementable de manera trivial mitjançant operadors **SWAP**:



Si eliminem l'últim qubit, podem obtenir una versió modular que ignora el qubit de major pes del registre $|\mathbf{k}\rangle_{n+1}$. Assumirem sense perdre generalitat que aquest últim qubit que seria perdut sempre valdria $|0\rangle$, de manera que obtenim:



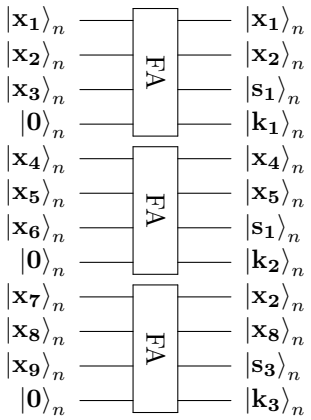
Amb representació equivalent:



Aquesta transformació pot semblar irrellevant, però és en realitat vital per a dissenyar càlculs més complexos ja que la suma original que cerquem $a+b+c$ és exactament igual a $s+k$, on $k_0 = 0$. Una forma de justificar aquest fet és imaginant l'algorisme de suma binària habitual amb tres operands, sense propagar els ròssecs:

$$\begin{array}{c|c} & 0101101 \\ + & 1101100 \\ + & 1011001 \\ \hline = & 2213202 \\ \hline = & 0011000 \\ & 1101101 \end{array}$$

És a dir, sumar s i k és equivalent a sumar els tres operands inicials. Aquest procediment es pot repetir tantes vegades com es vulgui per a transformar un nombre arbitrari de registres $|\mathbf{x_1}\rangle_n, |\mathbf{x_2}\rangle_n, \ldots, |\mathbf{x_m}\rangle_n$ a dos $|\mathbf{s}\rangle_n$, $|\mathbf{k}\rangle_n$ en $O(log\ m)$ passos aplicant blocs **FA** a ternes de registres en forma d'arbre. Per exemple, si tenim 9 registres a sumar:



Els podem transformar en un sol pas a 6 registres $|\mathbf{s_i}\rangle_n$,$|\mathbf{k_i}\rangle_n$, els quals compliran:

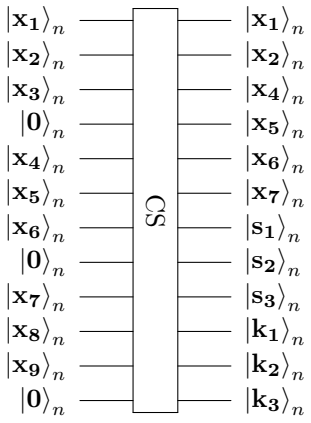
$$\sum_{i=1}^{9} x_i = \sum_{j=1}^{3} s_j + k_j$$

De manera general, una aplicació d'aquest pas permet reduir el nombre de registres a sumar en $\lfloor \frac{1}{3} \rfloor$, de manera que la quantitat de passos necessaris per a transformar m registres a 2 amb la mateixa suma és de l'ordre de $O(log\ m)$.

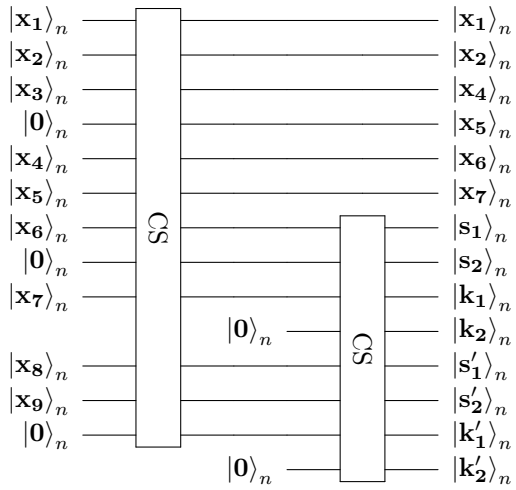
Una suma de m registres requeriria $m-1$ aplicacions d'un circuit de suma, però gràcies a aquest mètode podem realitzar aquesta mateixa suma amb $O(log\ m)$ passos de Carry-Save i una sola aplicació d'un circuit de suma de dos registres.

Per a facilitar la comprensió dels dissenys, representarem aquest pas de Carry-Save mitjançant l'operador **CS**, on també reorganitzarem les sortides mitjançant portes **SWAP**, les quals recordem són completament innecessàries i no s'hauríen de tenir en compte en anàlisis de complexitats ni implementacions reals del circuit.

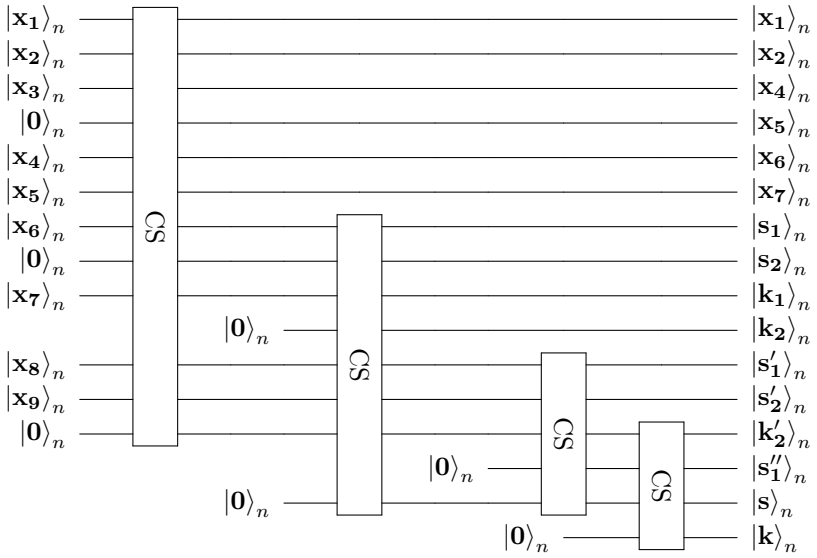
Aquest circuit actua sobre m registres, usa $\lfloor \frac{m}{3} \rfloor$ registres de qubits addicionals i redueix el nombre de registres a sumar en el mateix valor:



Aquest circuit es pot aplicar de nou sobre els $n - \lfloor \frac{n}{3} \rfloor$ registres de nou per a reduir encara més el nombre d'operands:

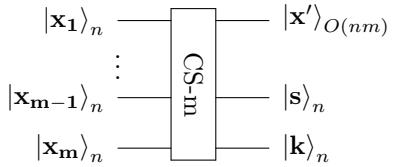


El qual, de manera general, cal aplicar-se $O(log\ m)$ vegades per a obtenir el resultat de dos registres esperat:

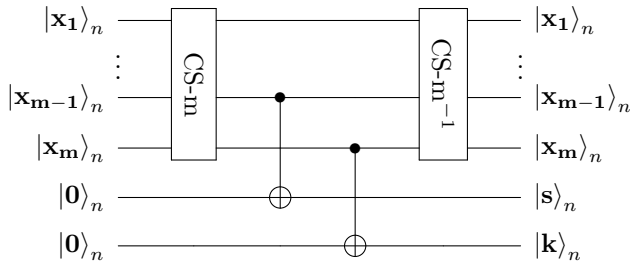


Podriem, per exemple, realitzar la suma de s i k sobre un nou registre i desfer tota la computació aplicant la seva inversa, obtenint el sumatori de tots els registres en pràcticament la mateixa profunditat que la suma de dos d'ells.

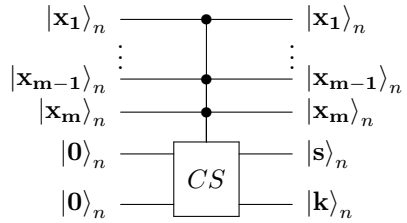
Per a simplificar futurs diagrames, representarem aquest esquema d'aplicació successiva de les $O(m)$ portes **CS** que calen per a obtenir els dos resultats mitjançant l'operador **CS-m**, on ignorarem els qubits auxiliars:



Com que de moment tan sols ens interessen els dos registres $|\mathbf{s}\rangle_n$ i $|\mathbf{k}\rangle_n$, aplicarem aquest circuit i posteriorment la seva inversa, tot realitzant una còpia dels resultats en dos nous registres addicionals:



Cal recordar que aquest bloc usa $O(nm)$ qubits temporals per a les execucions successives de **CA**. Anomenarem aquest bloc també **CS**, el qual no serà ambigu ja que en no modificar els operands, podem representar-lo mitjançant la notació d'operador condicionat:

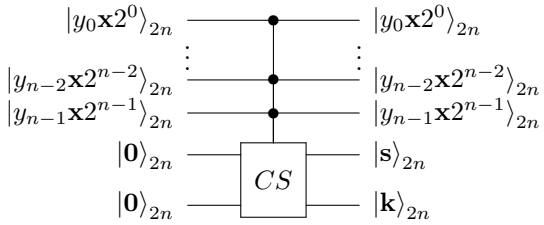


Producte mitjançant Carry-Save

La utilitat de Carry-Save en el disseny d'operadors de multiplicació és doncs òbvia. En un producte estem realitzant la suma d'un dels registres ponderada per cada potència de dos i condicionada a l'estat d'un dels qubits de l'altre registre:

$$x \cdot y = y_{n-1}x2^{n-1} + y_{n-2}x2^{n-2} + \cdots + y_1x2^1 + y_0x2^0$$

Si fóssim capaços d'obtenir tots els sumands y_ix2^i en paral·lel, podriem transformar-los en dos registres amb la mateixa suma en $O(log\ n)$ passos:



On $s + k$ és igual al producte desitjat, sempre i quan el tamany dels registres sigui prou gran per a que mai es perdi el valor del bit de major pes en l'aplicació de les **CA** internes, ja que recordem hem assumit que elròssec de cada suma, corresponent al qubit de major mes de cada k, és 0.

La millora d'eficiència a cost de memòria és un paradigma molt comú en la ciència de la computació, i és exactament els que ens permetrà obtenir aquest estat inicial sense augmentar la fita de profunditat que ens proporciona Carry-Save. Assumim que comencem amb els dos operands el producte dels quals volem calcular:

$$\begin{array}{r} |\mathbf{y}\rangle_n \ \text{——} \\ |\mathbf{x}\rangle_n \ \text{——} \end{array}$$

Ampliem la mida del segon operand per a que pugui admetre rotacions de fins a $n-1$ i afegim $n-1$ registres temporals del mateix tamany, on posteriorment hi emmagatzemarem cadascun dels n termes de la suma:

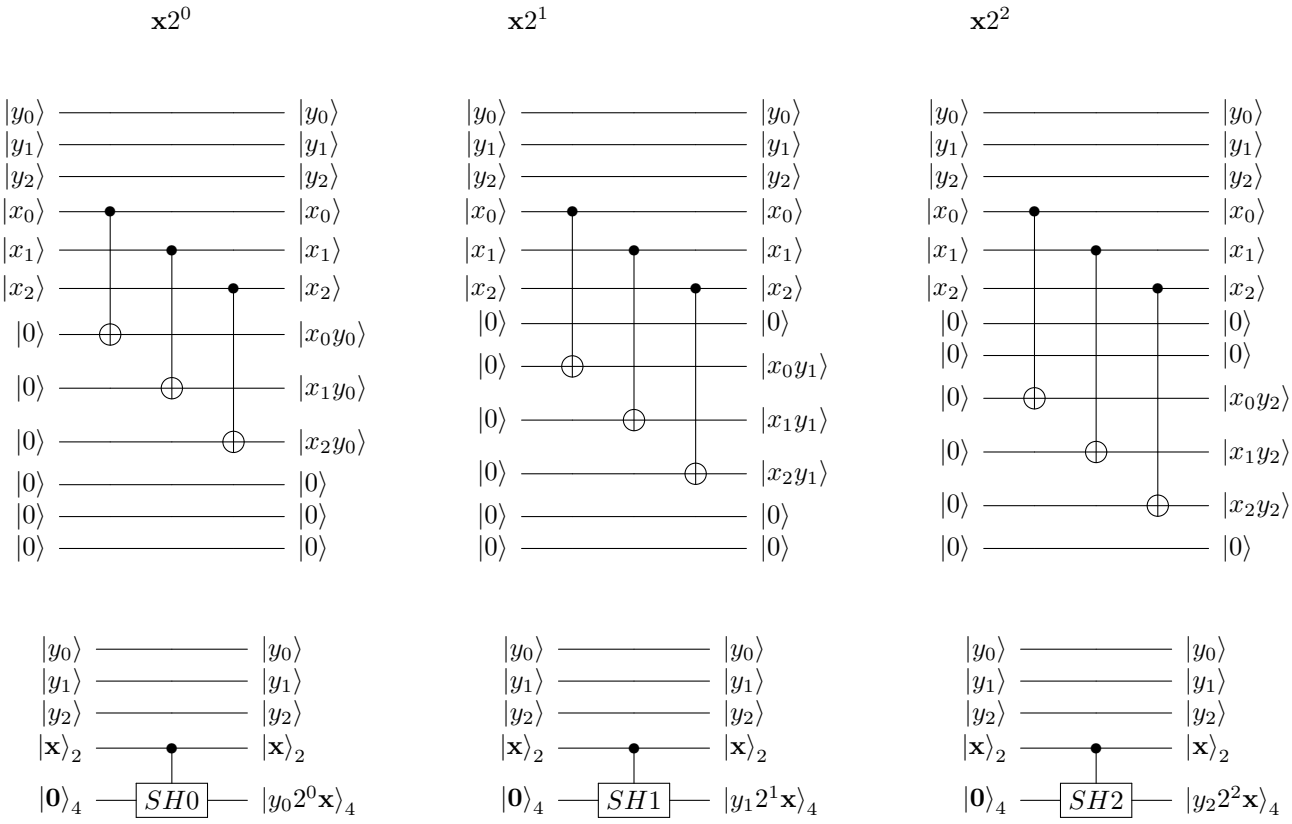
$$\begin{array}{r} |\mathbf{y}\rangle_n \ \text{——} \\ |\mathbf{x}\rangle_{2n} \ \text{——} \\ |\mathbf{0}\rangle_{2n} \ \text{——} \\ \vdots \\ |\mathbf{0}\rangle_{2n} \ \text{——} \end{array}$$

Mitjançant portes **Fan-Out** podriem copiar el valor de x a tots els nous registres, però molt pocs documents assumeixen l'existència d'aquest operador, de manera que mostrarem una manera alternativa de realitzar aquesta còpia, la qual ens permetrà també realitzar la rotació de cada terme en el mateix pas.

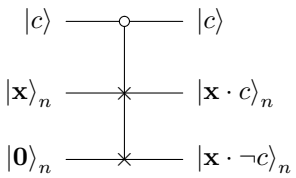
Assumim de moment que estem realitzant el producte de dos registres de tres qubits $|\mathbf{y}\rangle_3$ i $|\mathbf{x}\rangle_3$, i volem calcular $y_i\mathbf{x}2^i$ sobre un dels registres temporals $|\mathbf{0}\rangle_6$:

$$\begin{array}{r} |y_0\rangle \ \text{——} \\ |y_1\rangle \ \text{——} \\ |y_2\rangle \ \text{——} \\ |x_0\rangle \ \text{——} \\ |x_1\rangle \ \text{——} \\ |x_2\rangle \ \text{——} \\ |0\rangle \ \text{——} \\ |0\rangle \ \text{——} \\ |0\rangle \ \text{——} \\ |0\rangle \ \text{——} \\ |0\rangle \ \text{——} \\ |0\rangle \ \text{——} \end{array}$$

Podem copiar els qubits de $|\mathbf{x}\rangle_2$ directament mitjançant **CNOT**, però podem canviar els qubits control·lats per a efectivament multiplicar l'operand per una potència de 2, de manera que podriem calcular qualsevol dels termes de la suma. Anomenarem aquesta porta **SHk** (*Shift k*), ja que realitza una còpia amb un *shift* de k:



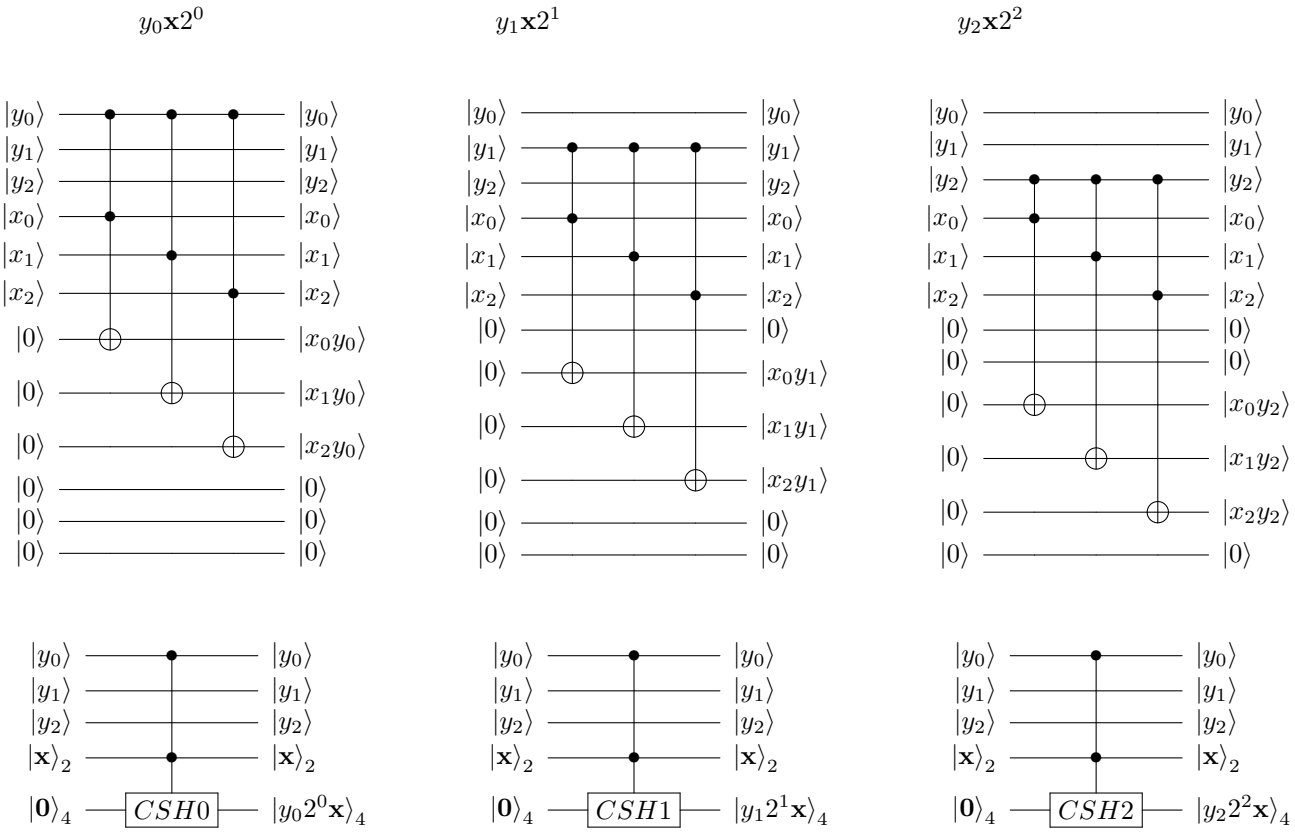
Els termes que volem, però, són aquests multiplicats per cada y_i. Una solució seria usar un bloc que anul·lés aquest registre en el cas que y_i fos $|0\rangle$, el qual anomenarem **CNULL** ja que realitza una anul·lació condicionada del registre. Aquest es pot implementar de manera general mitjançant un **SWAP** de l'operand amb un registre temporal inicialitzat a $|\mathbf{0}\rangle_n$ condicionat a l'estat $|0\rangle$ d'aquest qubit de control:



Sempre recordant que cal repetir l'operador per a retornar el valor original amb el mateix registre temporal en cas que es realitzi la substitució, i mai es pot modificar aquest ja que l'operació no seria reversible.

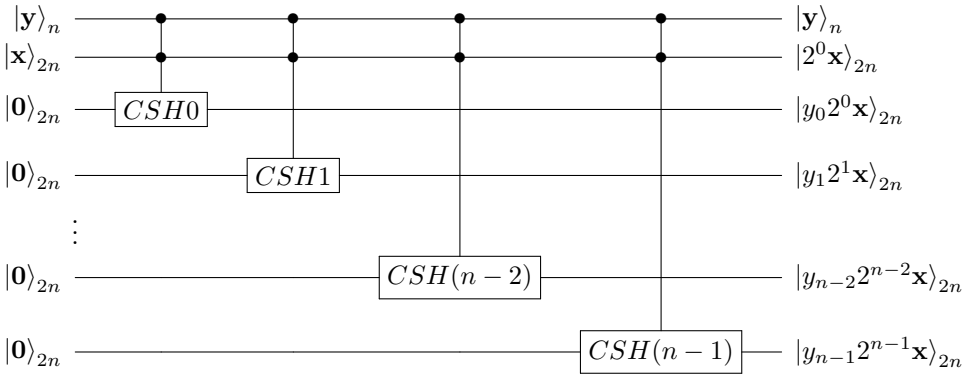
Com que totes les **SWAP** individuals venen condicionades pel mateix qubit, la profunditat d'aquest circuit seria lineal en n. Mitjançant un registre de n qubits temporal i portes **Fan-Out** o **CNOT** es pot realitzar la còpia del qubit de condició en temps $O(1)$ i $(log\ n)$ respectivament per a poder aplicar totes les **SWAP** en paral·lel.

En el nostre cas, podem usar una implementació alternativa basada en condicionar les portes que calculen cada terme sobre els registres inicialment buits a aquest nou qubit substituïnt les **CNOT** per **Toffoli**, el qual es pot també realitzar en temps lineal sense memòria extra o logarítmic amb un registre addicional temporal. Podem doncs, crear els operadors **CSH-k**, els quals realitzen una còpia d'un registre shiftada k posicions o deixen el resultat a $|\mathbf{0}\rangle_n$ en funció del qubit de condició:



De nou, una porta **Fan-Out** permetria copiar aquest registre i obtenir tots els termes en una sola operació, però com que hem assumit que aquest operand no existeix, n'usarem la implementació detallada anteriorment (15) de profunditat logarítmica.

Podem, doncs, obtenir tots els termes en el nostre sistema:



Hem omès el **Fan-Out** logarítmic de $|\mathbf{y}\rangle_n$ en el diagrama, però una implementació real hauria d'afegir $n-1$ registres $|\mathbf{0}\rangle_n$ i convertir-los a $|\mathbf{y}\rangle_n$ amb **CNOT**, i posteriorment aplicar les **CSHk** en un sol pas, cadascuna control·lada per una còpia de y diferent.

Amb aquest mètode podem obtenir tots els termes de la suma en $O(log\ n)$ usant $O(n^2)$ qubits temporals. Podem combinar els dos passos amb les seves inverses per a obtenir el circuit final de multiplicació, mostrat a la figura 39.

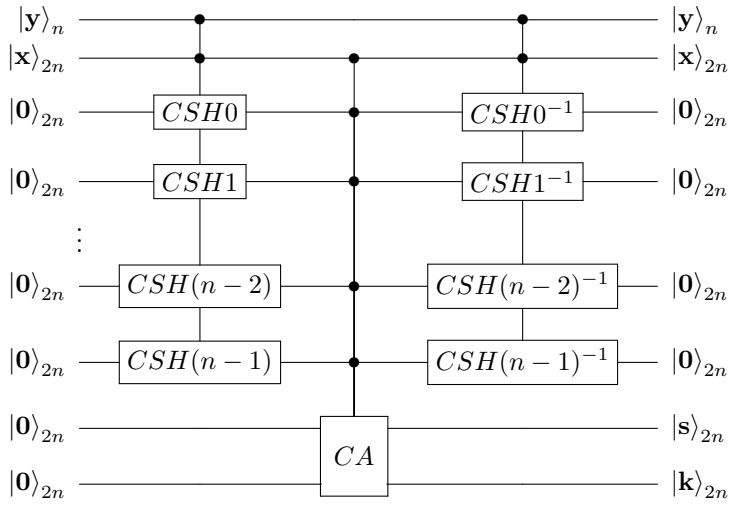
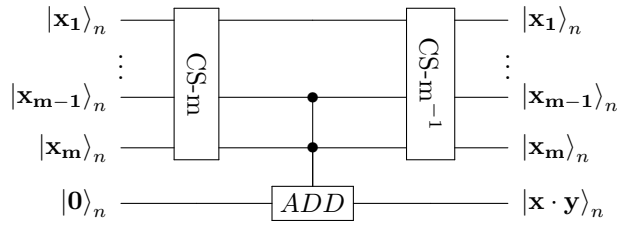


Figura 39: *Circuit que calcula el producte de naturals basat en Carry-Save. El diagrama no inclou els qubits auxiliars requerits pels blocs individuals ja que són dependents de la implementació triada, però s'han donat circuits de possibles dissenys dels d'aquests blocs.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Cal destacar que el circuit no acaba de calcular el producte desitjat, però això no és un problema ja que com hem vist, la representació de naturals com a la suma és útil quan es treballa amb aritmètica Carry-Save, el qual aprofitarem en el futur. Si es volgués calcular el producte final tan sols cal modificar el bloc **CA** per a usar qualsevol dels blocs **ADD** per a sumar els dos resultats:



Complexitat

L'estudi de la complexitat d'aquest disseny és similar al del circuit de producte bàsic, ja que hi ha diverses possibles opcions per a implementar els blocs que el componen, però assumirem que s'usen els blocs que hem mostrat anteriorment.

De manera general, aquest disseny de producte de dos registres de n qubits consta de $\approx 2n$ blocs **CSHk** i $\approx log\ n$ blocs **CS**, cadascun compost de $O(n)$ **FA**, tot i que podem reduir aquesta fita si tenim en compte que el nombre d'operadors **FA** requerits va disminuint. El nombre d'aquestes portes que calen per a implementar l'ièssim **CS** és aproximadament igual a $\frac{2n}{3}(\frac{2}{3})^i$, i per tant, la suma de $O(log\ n)$ d'aquests termes és:

$$\frac{2n}{3} + \frac{2n}{3}\frac{2}{3} + \frac{2n}{3}(\frac{2}{3})^2 \cdots + 1 = \frac{2n}{3}(1 + \frac{2}{3} + (\frac{2}{3})^2 + \cdots + 1)$$

Si continuem expandint el terme de la dreta infinitament obtenim una fita superior de la complexitat, el qual és una sèrie geomètrica amb valor conegut:

$$\frac{2n}{3}(1 + \frac{2}{3} + (\frac{2}{3})^2 + \cdots + 1) < \frac{2n}{3}(1 + \frac{2}{3} + (\frac{2}{3})^2 + \cdots) = \frac{2n}{3}3 = O(n)$$

És a dir, el circuit usa $O(n)$ blocs **FA**, el qual es tradueix a $O(n)$ portes **Toffoli** i **CNOT**. Sumant aquestes amb les $O(n^2)$ **Toffoli** requerides per als $O(n)$ blocs **CSHk**, obtenim un total de $O(n)$ **CNOT** i $O(n^2)$ **Toffoli**.

Pel que fa a la profunditat del disseny, que és el que generalment es vol minimitzar, tenim $O(log\ n)$ passos per a realitzar les còpies del registre x ($O(1)$ si existíssin **Fan-Out**) més $O(log\ n)$ passos per al bloc **Carry-Save**, obtenint una profunditat total de $O(log\ n)$ per a calcular els dos registres s i k. En cas que es desitgés calcular el valor del producte, es podria aplicar el sumador Carry-Lookahead, per a una complexitat final de també $O(log\ n)$.

Hem omès els qubits temporals requerits pels subcircuits per a facilitar la representació, el qual no és important en l'anàlisi asimptòtic de la mida del circuit ja que el factor limitant d'aquesta mètrica són les n còpies de $|\mathbf{x}\rangle_{2n}$ que cal realitzar, el qual implica una complexitat espacial de $O(n^2)$.

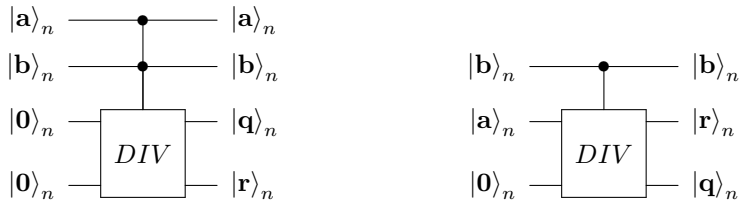
Si es volgués treballar amb enters, caldria ajustar aquesta anàlisi per a tenir en compte els blocs de negació condicionada addicionals i els qubits per a emmagatzemar els signes dels operands.

3.6 Divisió de naturals i enters

La divisió no és una operació de gaire interès pel que fa a la factorització o càlcul del logaritme discret, i conseqüentment no ha estat un tòpic estudiat en la investigació de la computació quàntica. Les publicacions centrades en aquesta operació són escasses, tant és així que no es va publicar un disseny complet per al càlcul del quocient i residu fins el 2011 ([29]), el qual es basa en l'algorisme clàssic de divisió per resta condicionada.

Malauradament, no existeixen (a dia d'avui) millors algorismes per a dividir naturals, de manera que l'única opció és millorar els circuits existents per a reduir la profunditat o complexitat espacial. El disseny més recent, i amb menors requeriments espacials i temporals, va ser publicat el 2019 ([30]), i serà el que mostrarem a continuació amb petites modificacions per a que sigui aplicable de manera més general.

El prototips generals d'aquesta operació es mostren a la figura 40. Tot i que tan sols presentem un disseny que actua sobre un dels operands, hem donat eines prèviament per a obtenir una implementació de la versió que actua sobre un registre addicional.



(a) Sobre un registre addicional. (b) Sobre un operand.

Figura 40: *Prototips de l'operador DIV per a calcular el quocient i residu de la divisió entera de dos registres de n qubits sobre un registre addicional (a) i sobre un operand (b).* **Font:** Elaboració pròpia mitjançant Q-Circuit.

En general, la divisió entera de dos naturals $\frac{a}{b}$ es pot representar com:

$$a = qb + r$$

On q és el quocient i r el residu, el qual compleix $0 \leq r < b$.

Podem descompondre el producte qr tal i com hem fet a la secció anterior:

$$\begin{aligned} a &= qb + r \\ &= (q_{n-1}q_{n-2}\dots q_1q_0)b + r \\ &= (2^{n-1}q_{n-1} + 2^{n-2}q_{n-2} + \dots + 2^1q_1 + 2^0q_0)b + r \\ &= q_{n-1}2^{n-1}b + q_{n-2}2^{n-2}b + \dots + q_12^1b + q_02^0b + r \end{aligned}$$

El qual es pot representar com una suma normal:

$$\begin{array}{cccccccccc}
 & & & & r_{n-1} & r_{n-2} & \dots & r_1 & r_0 & \\
 & & & & b_{n-1} & b_{n-2} & \dots & b_1 & b_0 & \cdot q_0 \\
 & & & b_{n-1} & b_{n-2} & \dots & b_1 & b_0 & & \cdot q_1 \\
 & & b_{n-1} & b_{n-2} & \dots & b_1 & b_0 & & & \cdot q_{n-2} \\
+ & b_{n-1} & b_{n-2} & \dots & b_1 & b_0 & & & & \cdot q_{n-1} \\
\hline
a_{2n-1} & a_{2n-2} & a_{2n-3} & a_{2n-4} & \dots & a_3 & a_2 & a_1 & a_0 &
\end{array}$$

A partir dels bits de a podem inferir els valors dels bits de q ja que per exemple, si a és major a $b2^{n-1}$, sabem que q_{n-1} serà 1. Amb aquest coneixement, podem restar $q_{n-1}2^{n-1}b$ de a, obtenint una suma:

$$\begin{array}{cccccccccc}
 & & & & r_{n-1} & r_{n-2} & \dots & r_1 & r_0 & \\
 & & & & b_{n-1} & b_{n-2} & \dots & b_1 & b_0 & \cdot q_0 \\
 & & & b_{n-1} & b_{n-2} & \dots & b_1 & b_0 & & \cdot q_1 \\
 & & & & & & & & & \\
+ & & b_{n-1} & b_{n-2} & \dots & b_1 & b_0 & & & \cdot q_{n-2} \\
\hline
0 & a'_{2n-2} & a'_{2n-3} & a'_{2n-4} & \dots & a'_3 & a'_2 & a'_1 & a'_0 &
\end{array}$$

Podem repetir el procediment per a inferir el valor de q_{n-2}, i així successivament fins a trobar tot q, quan podrem afirmar que el valor restant de a és igual al residu r.

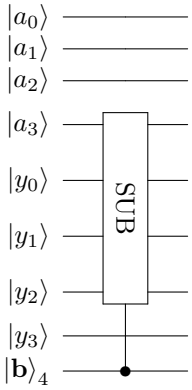
L'algorisme bàsic es basa, doncs, en realitzar restes condicionades al fet que el valor restant del registre a sigui major a b multiplicat per una potència de dos. Com que cal una resta per a comparar dos registres, la implementació que mostrarem realitza una resta incondicional seguida d'una suma condicionada.

Assumim que tenim els registres $|\mathbf{a}\rangle_4$ i $|\mathbf{b}\rangle_4$ amb els operands positius en Complement a 2, i un $|\mathbf{0}\rangle_4$ addicional pel resultat:

$$|\mathbf{y}\rangle_n = \begin{array}{r} |\mathbf{a}\rangle_4 \ \text{——} \\ |\mathbf{0}\rangle_4 \ \text{——} \\ |\mathbf{b}\rangle_4 \ \text{——} \end{array}$$

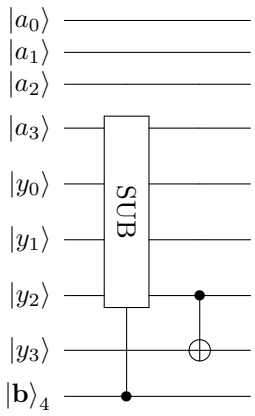
En tot moment operarem amb un registre de tamany n, el qual començarà sent $y_{n-2}y_{n-3}\dots y_0a_{n-1}$, el qual és equivalent a la part entera de $\frac{a}{2^{n-1}}$. A cada iteració anirem movent aquest registre a la dreta, de manera que a la iteració i treballarem amb els n qubits $y_{n-1-i}\dots y_0a_{n-1}\dots a_{n-i}$, els quals representen la part entera de $\frac{a}{2^{n-i}}$. Si en qualsevol d'aquestes iteracions, es compleix que b és major o igual a aquest registre, sabem que podem restar $b2^{n-i}$ de a, i que per tant $q_{n-i} = 1$.

Primerament restem b al registre $y_2y_1y_0a_3$:



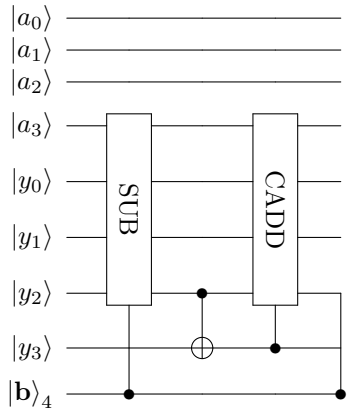
Cal recordar que hem usat la notació condicionada ja que l'operador **SUB** no altera els qubits d'un dels operands.

Podem conèixer el resultat de la comparació copiant el signe del valor obtingut de la resta sobre y_3. Cal recordar que estem operant amb enters en Ca2, i per tant el qubit que conté el signe del registre resultat és y_2.

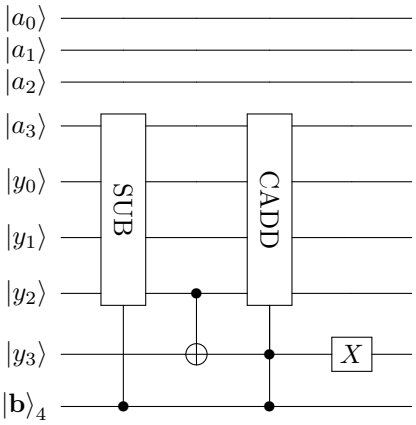


Si aquest resultat és negatiu, caldrà desfer la resta, ja que la condició per a realitzar-la era que b fos major o igual al valor del registre.

Podem realitzar aquesta operació mitjançant qualsevol implementació del bloc **CADD** de suma condicionada a un qubit, incloent les dues opcions presentades anteriorment.

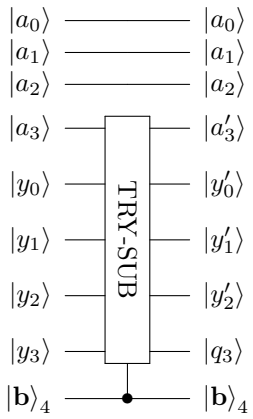


Finalment calculem el bit del quocient pertinent, en aquest cas q_3. Com s'ha mencionat abans, aquest val 1 si ha estat possible realitzar la resta i 0 si no, de manera que tan sols cal negar el qubit y_3.



És a dir, realitzem la suma de b sobre el registre condicionada a y_3.

Aquest conjunt de portes es representaran combinades en el bloc **TRY-SUB**:



On el registre resultat $y'_2y'_1y'_0a'_3$ és igual a l'original si no s'ha fet la resta, i és igual a la diferència si s'ha pogut fer.

En general, aquest bloc actua sobre dos enters en Ca2 de n qubits i un qubit pel resultat. Si el primer operand és major al segon, es realitza la resta i el qubit de sortida val 1, en cas contrari no es realitza cap resta i el qubit de sortida és 0.

Podem concatenar n d'aquests blocs per a calcular els n qubits del quocient, el qual es mostra a la figura 41.

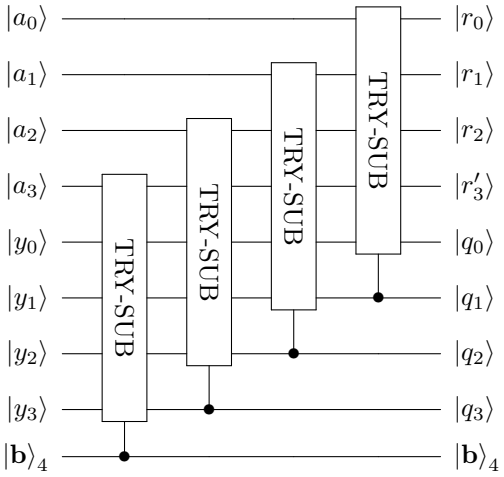


Figura 41: *Divisió de naturals basada en l'algorisme tradicional per a calcular el quocient i residu de dos enters positius en Ca2 de n qubits. El diagrama no inclou els qubits auxiliars requerits pels blocs individuals ja que són dependents de la implementació triada, però s'han donat circuits de possibles dissenys d'aquests blocs.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Un cop trobats tots els qubits del quocient, podem assegurar que el que quedi al registre $|\mathbf{a}\rangle_n$ serà el residu de la divisió.

Cal destacar que, tot i que el circuit està dissenyat per a treballar amb enters en complement a 2, aquests han de ser sempre positius per a assegurar un correcte funcionament de l'algorisme, de manera que caldria adoptar un esquema de negació condicionada similar a la multiplicació, però no caldria usar un qubit addicional per al signe. En cas que treballem amb naturals, si que caldrà afegir un $|0\rangle$ com a qubit de major pes a cada registre per a codificar-lo com a enter positiu.

Gràcies a la reversibilitat dels operadors quàntics, podem dissenyar circuits de producte d'enters positius en complement a 2 invertint qualsevol implementació de la divisió. Com que el circuit de divisió que hem dissenyat opera sobre un dels operands, podem donar una implementació trivial per al segon prototip d'operador **PROD** de la figura 42:

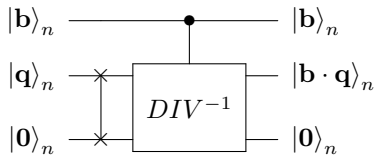


Figura 42: *Circuit que calcula el producte de naturals basat en inversió de la divisió.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Complexitat

El disseny conté un total de n portes **TRY-SUB**, cadascun format per 1 bloc **SUB**, 1 bloc **CADD**, una **CNOT** i una **X**.

En cas que es volgués minimitzar la complexitat temporal, mesurada com la profunditat del circuit, es podrien implementar els dos blocs compostos amb Carry-Lookahead per a assolir una profunditat logarítmica en cada iteració, per una profunditat total de $O(nlog\ n)$. Cada iteració requeriria també $O(2n)$ qubits auxiliars per a la resta i la suma condicional, els quals es poden reutilitzar ja que aquestes s'executen de manera seqüencial.

Si en canvi en desitgés optimitzar la memòria per a poder implementar el sistema en un ordinador quàntic el més bàsic possible, es podrien implementar les operacions amb Ripple-Carry o mitjançant la transformada de Fourier quàntica, ambdues de profunditat lineal en n. Aquest disseny tindria una profunditat total de $O(n^2)$ però no requeriria cap qubit auxiliar.

En general, el nombre de portes necessàries per a implementar cada iteració **TRY-SUB** és de $O(n)$, a menys que s'usi la transformada de Fourier quàntica per a la resta i suma condicionada, en qual cas caldrien $O(n^2)$ operadors per a realitzar i desfer aquesta transformació. Com a mínim, doncs, el nombre de portes requerides per a calcular el quocient i residu de la divisió de dos enters o naturals és de $O(N^2)$.

3.7 Exponenciació de naturals

El càlcul de potències és una operació normalment implementada en software i no en hardware, de manera que no caldria donar un circuit específic per a aquest càlcul ja que una versió bàsica pot ser trobada trivialment per a qualsevol individu amb uns mínims coneixements de programació. De totes maneres, el càlcul de potències és una operació vital en l'algorisme de factorització de Shor, i conseqüentment ha estat subjecte de multitud d'estudis i investigacions, els quals s'han centrat en optimitzar al màxim la profunditat els circuits d'exponenciació o reduir el nombre de qubits addicionals per a poder executar l'algorisme en els ordinadors quàntics "primitius"que tenim actualment.

Aquesta gran utilitat de l'exponenciació justifica dedicar una secció a l'estudi i presentació dels millors circuits i tècniques publicades per a realitzar aquesta operació. Com hem fet amb la resta d'operacions, proveirem dissenys que optimitzin o bé la profunditat o la memòria requerida per a realitzar el càlcul.

Com el amb producte i la suma, el primer circuit per a calcular potències va ser publicat un any després dels algorismes de Shor, el qual realitza el producte modular repetit basant-se en exponenciació logarítmica([2]). Les idees d'aquest document van ser expandides en publicacions dedicades a estudiar implementacions completes de l'algorisme de factorització, tot i que sovint s'usen mesures dels qubits ([31], [32]) per a condicionar l'aplicació de certes portes i eliminar computació. Aquesta tècnica sol ser acceptada ja que hem assumit en la definició del model que la mesura és una operació bàsica de tot ordinador quàntic, però en aquest document estem tractant de proporcionar dissenys reversibles més generalitzables, de manera que no presentarem aquestes opcions.

Estudis posteriors apliquen diverses tècniques per a millorar el rendiment del circuit, ja sigui mitjançat la QFT per a reduir la profunditat en operar sempre amb l'operand base transformat ([16], [33]) o mitjançant aritmètica Carry-Save per a estalviar-se el càlcul de cada suma individual durant els productes ([24], [34]). Cal destacar que, tal i com ha passat amb les operacions anteriors, la gran majoria d'aquests estudis han estat realitzats directa o indirectament en relació amb els algorismes de Shor, de manera que tracten d'optimitzar el càlcul de a^b modular on a és un valor fixe, sovint a cost de la generalitat del circuit (per exemple [35]).

En aquest treball hem estudiat les idees bàsiques desenvolupades en els documents més prominents presentant circuits d'exponenciació i les hem adaptat per a realitzar l'operació de manera no modular sobre dos naturals qualsevol, amb prototip mostrat a la figura 43.

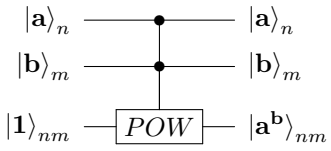


Figura 43: *Prototip de l'operador POW per a realitzar l'exponenciació d'un registe de n qubits en funció d'un altre registre de m qubts sobre un registre addicional de mn qubits.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

3.7.1 Circuit bàsic

L'exponenciació és definida com una multiplicació de la base repetida tantes vegades com l'exponent, de manera que podriem dissenyar un circuit que realitzés aquest producte i realitzés la subtracció de l'exponent de una unitat, repetit tantes vegades com calgui fins a que aquest sigui zero. El problema és que quan treballem amb una superoisició d'estats, pot ser que uns exponents siguin majors que altres, fent necessària la realització de 2^n productes condicionats al valor no nul de l'exponent. Afortunadament existeix un mètode anomenat exponenciació logarítmica, o exponenciació ràpida, la qual permet obtenir el mateix resultat amb $O(n)$ productes.

Per definició, a^b es pot representar com:

$$\begin{aligned} a^b &= a^{b_{m-1}b_{m-2}\ldots b_1 b_0} \\ &= a^{b_{m-1}2^{m-1}+b_{m-2}2^{m-2}+\cdots+b_1 2^1+b_0 2^0} \\ &= a^{b_{m-1}2^{m-1}} \cdot a^{b_{m-2}2^{m-2}} \cdot \ \ldots \ \cdot a^{b_1 2^1} \cdot a^{b_0 2^0} \end{aligned}$$

El qual és equivalent al producte de tot terme a^{2^i} condicionat a l'estat de b_i. Gràcies a aquesta representació, tan sols cal realitzar n multiplicacions per a obtenir cada terme i $w(m)$ per a multiplicar aquests termes amb el registre resultat, inicialitzat a 1. Per tant, per a dissenyar un circuit basat en aquesta implementació caldrà primer poder aplicar operadors de multiplicació condicionada **CPROD** i calcular el quadrat d'un natural a^2 **POW2**.

Producte condicionat de naturals

De la mateixa manera que hem mostrat un mètode general per a transformar qualsevol operador **ADD** en un de suma condicionada **CADD** mitjançant dos mètodes, detallarem com realitzar aquesta mateixa tasca per a qualsevol operador **PROD**. El prototip d'aquest és mostra a la figura 44.

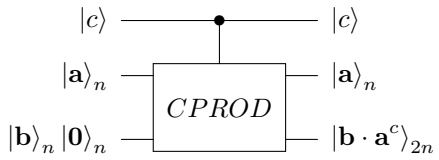


Figura 44: *Prototip de l'operador **CPROD** per a realitzar una multiplicació condicionada a l'estat d'un qubit.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

El primer mètode es basa en transformar un dels operands en un valor que no canviï el resultat, tal i com es mostra a al figura 45.

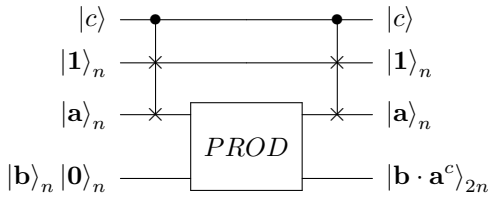


Figura 45: *Implementació de l'operador **CPROD** mitjançant anul·lació del primer operand per a multiplicar dos registres de n sobre el segon només quan el qubit $|c\rangle$ sigui $|1\rangle$.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

La complexitat d'aquest circuit és equivalent a la del bloc **PROD** que el compon.

El segon mètode es basa en condicionar totes les portes que canvien el resultat al qubit de condició, mitjançant un dels dos mètodes descrits anteriorment.

Circuit de càlcul del quadrat

De la mateixa manera que en el disseny de circuits de producte hem mostrat un operador capaç de construir tots els termes $a_i b 2^i$ de la suma final, per a realitzar el càlcul de potències mitjançant exponenciació logarítmica cal una porta capaç de calcular tots els termes $a^{b_i 2^i}$. Aquest circuit l'anomenarém **POW2**, i tindrà el prototip mostrat a la figura 46.

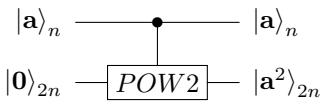
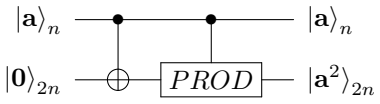
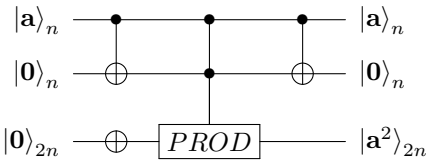


Figura 46: *Prototip de l'operador POW2 per a calcular el quadrat d'un natural de n qubits sobre un registre addicional.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Malauradament no ha ha dissenys de circuits que realitzin aquesta operació sobre el mateix operand, de manera que caldrà usar un registre addicinal. Podem usar qualsevol implementació de **PROD** per a realitzar aquesta oepració copiant el valor de l'operand i realitzant la multiplicació habitual:



O amb les versions que actuen sobre un registre addicional:



Tots amb complexitats temporals i espacials equivalents a les dels dissenys de **PROD** en els quals es basin, més els $O(n)$ qubits auxiliars per als segons.

Aquestes implementacions es basen en una operació més general que la realització d'un quadrat, de manera que no seria erroni pensar que es podria obtenir un disseny de circuit de multiplicació optimitzat per a qual els dos operands són iguals. En efecte, diversos documents ([36], [37]) han estudiat l'optimització d'aquesta operació en particular, obtenint circuits amb aproximadament la meitat de portes i profunditat.

El producte d'un natural $a = a_{n-1} \dots a_1 a_0$ per sí mateix es pot reescriure:

$$\begin{aligned}\mathbf{a}\cdot\mathbf{a} &= \mathbf{a}\cdot(a_{n-1}\dots a_1a_0)\\ &= \mathbf{a}\cdot(a_{n-1}2^{n-1}+\cdots+a_12^1+a_02^0)\\ &= \sum_{i=0}^{n-1} a_i2^i\mathbf{a}\\ &= \sum_{i=0}^{n-1} a_i2^i(a_{n-1}2^{n-1}+\cdots+a_12^1+a_02^0)\\ &= \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_ia_j2^{i+j}\end{aligned}$$

Cal veure que els índexs i i j són simètrics, de manera que cada terme on $i \neq j$ apareix dues vegades en el sumatori i cada terme $i = j$ una. Aquest fet es pot visualitzar fàcilment si es representen tots els termes del producte en una taula:

	a_02^0	a_12^1	a_22^2	$\cdots$	$a_{n-3}2^{n-3}$	$a_{n-2}2^{n-2}$	$a_{n-1}2^{n-1}$
a_02^0	a_02^0	$a_1a_02^1$	$a_2a_02^2$	$\cdots$	$a_{n-3}a_02^{n-3}$	$a_{n-2}a_02^{n-2}$	$a_{n-1}a_02^{n-1}$
a_12^1	$a_0a_12^1$	a_12^2	$a_2a_12^3$	$\cdots$	$a_{n-3}a_12^{n-2}$	$a_{n-2}a_12^{n-1}$	$a_{n-1}a_12^n$
a_22^2	$a_0a_22^2$	$a_1a_22^3$	a_22^4	$\cdots$	$a_{n-3}a_22^{n-3}$	$a_{n-1}a_22^n$	$a_{n-1}a_22^{n+1}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
$a_{n-3}2^{n-3}$	$a_0a_{n-3}2^{n-3}$	$a_1a_{n-3}2^{n-2}$	$a_2a_{n-3}2^{n-1}$	$\cdots$	$a_{n-3}2^{2n-6}$	$a_{n-2}a_{n-3}2^{2n-5}$	$a_{n-1}a_{n-3}2^{2n-4}$
$a_{n-2}2^{n-2}$	$a_0a_{n-2}2^{n-2}$	$a_1a_{n-2}2^{n-1}$	$a_2a_{n-2}2^n$	$\cdots$	$a_{n-3}a_{n-2}2^{2n-5}$	$a_{n-2}2^{2n-4}$	$a_{n-1}a_{n-2}2^{2n-3}$
$a_{n-1}2^{n-1}$	$a_0a_{n-1}2^{n-1}$	$a_1a_{n-1}2^n$	$a_2a_{n-1}2^{n+1}$	$\cdots$	$a_{n-3}a_{n-1}2^{2n-4}$	$a_{n-2}a_{n-1}2^{2n-3}$	$a_{n-1}2^{2n-2}$

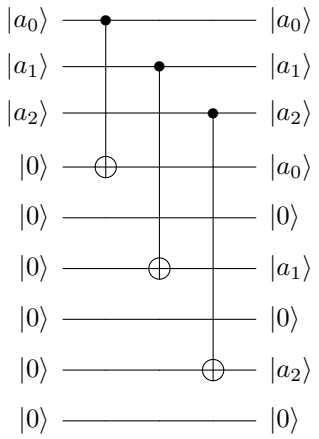
Aquesta simetria ens permet representar la suma de tots els termes com el doble del sumatori d'una de les meitats de la matriu més la suma dels termes de la diagonal. Podem forçar un ordre entre els índexs per a estalviar-nos aproximament la meitat d'aquests termes en l'equació:

$$\begin{aligned}\mathbf{a}\cdot\mathbf{a} &= \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_ia_j2^{i+j}\\ &= \sum_{i=0}^{n-1} a_i2^{2i} + 2\sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1} a_ia_j2^{i+j}\\ &= \sum_{i=0}^{n-1} a_i2^{2i} + \sum_{i=0}^{n-2} a_i2^{i+1}\sum_{j=i+1}^{n-1} a_j2^j\end{aligned}$$

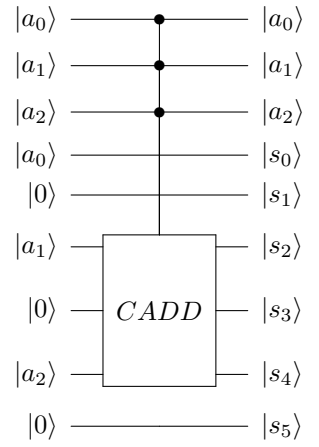
Assumim que volem calcular el quadrat d'un natural de 3 qubits. L'estat inicial del sistema és:

$|a_0\rangle$
$|a_1\rangle$
$|a_2\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$

Podem afegir cadascun dels termes $\sum_{i=0}^{n-1} a_i 2^{2i}$ al resultat mitjançant portes **CNOT**:

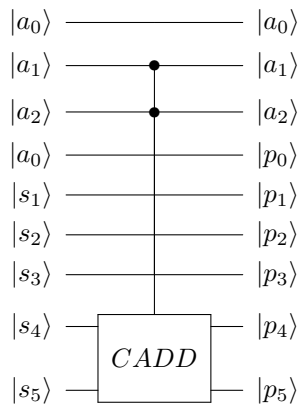


La resta de termes són equivalents a la suma d'un prefix de l'operand ponderat per una potència de dos. De manera similar a com s'ha fet en el disseny del circuit de divisió, realitzarem aquesta ponderació movent les posicions dels registres en realitzar la suma. Per exemple, per a sumar el primer dels termes $a_0 2^1 \sum_{j=1}^{2} a_j 2^j = a_0 2^2 a_2 a_1$, tan sols cal realitzar una suma de $a_2 a_1$ a partir del tercer qubit controlada per l'estat de $|a_0\rangle$:



On $|\mathbf{s}\rangle_{2n} = \sum_{i=0}^{n-1} a_i 2^{2i} + a_0 2^2 a_2 a_1$.

El segon terme a sumar és $a_1 2^2 \sum_{j=2}^{2} a_j 2^j = a_1 2^4 a_2$, el qual és equivalent a realitzar una suma de a_2 mogut tres posicions a l'esquerra condicionada a l'estat $|a_1\rangle$:



On $|\mathbf{p}\rangle_{2n}$ és igual a:

$$a_2 2^4 + a_1 2^2 + a_0 + 2a_0 \cdot a_2 a_1 2^1 + 2^2 a_1 \cdot a_2 2^2$$

$$a_2 00 \cdot 2^2 + a_1 0 \cdot 2^1 + a_0 + 2a_0 \cdot a_2 a_1 0 + 2^2 a_1 \cdot a_2 00$$

El qual és equivalent al valor del producte amb la reducció de termes, fàcilment visible si representem la multiplicació de la manera tradicional:

			a_2	a_1	a_0	
$\cdot$			a_2	a_1	a_0	
			$\underline{a_2}$	$\underline{a_1}$	$\mathbf{a_0}$	$\cdot a_0$
		$\underline{a_2}$	$\mathbf{a_1}$	a_0		$\cdot a_1$
$+$	$\mathbf{a_2}$	a_1	a_0			$\cdot a_2$

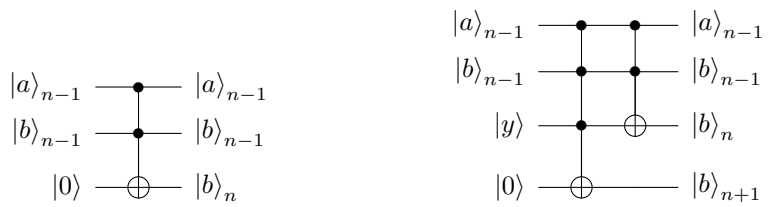
Els termes en negreta tan sols apareixen un cop a la suma, i són els individuals afegits directament en el primer pas, i els termes subratllats apareixen dues vegades en el sumatori final, i han estats afegits al producte en el segon pas.

Cal destacar que aquestes sumes condicionades poden generar ròssecs, els quals han de ser propagats estenent els circuits de suma. En general, a la iteració i de l'algorisme, es suma un prefixe de tamany $i-1$ sobre $i-1$ qubits del resultat, més un pel ròssec, usant qualsevol bloc ADD no modular estudiat. A causa de la suma inicial dels termes de la diagonal, en les iteracions on aquest qubit del ròssec sigui parell, indicant que ha estat modificat en el primer pas, caldrà propagar un ròssec addicional, el qual és assolible trivialment amb una **Toffoli-4** per a calcular aquest ròssec addicional i una **CNOT** per a modificar el ròssec original. Implementacions per als dos casos es mostren a la figura 47.

El circuit complet per a calcular el quadrat d'un natural de 3 qubits es mostra a la figura 48, el qual ha estat obtingut unint els tres subcircuits de suma de cada terme.

La complexitat d'aquest disseny depèn de la del bloc **CADD** usat, particularment de l'operador **ADD** que el forma.

Si tractem de minimitzar el nombre de qubits addicionals, podem usar la implementació Ripple-Carry d'aquest operador amb el condicionament de totes les portes per a condicionar-la per a obtenir operadors **CADD** de profunditat lineal i cap qubit temporal. La profunditat total és, per tant, la suma de les profunditats de cada sumador, les quals varien de n a 2, de manera que en total sumen $O(n^2)$. Tot i que la complexitat asimptòtica de la profunditat és igual que en el producte normal amb Ripple-Carry, a la pràctica és aproximadament la meitat ja que en la versió no optimitzada cada suma és sobre n qubits:



(a) Suma no modular normal sobre un qubit a $|0\rangle$.

(b) Suma no modular sobre un qubit a un estat qualsevol $|y\rangle$ amb ròssec addicional.

Figura 47: *Variants de ADD per a la suma usual sobre un ròssec inicialitzat a $|0\rangle$ (a) o sobre un ròssec en un estat arbitrari (b).* **Font:** Elaboració pròpia mitjançant Q-Circuit.

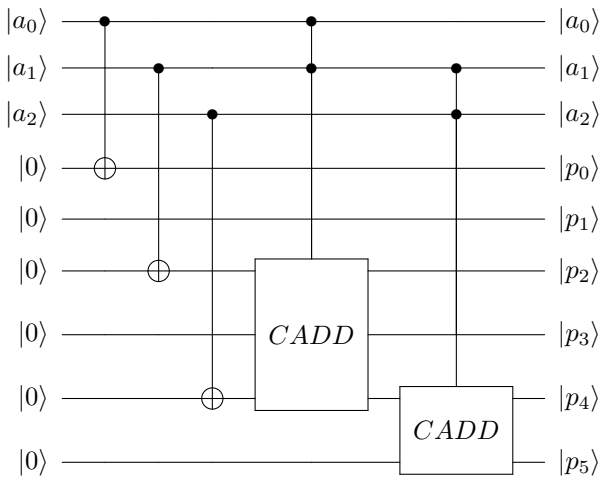


Figura 48: *Implementació de l'operador POW2 mitjançant l' optimització del circuit de producte de naturals per a operands iguals.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

$$\sum_{i=0}^{n-1} n \approx n^2$$

En canvi, en el nostre disseny, la profunditat de cada suma es va reduint ja que cada operació treballa amb menys qubits:

$$\sum_{i=0}^{n-1} i \approx \frac{n^2}{2}$$

Amb sumadors Ripple-Carry, doncs, el circuit no usa cap qubit addicional, tal i com era el cas amb el producte habitual.

Si en canvi es vol optimitzar la profunditat, es poden usar sumadors Carry-Lookahead per a sumar en $O(log\ n)$ usant $O(n)$ qubits temporals, els quals poden ser reutilitzats ja que les sumes no són paral·leles.

La profunditat total serà, doncs:

$$\sum_{i=0}^{n-1} log(i) = O(n\ log\ n)$$

De nou, mateixa complexitat que en el producte normal però menor a la pràctica gràcies a la reducció de termes a sumar.

Un lector perspicaç se n'adonarà que aquest esquema de suma repetida pot ser optimitzat mitjançant aritmètica Carry-Save. Efectivament, es pot dissenyar una versió Carry-Save d'aquest circuit seguint el mateix procediment mostrat a la secció del càlcul del producte de profunditat $O(log\ n)$ amb $O(n^2)$ qubits temporals. S'ha omès aquesta versió ja que en les seccions properes s'estudiarà en detall com aprofitar Carry-Save per a calcular potències eficientment.

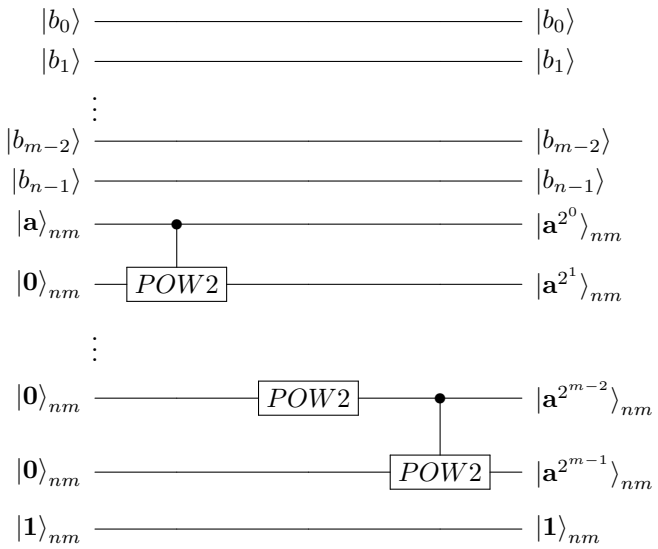
Circuit d'exponenciació bàsic

El circuit d'exponenciació bàsica es basa en inicialitzar el registre resultat a $|\mathbf{1}\rangle_{nm}$, anar calculant de manera seqüencial cada terme a^{2^i} i afegint-lo al producte final mitjançant una multiplicació condicionada a l'estat de $|b_i\rangle$.

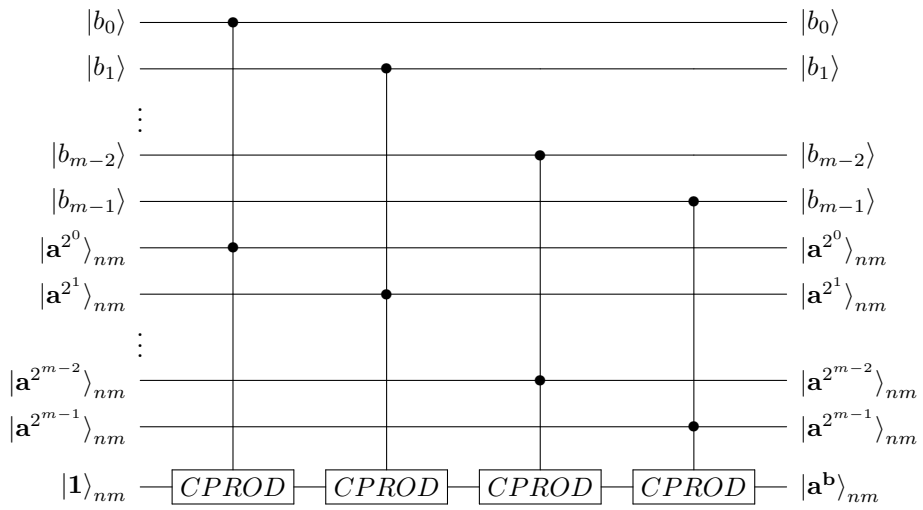
Assumim que comencem en l'estat:

$$\begin{aligned} |\mathbf{b}\rangle_n\ &— \\ |\mathbf{a}\rangle_n\,|\mathbf{0}\rangle_n\ &— \\ |\mathbf{1}\rangle_{n^2}\ &— \end{aligned}$$

En l'algorisme bàsic del producte teníem un operador capaç de calcular cada terme sobre el mateix registre, però el nostre circuit per a calcular el quadrat opera sobre un registre addicional, de manera que caldrà obtenir cadascun dels termes en un registre diferent:



Seguidament podem afegir tots els termes realitzant productes de cada potència del primer operand condicionats als estats dels qubits del segon:



Desfent els càlculs dels termes del producte podem obtenir el resultat desitjat. El circuit complet, obtingut unint els tres passos, es mostra a la figura 49.

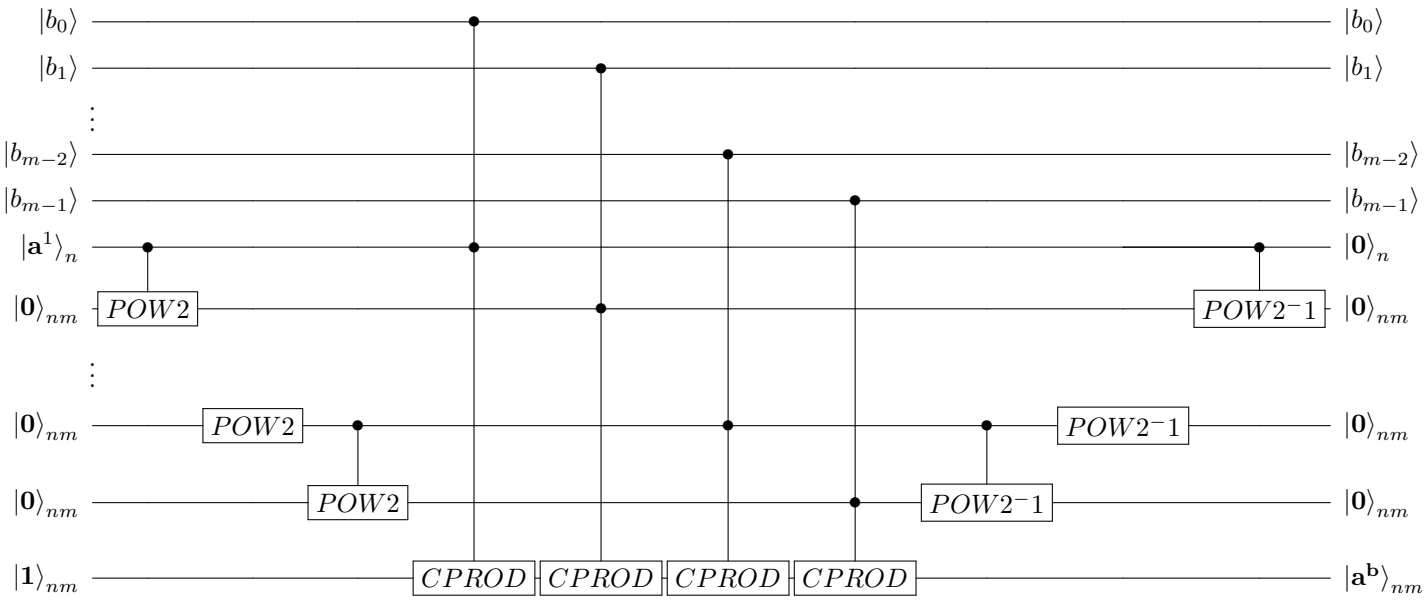


Figura 49: *Implementació bàsica de l'operador POW per a realitzar l'exponenciació no modular a^b de dos registres de $|\mathbf{a}\rangle_n$ i $|\mathbf{b}\rangle_n$ sobre un registre addicional.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Complexitat

En general, el circuit està format per $2m$ blocs **POW2** i m blocs **PROD** no modulars.

Si es tracta de minimitzar el nombre de qubits temporals, es poden usar les implementacions Ripple-Carry o QFT dels operadors. Tant les portes **POW2** com **PROD**, implementades amb aquests mètodes, tenen profunditats $O((nm)^2)$, de manera que la profunditat total és de $O(n^2m^3)$. Aquests operadors poden ser implementats per a no requerir cap qubit addicional, de manera que en total tan sols calen els $O(nm(m-1))$ per a emmagatzemar els $m-1$ termes addicionals del producte, cadascun de mida nm.

Si en canvi es vol optimitzar la complexitat temporal del circuit, es poden usar les implementacions Carry-Lookeahead dels blocs **CPROD** amb profunditats $O(nm\ log\ nm)$ per a una profunditat total de $O(nm^2\ log\ nm)$. Cadascun d'aquests operadors usa $O(n)$ qubits temporals, els quals poden ser reutilitzats, deixant el còmput total de qubits auxiliars a $O(nm(m-1))$, de nou, a causa dels registres pels termes del producte.

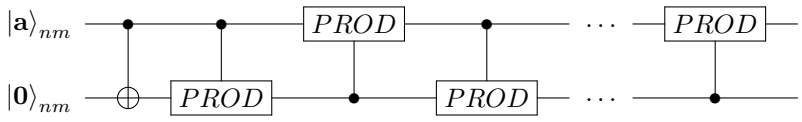
3.7.2 Optimització de memòria amb codificació de Fibonacci

En tots els apartats hem donat un circuit bàsic per a realitzar cada operació ja que aquests solen ser els que requereixen un menor nombre de qubits addicionals, però en el cas de l'exponenicació hem obtingut un circuit que requereix de l'ordre de m registres de nm qubits temporals, resultat que deixa molt a desitjar, sobretot en comparació als costos espacials de les implementacions bàsiques de les altres operacions.

El culpable d'aquest nombre elevat de qubits addicionals és l'operador **POW2**, pel qual no existeixen dissenys que modifiquin l'operand, de manera que cada aplicació d'aquest s'ha de realitzar sobre un nou registre. Un podria pensar que es podrien anar reutilitzar dos registres, seguint la seqüència:

$$\begin{matrix} a \\ 0 \end{matrix} \to \begin{matrix} a \\ a^2 \end{matrix} \to \begin{matrix} 0 \\ a^2 \end{matrix} \to \begin{matrix} a^4 \\ a^2 \end{matrix} \to \begin{matrix} a^4 \\ 0 \end{matrix} \to \cdots \to \begin{matrix} 0 \\ a^m \end{matrix}$$

El problema d'aquesta seqüència és que per a transformar a^{2n} a 0 cal a^n, de manera que no podem eliminar els resultats entremitjos. De totes maneres, podem usar una idea similar per a obtenir una seqüència diferent. Començant amb $|\mathbf{a}\rangle_{nm}$ i un sol registre temporal $|\mathbf{0}\rangle_{nm}$, anem realitzant multiplicacions alternatives entre aquests dos:



La seqüència de valors presents en els registres després de cada pas és:

$$\begin{matrix} a \\ 0 \end{matrix} \to \begin{matrix} a \\ a \end{matrix} \to \begin{matrix} a \\ a^2 \end{matrix} \to \begin{matrix} a^3 \\ a^2 \end{matrix} \to \begin{matrix} a^3 \\ a^5 \end{matrix} \to \begin{matrix} a^8 \\ a^5 \end{matrix} \to \cdots$$

Es pot provar que els exponents dels valors segueixen la seqüència:

$$\begin{matrix} F_1 \\ F_0 \end{matrix} \to \begin{matrix} F_1 \\ F_0 + F_1 = F_2 \end{matrix} \to \begin{matrix} F_1 + F_2 = F_3 \\ F_2 \end{matrix} \to \begin{matrix} F_3 \\ F_2 + F_3 = F_4 \end{matrix} \to \begin{matrix} F_3 + F_4 = F_5 \\ F_4 \end{matrix} \to \begin{matrix} F_5 \\ F_4 + F_5 = F_6 \end{matrix} \to \cdots$$

On F_n denota l'element n de la sèrie de Fibonacci, definida per la recurrència:

$$F_{n+1} = F_n + f_{n-1}$$

Amb casos base $F_0 = 0$ i $F_1 = 1$.

Anteriorment érem capaços de reconstruir el producte final gràcies a la descomposició de l'exponent a partir dels bits de la seva representació en base dos, el qual ens permetia definir el resultat com el producte del conjunt de termes a^{2^i} condicionats a l'estat del bit i de l'exponent:

$$a^b = a^{b_{m-1}2^{m-1}} \cdot a^{b_{m-2}2^{m-2}} \cdot \ \ldots \ \cdot a^{b_1 2^1} \cdot a^{b_0 2^0}$$

El conjunt de termes que tenim ara són de la forma a^{F_i}, de manera que si fóssim capaços de representar l'exponent b en funció d'aquests de la manera següent:

$$b = b'_1 F_1 + b'_2 F_2 + b'_3 F_3 + \ldots$$

Podríem calcular el producte final mitjançant el mateix procediment:

$$a^b = a^{b'_1 F_1} \cdot a^{b'_2 F_2} \cdot a^{b'_3 F_3} \ldots$$

El qual ens permetria estalviar-nos pràcticament tots els qubits temporal requerits pel disseny bàsic.

Per a obtenir el circuit final cal, doncs, dissenyar un operador capaç de calcular aquesta representació de l'exponent.

Codi de Fibonacci

Gràcies al teorema de Zeckendorf sabem que tot nombre natural b pot ser representat de manera única mitjançant l'anomenada codificació de Zeckendorf $c_1 c_2 c_3 \ldots c_k$ que compleixen:

$$b = \sum_{i=1}^{k} F_i c_k$$

On $c_1 c_2 c_3 \ldots c_k$ no conté dos 1 consecutius.

El nombre de bits necessaris per a representar aquesta codificació és major al del natural original, de manera que el primer pas és trobar aquesta k que definirà la mida del registre de la codificació.

No tindria sentit tenir en compte valors F_i majors al que podem representar en m bits, de manera que k es defineix com al valor màxim tal que $F_k < 2^m$. Com que m és un valor fix, el tamany del registre de l'exponent, podem precalcular aquesta k, permetent-nos també saber el tamany del registre de la codificació de Zeckendorf d'aquest.

A partir d'un natural b podem trobar la seva codificació mitjançant un algorisme voraç, el qual consisteix en anar afegint al conjunt el F_i més gran que sigui menor al nombre. Seguidament repetim el procés amb la diferència. Per exemple si tenim registres de 4 qubits, amb valor màxim de $2^4 - 1 = 15$, el conjunt de nombres de la sèrie de Fibonacci que poden formar part de les particions de qualsevol natural en aquest rang són:

$$1, 1, 2, 3, 5, 8, 13$$

Cal notar que tot i que es repeteix el 1, a la pràctica no es té en compte el primer ja que com hem esmentat, mai hi haurà dos elements seguits que formin part del conjunt que sumi al natural donat. Per tant k serà igual a 6:

$$b = 1c_1 + 2c_2 + 3c_3 + 5c_4 + 8c_5 + 13c_6$$

Si volem trobar la representació de 12, podem anar trobant els dígits d'un en un.

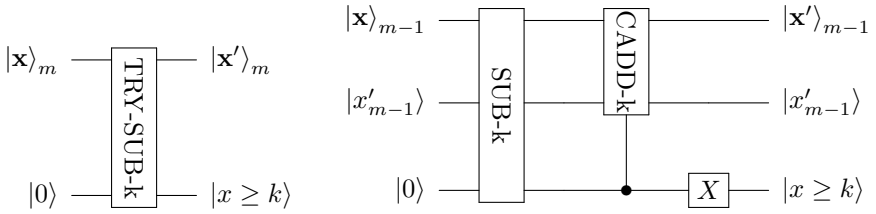
$$\begin{aligned} 12 < 13 &\implies c_6 = 0 \\ 12 \geq 8 &\implies c_5 = 1 \\ 12 - 8 < 5 &\implies c_4 = 0 \\ 12 - 8 \geq 3 &\implies c_3 = 1 \\ 12 - 8 - 3 < 2 &\implies c_2 = 0 \\ 12 - 8 - 3 \geq 1 &\implies c_1 = 1 \end{aligned}$$

Conseqüentment:

$$12 = 1 \cdot 1 + 2 \cdot 0 + 3 \cdot 1 + 5 \cdot 0 + 8 \cdot 1 + 13 \cdot 0$$

Dissenyar un circuit capaç de calcular aquesta codificació per qualsevol nombre és similar a la divisió en que cal realitzar restes condicionades i emmagatzemar en un qubit si ha estat possible realitzar-la o no.

Seguint el procediment mostrat a la secció de negació d'enters per a crear sumadors d'una sola unitat, podem dissenyar circuits capaços de realitzar una suma d'un valor constant sobre el mateix operand, els quals poden ser invertits per a calcular la resta. Aquests blocs **ADD-k** i **SUB-k** ens permetes construir un operador **TRY-SUB-k**, l'efecte i circuit dels quals es mostra a la figura 50, on x' és igual a x si $x < k$ o $x - k$ en cas contrari.



(a) Prototip. (b) Implementació.

Figura 50: *Prototip (a) i implementació (b) de l'operador TRY-SUB-k per a tractar de restar k d'un registre i emmagatzemar la diferència i si s'ha pogut realitzar l'operació sobre un qubit addicional.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

El circuit per a obtenir la codificació de Zeckendorf d'un natural de m qubits es pot definir doncs de manera recursiva a partir del seu prototip, mostrat a la figura 51.

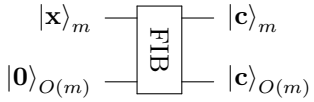
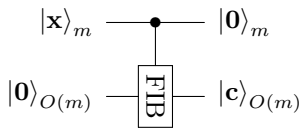
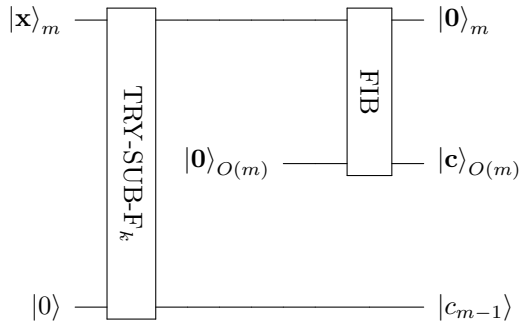


Figura 51: *Prototip de l'operador FIB per a calcular la codificació de Zeckendorf $|\mathbf{c}\rangle_{O(m)}$ d'un natural de m qubits.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

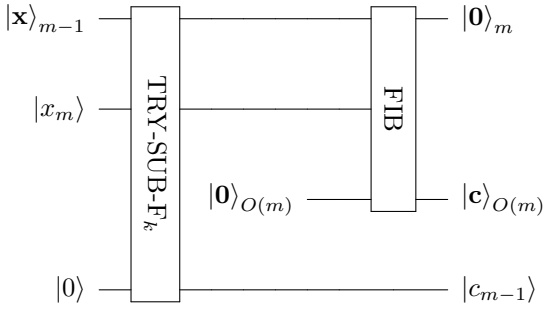
Si no ens importés gaire el nombre de qubits auxiliars podríem usar un prototip alternatiu, el qual ens facilita la definició recursiva:



Com s'ha detallant anteriorment, el procediment per a trobar cada c_i és el mateix, de manera que podem trobar el de pes major i repetir el procediment recursivament per a trobar tota la codificació:



El problema d'aquesta implementació és que no s'usen els qubits de $|\mathbf{x}\rangle_m$ que es van alliberant en reduïr-ne el seu valor, de manrea que usa més qubits temporals del que seria estrictament requerit. Un circuit que sí aprofités aquests qubits lliures tindria la forma:



L'única diferència d'aquesta implementació és que s'usa el nou qubit a $|0\rangle$ en el registre $|\mathbf{0}\rangle_{O(m)}$ del resultat de la crida recursiva. El problema és que la codificació de Zeckendorf té més bits que la codificació en base 2, de manera que no sempre tindrem un nou qubit nul després de cada iteració. Per exemple, si treballem amb registres de 3 qubits i tractem de codificar el natural $7 = 111_2$, la primera iteració tractarà de restar $F_5 = 5 = 101_2$ el és possible de manera que $c_5 = 1$ i el natural passa a ser $2 = 010_2$. Com que el qubit de major pes ha passat a 0 el podem usar en el registre pel resultat de la següent crida.

En general, per a alliberar un dels qubits del registre original, s'ha de complir, per tot x de m bits possible a codificar:

$$x - F_k \leq 2^{m-2}$$

En el pitjor dels casos, x serà màxim, amb valor $2^{m-1} - 1$, de manera que:

$$2^{m-1} - 1 - F_k \leq 2^{m-2}$$

$$F_k \geq 2^{m-1} - 2^{m-2} - 1 = 2^{m-2} + 1$$

Per tant, en la iteració en la que es calcula el valor de c_k, podem usar la versió que aprofita un qubit nul si i només si $F_k > 2^{m-2}$.

Tota recursió requereix de casos base, de manera que en donem un per $k \leq 1$. Aquesta versió de l'operador **FIB** tan sols accepta un registre de dos qubits $|\mathbf{x}\rangle_2$ i en calcula un de també dos qubits amb la codificació de Zeckendorf $|\mathbf{c}\rangle_2$ d'aquest natural de dos bits.

La representació de qualsevol natural x en binari és:

$$x = x_1 \cdot 2 + x_0$$

I la codificació mitjançant la sèrie de Fibonacci:

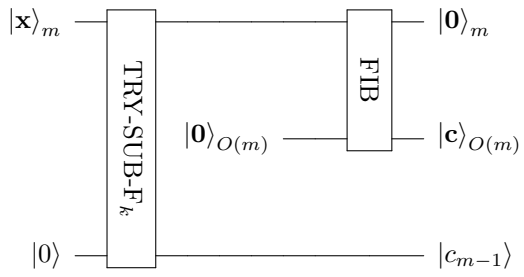
$$x = c_1 \cdot 2 + c_0$$

De manera que $c_i = x_i$ per $i \leq 2$, i el circuit que implementa l'operador **FIB** per a registres de 2 o menys qubits és:

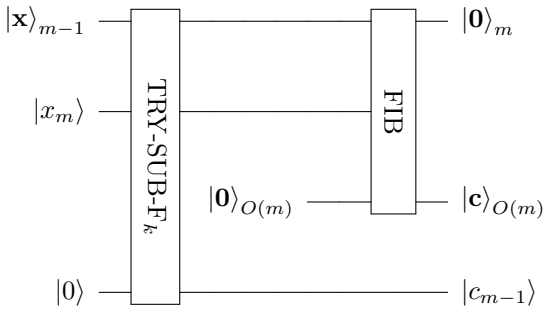
$$|x_0\rangle \text{ — } |c_0\rangle$$

$$|x_1\rangle \text{ — } |c_1\rangle$$

Per consistència, presentem les tres implementacions de l'operador **FIB** en la figura 52.



(a) Per $F_k \leq 2^{m-2}$.



(b) Per $F_k > 2^{m-2}$.

$|x_0\rangle$ — $|c_0\rangle$
$|x_1\rangle$ — $|c_1\rangle$

(c) Per $k \leq 2$.

Figura 52: *Implementacions de l'operador **FIB** per a casos on no podem alliberar cap qubit del natural (a), casos on si podem (b) i el cas base de la definició recursiva del circuit (c).* **Font:** Elaboració pròpia mitjançant Q-Circuit.

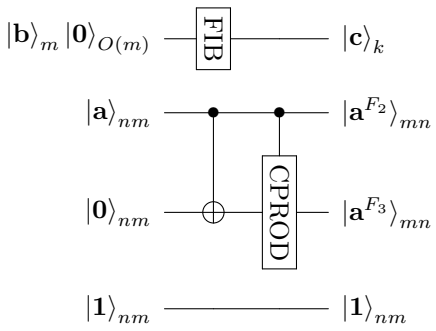
Circuit d'exponenciació

De la mateixa manera que hem calculat el resultat mitjançant el producte dels termes parcials a^{2^i} condicionats als bits de la representació en base 2 de l'exponent, podem calcular el mateix valor mitjançant el producte dels termes a^{F_k} condicionats als bits de la codificació de Zeckendorf.

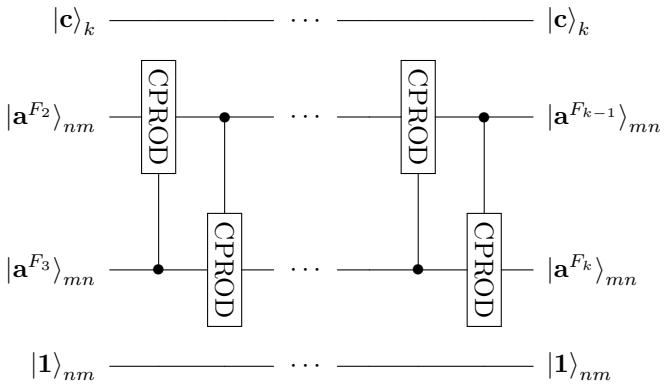
Assumim que comencem en l'estat:

$$|\mathbf{b}\rangle_m |\mathbf{0}\rangle_{O(m)} |\mathbf{a}\rangle_{nm} |\mathbf{0}\rangle_{nm} |\mathbf{1}\rangle_{nm}$$

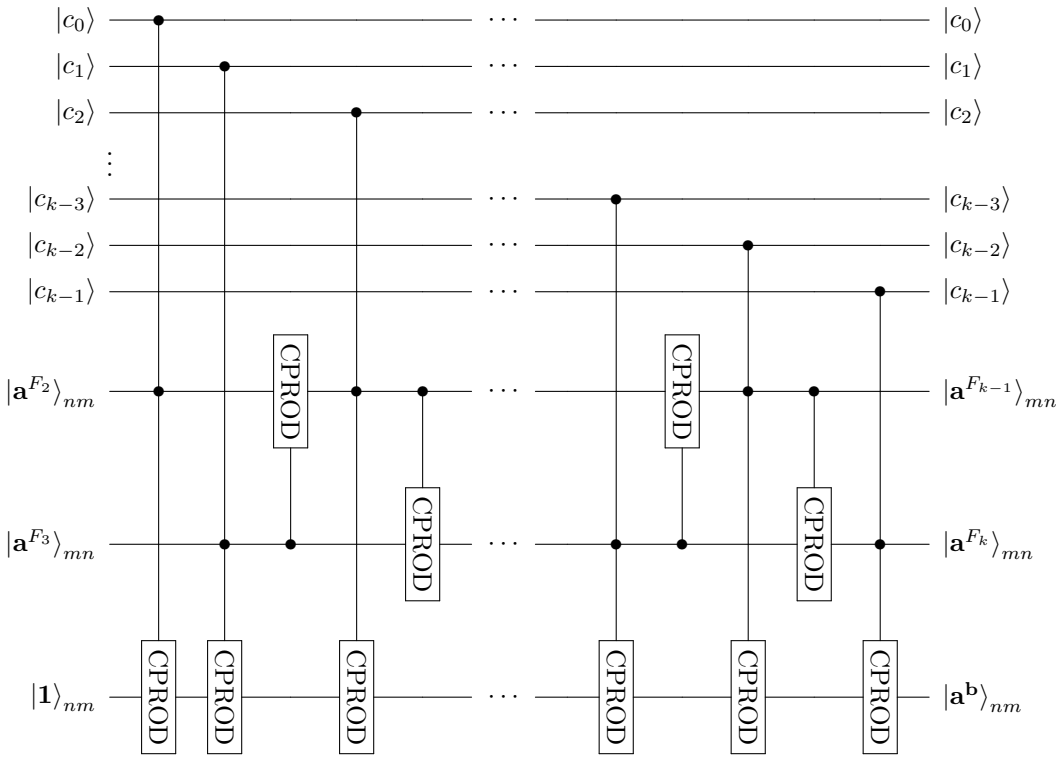
Comencem calculant la codificació de Zeckendorf de l'exponent i inicialitzant la sequència a a i a^2:



Donat k màxim tal que $F_k < 2^m$, calculem tots els termes $a^{F_{i+2}}$ per $0 \leq i < k$:



Finalment podem anar afegint els termes al producte final en funció dels estats dels qubits de la codificació de l'exponent:



Ara que ja tenim el resultat al registre corresponent, tan sols cal desfer la computació i retornar els operands i resistires temporals als seus estats originals. Aquest circuit per a calcular potències mitjançant la codificació basada en la sèrie de Fourier es mostra a la figura 53.

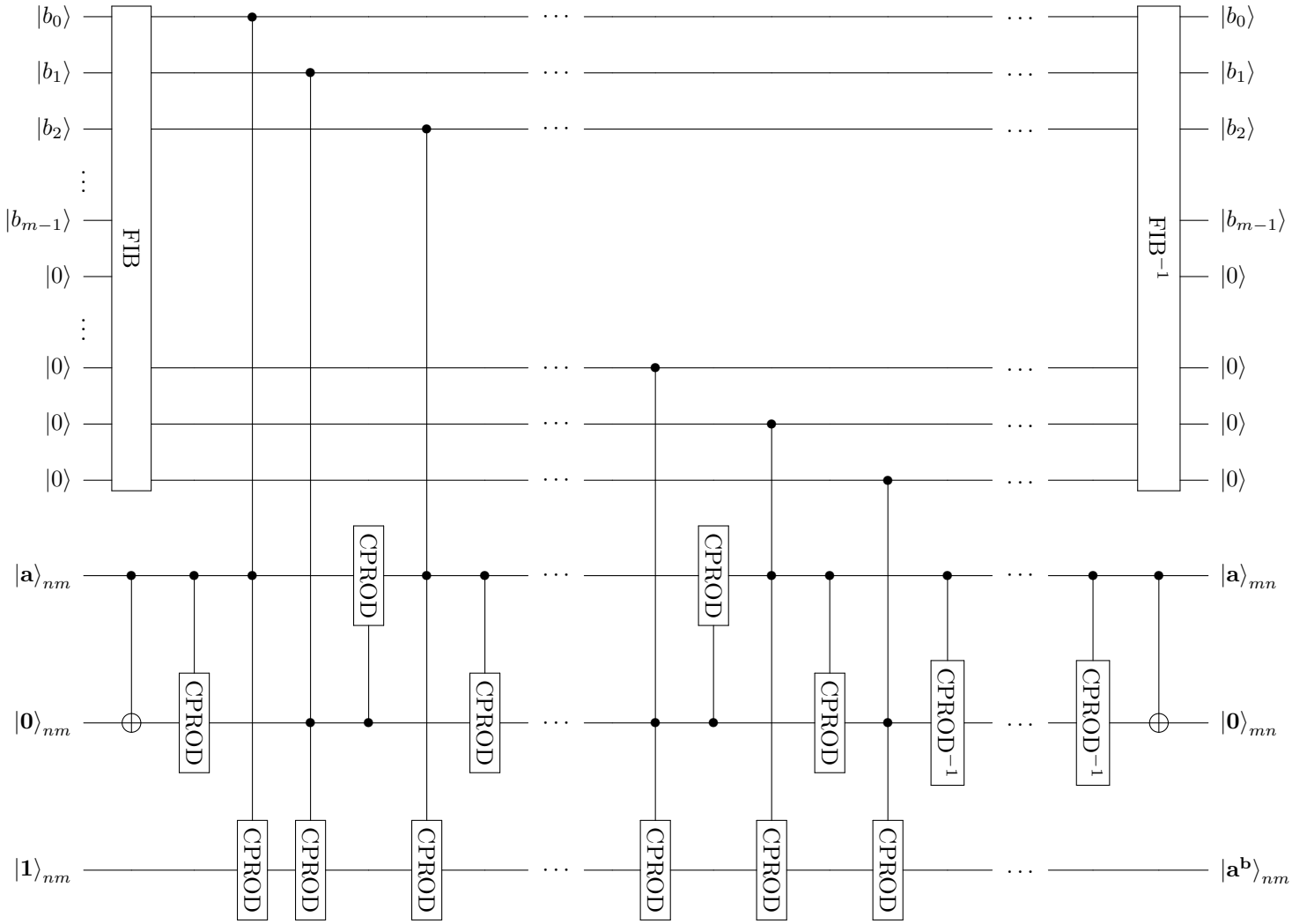


Figura 53: *Implementació de l'operador POW amb codificació de Fibonacci per a realitzar l'exponenciació d'un natural de n qubits per un altre de m.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Complexitat

Sabent que k és el nombre màxim tal que $F_k < 2^m$, la profunditat del circuit és de l'ordre de $O(1)$ portes **FIB** i $O(k)$ portes **CPROD** seqüencials. Per a obtenir una fita en funció de m, cal usar la formula de Binet, la qual dóna una expressió per al nombre i de la sèrie de Fourier:

$$F_n = \frac{1}{\sqrt{5}}(\phi^n - (-\phi)^{-n})$$

Per n gran, el segon terme tendeix a 0, el qual ens permet obtenir una bona aproximació del valor:

$$F_n \approx \frac{\phi^n}{\sqrt{5}}$$

Donat F_n, doncs, podem trobar n:

$$n \approx \frac{log(F_n) + \frac{1}{2}log(5)}{\log(\psi)}$$

Sabem que en el pitjor dels casos, el valor més gran de la sèrie que usem serà igual al valor màxim que podem representar en m bits, de manera que podem calcular aproximadament quin hauria de ser aquest element més gran de la sèrie de Fourier, o equivalentment, k:

$$k \approx \frac{log(2^m) + \frac{1}{2}log(5)}{\log(\psi)}$$

$$k \approx \frac{m + \frac{1}{2}log(5)}{\log(\psi)}$$

$$k = O(m)$$

Si es tracta de minimitzar la profunditat, podem usar les implementacions Carry-Lookahead dels operadors. Els blocs **FIB** apliquen $O(k) = O(m)$ blocs **TRY-SUB-k**, cadascun de profunditat $O(log\ m)$, cadascun dels quals requereix de $O(m)$ qubits temporals, els quals es poden reutilitzar. Els blocs **CPROD** tenen una profunditat de $O(nm\ log\ nm)$ i requereixen de $O(nm)$ qubits auxiliars.

En total, la profunditat del circuit és de $O(m\ log\ m + nm^2\ log\ nm) = O(nm^2\ lognm)$, igual que en la versió anterior. El nombre de qubits temporals, però, és ara de $O(m + nm) = O(nm)$, el qual és una reducció significant

Si es tracta de minimitzar el nombre de qubits addicionals, el qual era l'objectiu original de la modificació, es poden usar les implementacions QFT per les sumes i restes sense segon operand i les Ripple-Carry per quan hi ha un segon operand disponible, com en els blocs **CPROD**.

Els operadors **ADD-k** i **SUB-k** amb QFT tenen una profunditat de $O(m)$ i no usen cap qubit addicional, de manera que la porta **FIB** té una profunditat de $O(m^2)$ i tan sols usa els $O(m)$ qubits temporals per a emmagatzemar la codificació.

L'operador **PROD** està format per nm portes **ADD** condicionades, cadascuna amb profunditat lineal en nm i cap qubit addicional si s'implementen amb Ripple-Carry, per una profunditat total de $O(n^2m^2)$ i 0 qubits temporals requerits.

La profunditat total del circuit és doncs $O(n^2m^2)$, el qual és substancialment més gran que en el disseny anterior, però el nombre de qubits temporals requerits pels subcircuits s'ha reduït a 0, de manera que tan sols cal tenir registres per a emmagatzemar la codificació de l'exponent i pel registre temporal que usem per a emmagatzemar els termes de la sèrie de Fibonacci, el qual segueix sent $O(nm)$, però a la pràctica és aproximadament la meitat.

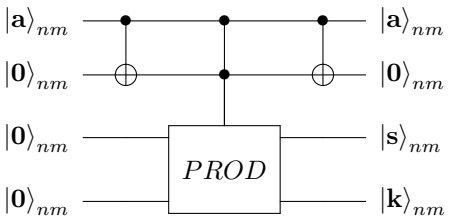
En comparació amb el circuit bàsic, però, el nombre de qubits addicionals s'ha reduït en un factor m, el qual és substancial.

3.7.3 Optimització de la profunditat amb Carry-Save

L'exponenciació no és més que una repetició del producte, i hem vist que mitjançant aritmètica Carry-Save podem obtenir un operador **PROD** capaç de multiplicar dos oeprands i obtenir-ne dos registres amb suma equivalent al producte desitjat en temps $O(log\ n)$. Podem, per tant, usar aquesta variació de l'operador de producte per a minimitzar la profunditat dels nostres circuits d'exponenciació.

Per a modificar el circuit d'exponenciació bàsic per a operar amb aritmètica Carry-Save, cal obtenir operadors **POW2** que treballin amb aquest mètode.

Un podria pensar que tan sols caldria usar els multiplicadors Carry-Save amb l'implementació bàsica de l'operador **POW2**:



On $s + k = a^2$.

Gràcies a no realitzar la suma dels dos resultats, aquest circuit té la profunditat desitjada de $O(log\ nm)$, però si hem de tornar a aplicar l'operació per a calcular a^4 a partir de $s + k = a^2$, haurem de realitzar aquesta suma igualment, el qual completament anul·la el benefici d'usar Carry-Save.

Cal, doncs, dissenyar un operador **POW2** que accepti un natural representat com a la suma de dos, amb prototip mostrat a la figura 54.

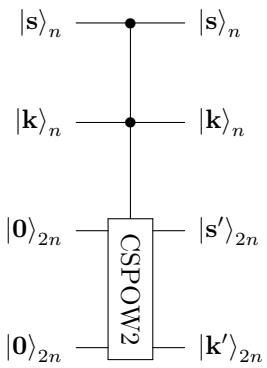


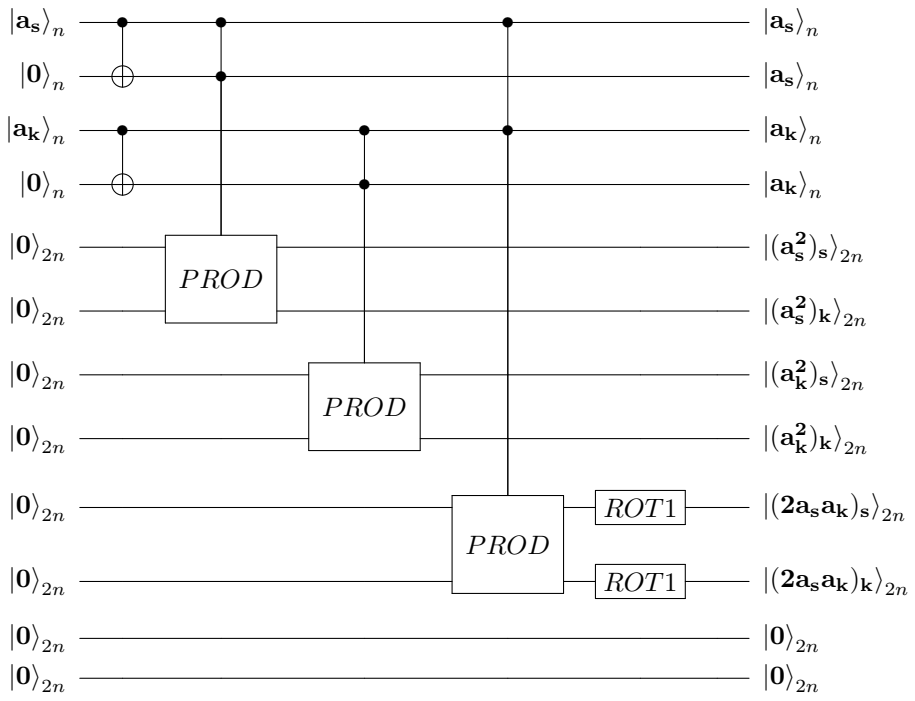
Figura 54: *Prototip de l'operador CSPOW2 per a calcular el quadrat d'un natural de n qubits sobre dos registres addicional en aritmètica Carry-Save. Els operands compleixen* $s' + k' = (s + k)^2$ **Font:** Elaboració pròpia mitjançant Q-Circuit.

Sabent que $a * s + k$, podem descompondre el quadrat en tres termes:

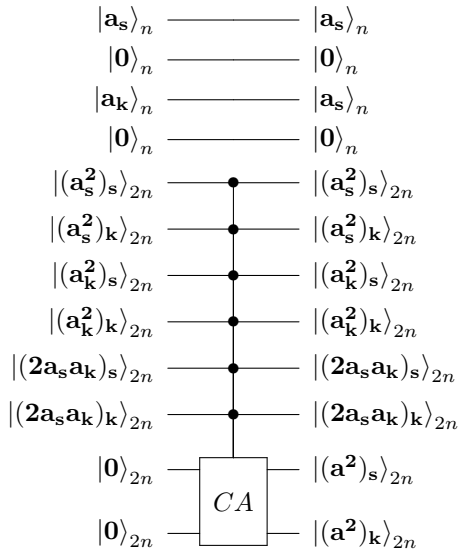
$$(s + k)^2 = s^2 + 2sk + k^2$$

Els quals podem calcular independentment en aritmètica Carry-Save. En els circuits posteriors usarem la notació x_s i x_k per a denotar la representació del valor de l'expressió x com a la suma de dos naturals.

El càlcul de a_s^2, $2a_s a_k$ i a_k^2 es pot realitzar, doncs, mitjançant el primer circuit presentat en aquesta secció per a calcular la segona potència d'un sol operand en Carry-Save:



El resultat desitjat és igual a la suma d'aquests 6 termes, però com que treballem en Carry-Save podem usar el bloc **CA** per a reduir la suma dels 6 termes a 2



Finalment tan sols cal desfer la computació dels termes que formen el quadrat per a reiniciar els qubits temporals. El circuit complet de l'operador **CSPOW2** per a calcular el quadrat en aritmètica Carry-Save es mostra a la figura 55.

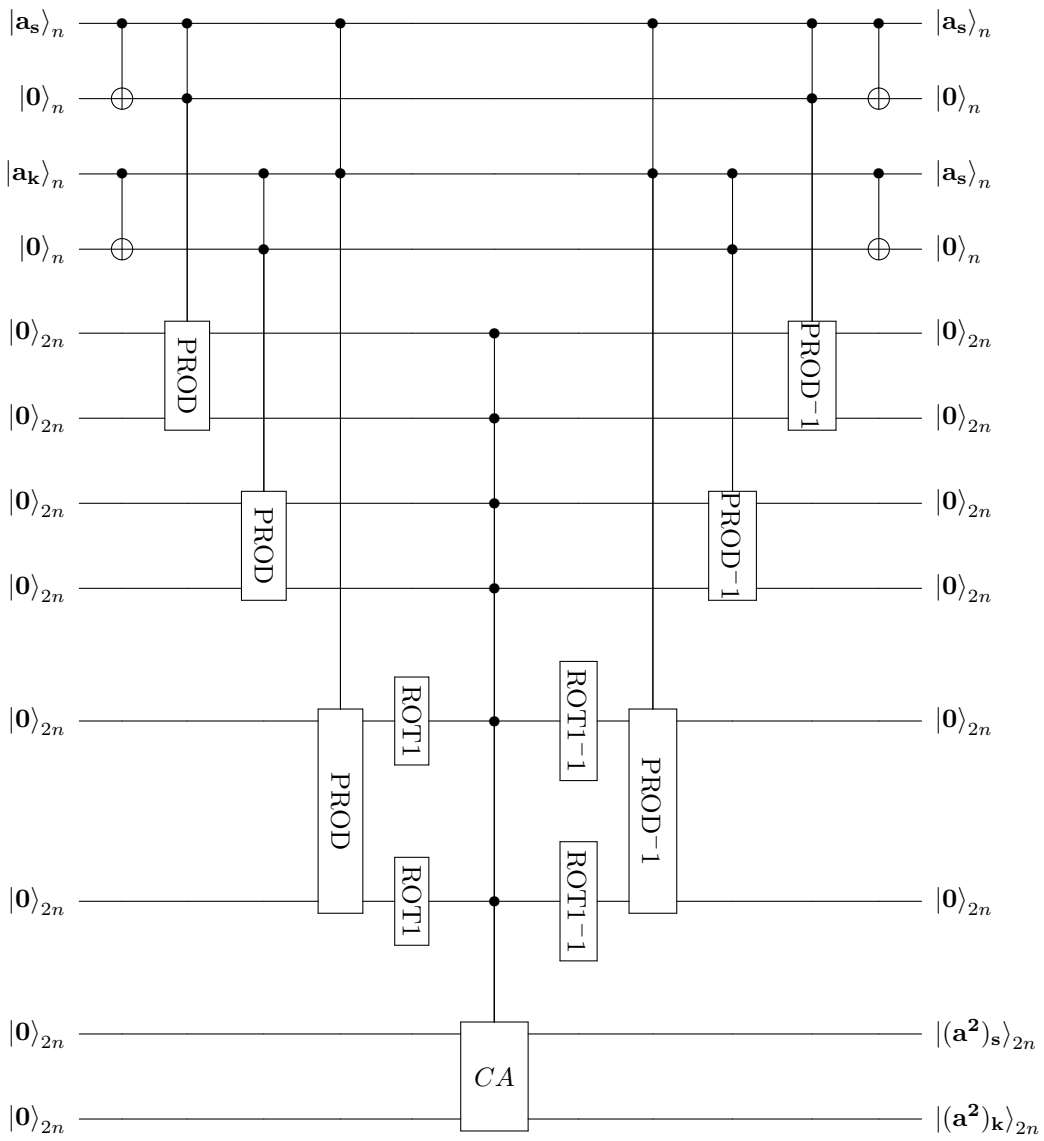
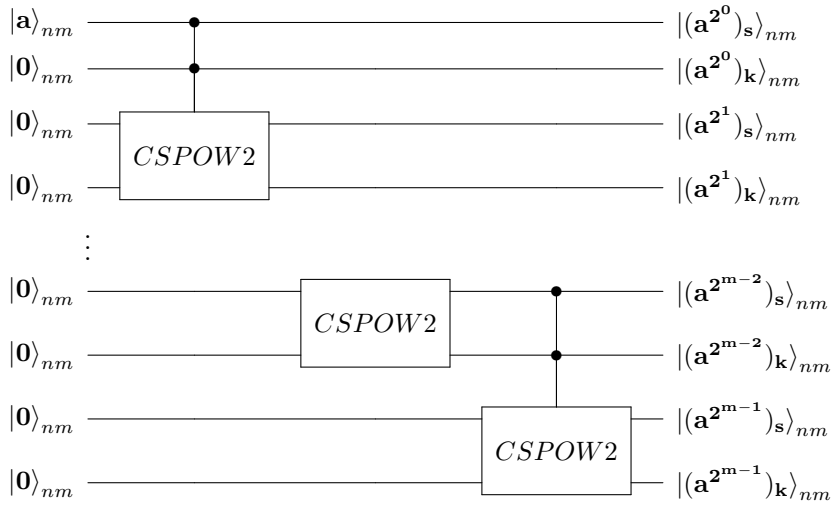


Figura 55: *Implementació de l'operador CSPOW2 per a calcular el quadrat d'un natural de n qubits sobre dos registres addicionals en aritmètica Carry-Save.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

La profunditat de cada bloc **PROD** Carry-Save és de $O(log\ n)$, i usa $O(n^2)$ qubits temporals. Els blocs **ROT** no requereixen da cap porta ni qubit addicional i el bloc **CA** té una profunditat constant en sumar sempre 6 registres i usa $O(n)$ qubits addicionals. En total, el circuit té una profunditat logarítmica en la mida dels registres i usa un nombre quadràtic de qubits temporals.

A partir d'aquest bloc, podem obtenir les representacions Carry-Save de tots els termes a^{2^i}:



El qual té una profunditat de $O(m\ log\ nm)$ i requereix de $O(n^2m^2)$ qubits temporals.

De la mateixa manera que hem hagut de dissenyar modificacions del bloc **POW2** per a operar amb naturals representats amb Carry-Save, cal obtenir un bloc **CSPROD** capaç de realitzar el producte de dos naturals en aquesta representació. Per consistència, el prototip d'aquest es mostra a la figura 56.

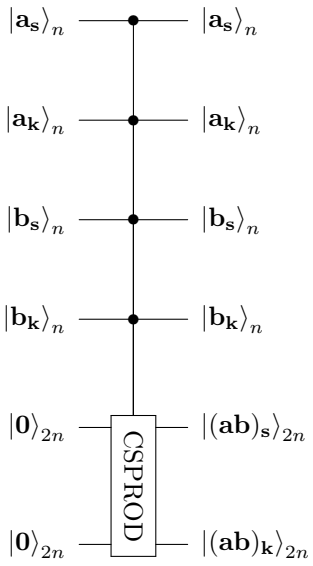
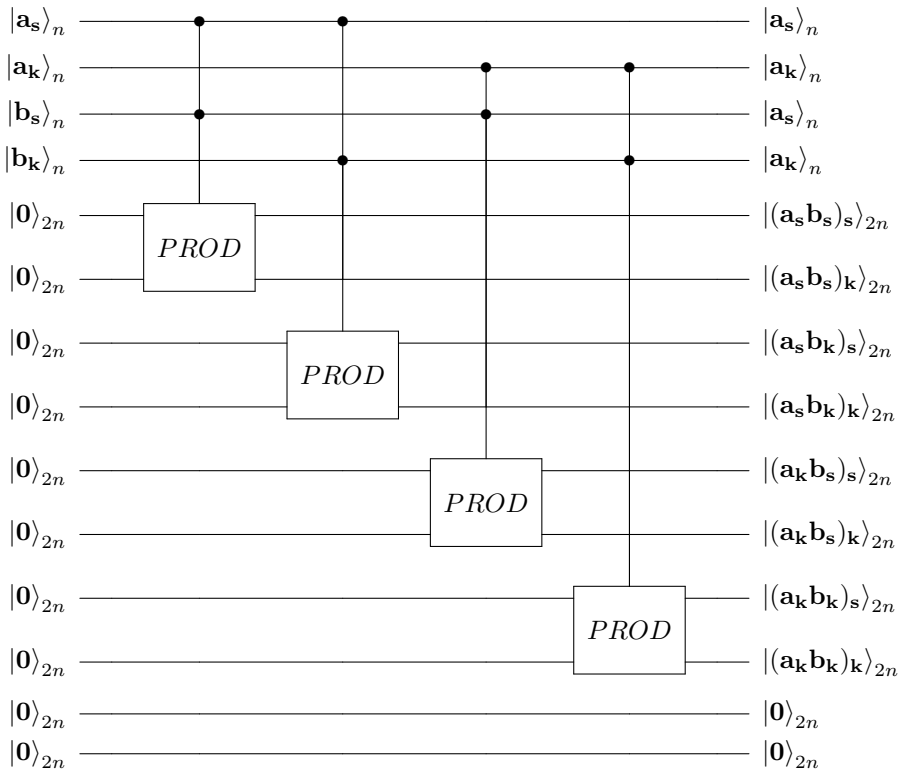


Figura 56: *Prototip de l'operador CSPROD per a calcular el producte de dos naturals de n qubits sobre dos registres addicional en aritmètica Carry-Save.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

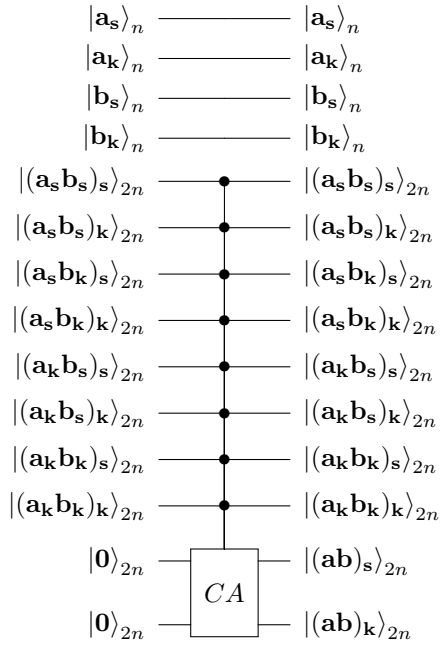
Assumint que els dos operands del producte són $a = a_s + a_k$ i $b = b_s + b_k$, podem descompondre el producte tal i com ho hem fet anteriorment:

$$\begin{aligned} a \cdot b &= (a_s + a_k) \cdot (b_s + b_k) \\ &= a_s b_s + a_s b_k + a_k b_s + a_k b_k \end{aligned}$$

De nou, podem usar el bloc **PROD** estudiat anteriorment per a calcular el producte Carry-Save de dos naturals en la seva representació normal, el qual ens permet calcular cadascun dels termes de la suma:



I combinar els 8 sumands en 2 gràcies al bloc **CA**.



El circuit complet de l'operador **CSPROD** capaç de calcular el producte de dos naturals completament mitjançant aritmètica Carry-Save es mostra a la figura 57.

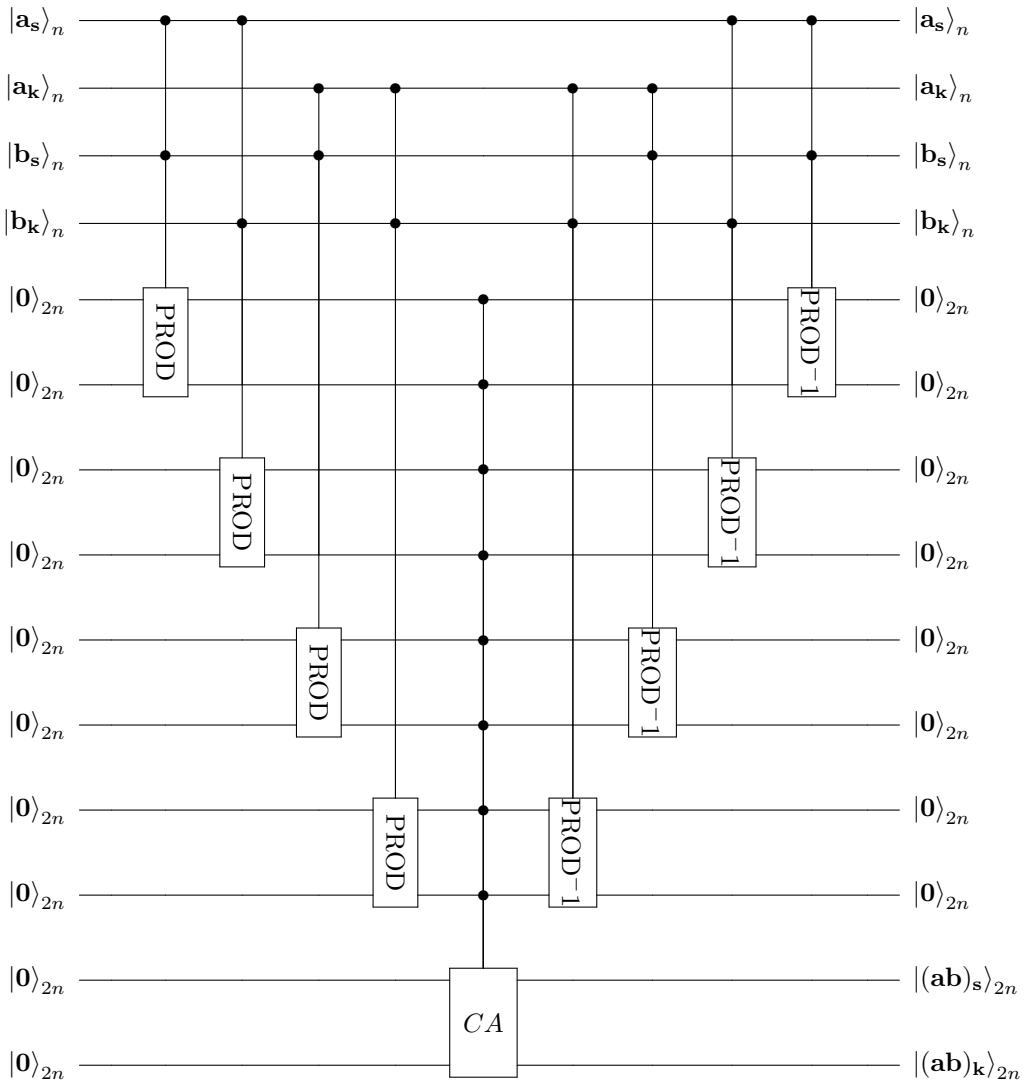
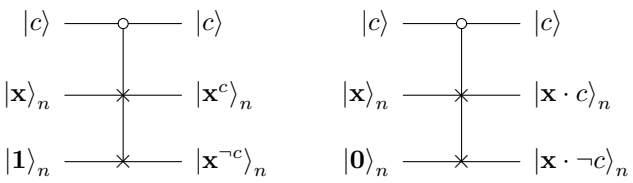


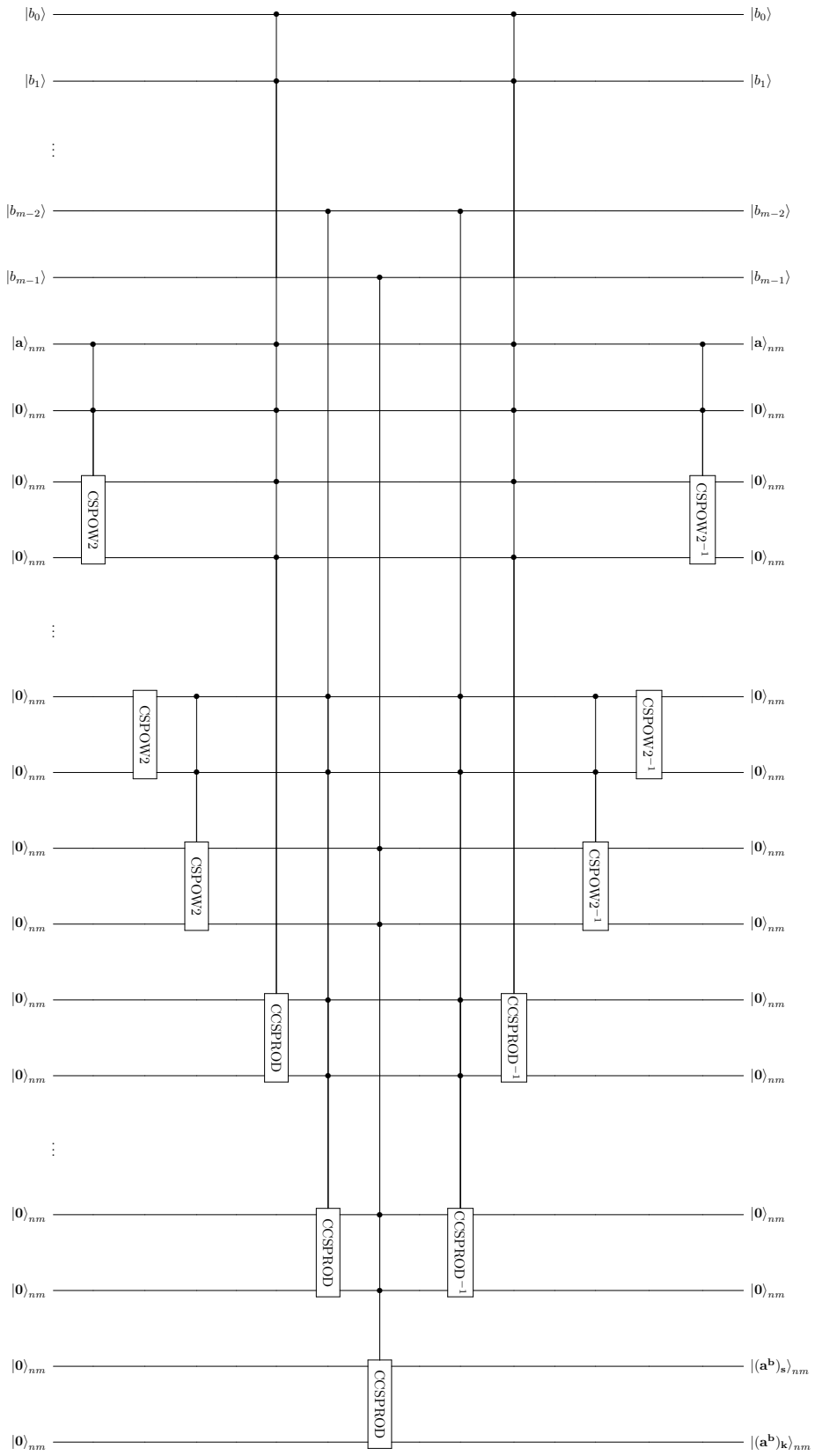
Figura 57: *Implementació de l'operador CSPROD per a calcular el producte de dos naturals de n qubits sobre dos registres addicionals en aritmètica Carry-Save.* **Font:** Elaboració pròpia mitjançant Q-Circuit.

Gràcies a aquests operadors podriem dissenyar versions Carry-Save dels dos circuits ja presentats dels blocs **POW** simplement substituint cada registre per dos que en continguin la seva representació equivalent en Carry-Save i cada operador per la seva versió dissenyada en aquesta secció.

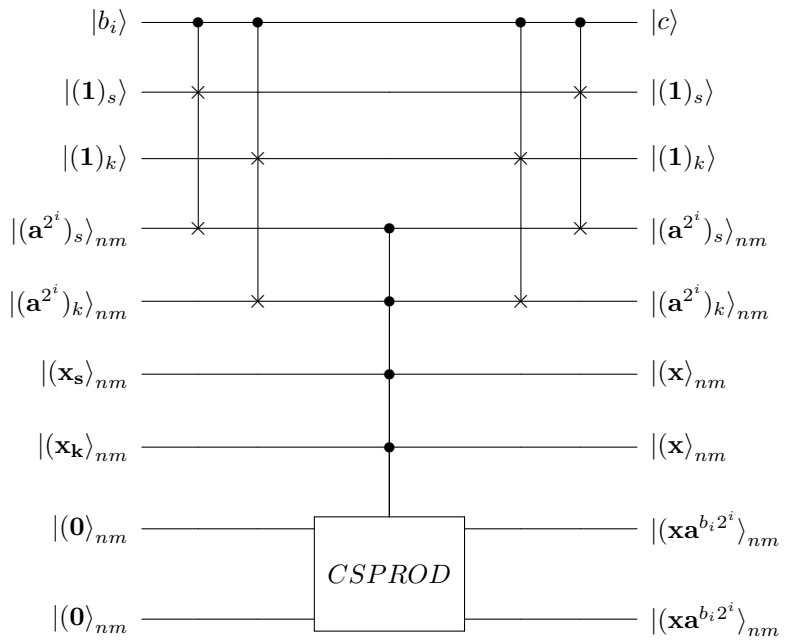
Com sempre, condicionar aquests operadors a l'estat d'un altre qubit es pot realitzar de dues maneres, una mitjançant un qubit temporal addicional per cada **Toffoli** en paral·lel i una altra augmentant la profunditat, però resulta més senzill calcular directament els termes $a^{b_i 2^i}$ substituint l'operand per un registre inicialitzat a $|\mathbf{1}\rangle_{nm}$ o $|\mathbf{0}\rangle_{nm}$ de manera condicionada mitjançant l'operador **CNULL**:



La versió amb codificació de Fibonacci ha estat dissenyada per a reduir el nombre de qubits temporals, de manera que no té gaire sentit desfer aquesta millora aplicant Carry-Save, de manera que tan sols donarem el circuit explícit derivat de la substitució mencionada:


$|b_0\rangle$
$|b_1\rangle$
$|b_{m-2}\rangle$
$|b_{m-1}\rangle$
$|\mathbf{a}\rangle_{nm}$
$|\mathbf{0}\rangle_{nm}$
CSPOW2
CSPOW2^{-1}
CCSPROD
CCSPROD^{-1}
$|(\mathbf{a^b})_\mathbf{s}\rangle_{nm}$
$|(\mathbf{a^b})_\mathbf{k}\rangle_{nm}$

On els blocs **CCSPROD** contenen un bloc **CNULL** per a cada operand condicionat al qubit de b corresponent:



Com que aquesta secció tracta de minimitzar el màxim possible la complexitat temporal dels circuits, realitzarem una segona modificació al circuit per a reduir la profunditat de la part d'aquest on es realitzen les multiplicacions condicionades.

Podem usar una idea similar a la usada en el disseny del multiplicador Carry-Save, on hem assolit profunditat logarítmica realitzant les sumes dels termes en forma d'àrbre. Com que el producte també és commutatiu podem definir el producte parcial $P[i,j]$ com:

$$P[i,j] = a^{b_i 2^i} \cdot a^{b_{i+1} 2^{i+1}} \dots a^{b_{j-1} 2^{j-1}} \cdot a^{b_j 2^j}$$

Amb cas base:

$$P[i,i] = a^{b_i 2^i}$$

I definició recursiva:

$$P[i,j] = P[i,k] \cdot P[k+1,j]$$

El qual ens permet calcular $P[0, m-1]$ a partir dels valors $P[i,i]$ en $O(log\ m)$ passos. El circuit complet que realitza l'exponenciació en aritmètica Carry-Save es mostra a la figura

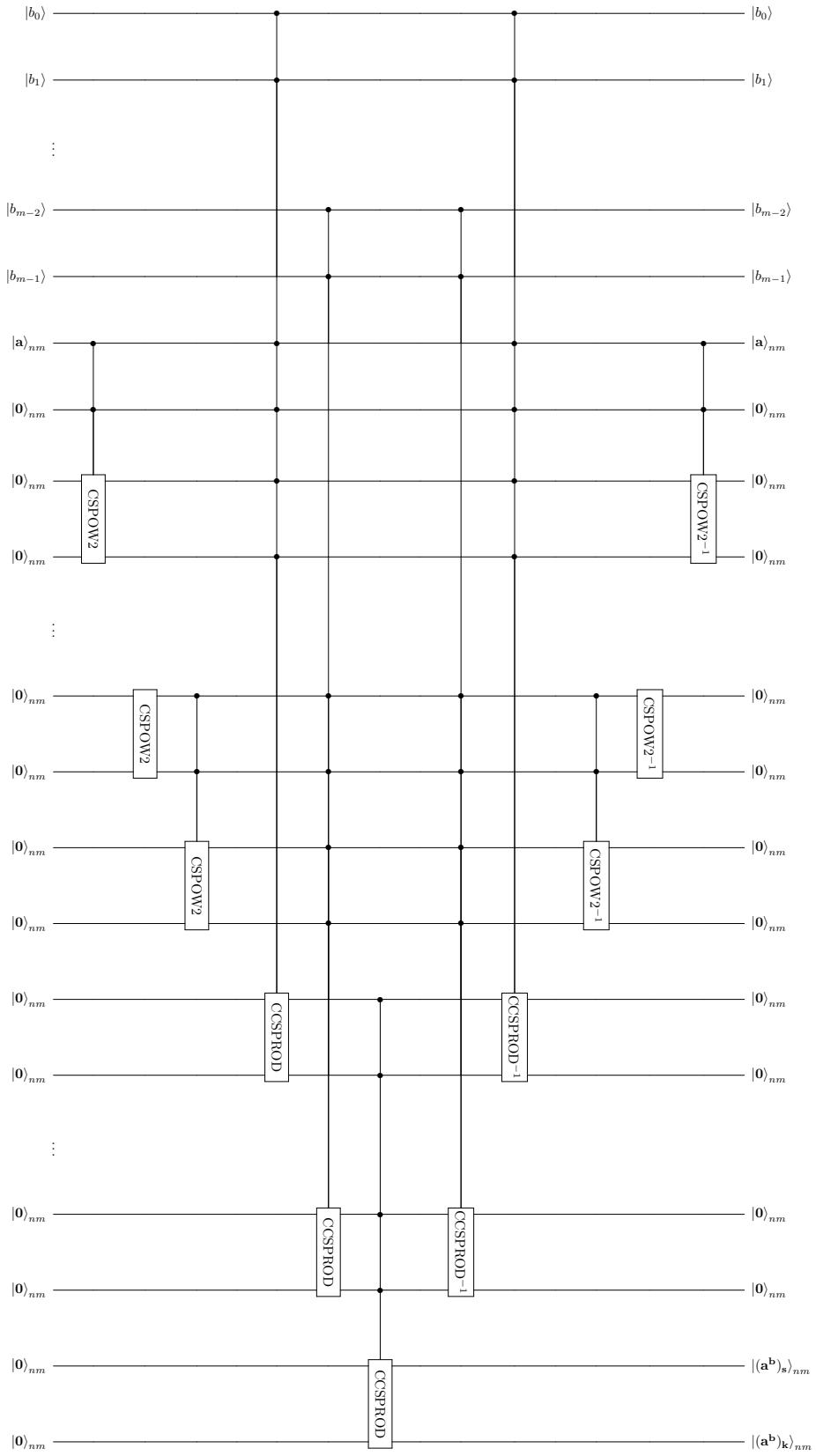


Figura 58: *Implementació Carry-Save de l'operador POW per a calcular l'exponenciació d'un registre de n qubits per un altre de m. L'arbre de càlcul dels productes no és estrictament correcte ja que no podem representar tots els passos a menys que s'assumeixi que* $m = 4$. **Font:** Elaboració pròpia mitjançant Q-Circuit.

Complexitat

En general, el circuit consisteix de $O(m)$ blocs **CSPOW2** i $O(m)$ blocs **CCSPROD**.

Cada bloc **CSPOW2** té una profunditat igual a la dels blocs **PROD** Carry-Save que el composen, de $O(log\ nm)$, per una profunditat total de $O(m\ log\ nm)$. Cada operador requereix de $O(nm)$ qubits per a condicionar l'operació i $O(n^2m^2)$ per als termes del producte del bloc **PROD**, els quals es poden reusar ja que les operacions són seqüencials.

Cada bloc **CCSPROD** té una profunditat també de $O(log\ nm)$ i requereix de $O(n^2m^2)$ qubits temporals, els quals no poden ser reutilitzats ja que aquesta segona part és paral·lela, per una complexitat espacial total de $O(n^2m^3)$ i temporal de $O(log(m)log(nm)) = O(log^2\ m + log\ m\ log\ n)$.

En total, el circuit usa $O(n^2m^3)$ qubits temporals i té una profunditat de $O(log^2\ m + log\ m\ log\ n)$.

4 Conclusions

Malgrat les limitacions del model de computació quàntica, hem estat capaços d'obtenir circuits que realitzin les operacions aritmètiques bàsiques sobre tots els estats en superposició presents en un sistema quàntic, de manera que podem assegurar que podem implementar qualsevol[17] algorisme o estructura de dades clàssica en un computador quàntic.

Tot i que la gran majoria d'estudis previs d'aquest camp han estat realitzats en relació als algorismes de Shor de factorització i càlcul del logaritme discret, hem estat exitosos en adaptar aquests dissenys o obtenir circuits completament nous per a realitzar operacions aritmètiques generals.

També hem estat capaços d'obtenir dissenys que optimitzin la complexitat espacial o temporal, fent la nostra recerca útil ja sigui per a implementar algorismes en els ordinadors quàntics bàsics actuals, els quals tenen una quantitat de qubits molt limitada, o per hipotètics processadors quàntics futurs, on la velocitat de computació és la mètrica a maximitzar.

A continuació es resumeixen els resultats i conclusions respectives de cada operació implementada.

4.1 Lògica booleana

Hem vist que podem calcular les 4 operacions **NOT**,**OR**,**AND**,**XOR** sobre un qubit addicional (out of place) inicialitzat a $|0\rangle$. A més podem també aplicar les portes **NOT** i **XOR** sobre els mateixos operands (in place) ja que són reversibles.

La taula 9 mostra el nombre de portes requerides per a implementar cadascuna d'aquestes operacions.

Operació	In place	X	CNOT	Toffoli
NOT	Si	1	0	0
	No	0	1	0
AND	No	0	0	1
OR	No	1 / 5 [18]	0	1
XOR	Si	0	1	0
	No	0	2	0

Taula 9: *Nombre de portes requerides per a implementar operacions de lògica booleana.* **Font:** Elaboració pròpia

També hem justificat que es pot implementar qualsevol operació booleana de la forma $f : \{0,1\}^n \to \{0,1\}^m$ sobre un registre addicional de m qubits descomponent-la en m funcions de la forma $f : \{0,1\}^n \to \{0,1\}$, cadascuna equivalent a una matriu de permutació de dimensió 2^n, el qual és un operador reversible i per tant implementable en un ordinador quàntic.

4.2 Suma de naturals i enters

Hem aconseguit adaptar l'algorisme bàsic de suma de naturals i enters en base 2 per a operar amb registres de qubits i n'hem proporcionat tres modificacions amb els seus avantatges i inconvenients.

Ripple-Carry sense qubits addicionals

Hem presentat una variant de l'implementació bàsica Ripple-Carry basada en la descomposició de la porta **MAJ** capaç de realitzar la suma en temps lineal sense usar cap qubit addicional. També s'han presentat modificacions per a realitzar aquesta operació de manera modular, sobre un registre addicional, o les dues variacions alhora.

[17] Sempre i quan es pugin establir fites espacials i temporals.

Aquest disseny és ideal quan la memòria és limitada, tal i com passa amb els ordinadors quàntics actuals dedicats a la recerca del camp. Tot i que hem donat modificacions per adaptar el circuit a la suma sobre un registre addicional, aquest disseny només es recomana quan no es disposa d'aquest altre registre, ja que fins i tot el circuit més bàsic és capaç de realitzar la suma amb aquest, amb menor cost de portes i profunditat.

La taula 10 mostra el nombre de portes requerides per a implementar cadascuna d'aquestes versions per a sumar registres de n qubits.

Modular	**In place**	**X**	**CNOT**	**Toffoli**
Si	Si	0	5n-2	2n-1
	No	0	6n-2	2n-1
No	Si	0	5n-1	2n
	No	0	6n-1	2n

Taula 10: *Nombre de portes requerides per a cadascuna de les versions del sumador Ripple-Carry sense qubits addicionals.* **Font:** Elaboració pròpia

Carry-lookahead amb profunditat logarítmica

Una altra variant del circuit de suma presentada ha estat la que usa la tècnica anomenada Carry-Lookahead per a obtenir una profunditat logarítmica a cost d'usar una quantitat de qubits temporals lineal en la mida dels registres.

Com s'ha fet abans, s'han donat les quatre versions d'aquest circuit que treballen sobre un operand o registre addicional i realitzen l'operació de manera modular o calculen també elròssec.

Aquest disseny és ideal quan la velocitat de computació de la suma és vital i es poden permetre els qubits temporals, i es recomana usar la versió que actua sobre un registre addicional quan sigui possible ja que l'alternativa té el doble de profunditat.

La taula 11 mostra el nombre de portes requerides per a implementar cadascuna d'aquestes versions per a sumar registres de n qubits.

Modular	**In place**	**X**	**CNOT**	**Toffoli**
Si	Si	$2n$	$4n-1$	$9n-6w(n)-6\lfloor log(n)\rfloor-2$
	No	0	$3n-1$	$5n-3w(n)-3\lfloor log(n)\rfloor-1$
No	Si	$2n+2$	$4n+3$	$9n-6w(n+1)-3\lfloor log(n+1)\rfloor+7$
	No	0	$3n+2$	$5n-3w(n+1)-3\lfloor log(n+1)\rfloor+4$

Taula 11: *Nombre de portes requerides per a cadascuna de les versions del sumador Carry-Lookahead per a sumar en temps logarítmic.* **Font:** Elaboració pròpia

Suma mitjançant la Transformada de Fourier Quàntica

La Transformada de Fourier Quàntica transforma un sistema a un estat que permet modificar el valor emmagatzemat mitjançant rotacions, el qual resulta especialment útil quan es volen realitzar sumes o restes sense usar qubits addicionals o fins i tot sense usar un segon operand.

Aquesta transformació no permet calcular el ròssec de la suma ja que aquest es perd en incrementar els exponents per sobre de 1 degut a que tan sols la part decimal d'aquests té un valor no trivial. De totes maneres no es recomana usar aquest mètode per a realitzar sumes sense qubits addicionals a no ser que en calgui realitzar moltes d'aquestes seguides, en qual cas es pot realitzar la transformació de profunditat quadràtica un sol cop i aplicar cada suma de profunditat lineal seqüencialment.

Aquesta operació consisteix de dues parts, la transformació i la suma en sí, i les dues són implementades únicament mitjançant rotacions condicionades i portes **H**. La taula12 mostra el nombre d'aquestes rotacions condicionades requerides per a implementar cadascuna de les parts del circuit.

	H	$\mathbf{R_k}$
QFT	n	$\frac{n(n+1)}{2}$
ADD	0	$\frac{n(n+1)}{2}$

Taula 12: *Nombre de portes requerides per a la Transformada de Fourier Quàntica i per al sumador en aquesta transformació.* **Font:** Elaboració pròpia

4.2.1 Comparació

Com s'ha mencionat anteriorment, si es volen estalviar qubits es recomana usar o bé el circuit bàsic Ripple-Carry o la versió sense registre addicional. Si en canvi es vol maximitzar el rendiment es recomana usar la versió Carry-Lookahead sempre i quan es pugui permetre la memòria addicional. Si es vol sumar un valor constant sense usar qubits addicionals, cal usar la Transformada de Fourier ja que qualsevol altra implementació calcula els ròssecs amb portes **ADD**, les quals no són reversibles a no ser que es calculin sobre un qubit addicional.

La taula 13 mostra una comparativa entre les complexitats espacials i temporals de les tres modificacions.

Implementació	**Profunditat**	**Qubits addicionals**
Ripple-Carry	$O(n)$	0
Carry-Lookeahed	$O(log\ n)$	$O(n)$
QFT	$O(n)$	0
QFT ADD	$O(n)$	0

Taula 13: *Complexitats espacials i temporals dels blocs ADD.* **Font:** Elaboració pròpia

4.3 Negació d'enters

La negació d'enters en Complement a 2 es pot calcular mitjançant una negació dels bits individuals seguida de la suma d'una unitat. La primera operació és trivial, i hem proporcionat diverses opcions per a realitzar la suma de la unitat.

Si es vol minimitzar el nombre de qubits addicionals es recomana usar la suma d'una unitat basada en la Transformada de Fourier Quàntica, ja que tan sols cal aplicar un nombre constant de rotacions per a realitzar aquesta operació. Mètodes com Ripple-Carry o Carry-Lookahead tenen diferents requeriments de qubits i complexitats temporals, però en general si es vol minimitzar el temps de càlcul cal usar, com sempre, el sumador Carry-Lookahead.

La taula 14 mostra una comparativa entre les complexitats espacials i temporals de les tres modificacions.

Suma d'una unitat	**Profunditat**	**Qubits addicionals**
Ripple-Carry	$O(n)$	$O(n)$
Carry-Lookeahed	$O(log\ n)$	$O(n)$
QFT	$O(n)$	0

Taula 14: *Complexitats espacials i temporals dels blocs NEG.* **Font:** Elaboració pròpia

Cal destacar que tot i que les complexitats espacials de les implementacions Ripple-Carry i Carry-Lookahead siguin iguals, a la pràctica la primera tan sols requereix n qubits pels ròssecs, mentre que la segona en requereix com a mínim el doble, el qual augmenta encara més en funció de $w(n)$.

4.4 Resta de naturals i enters

La reversibilitat dels operadors quàntics sol ser un obstacle en el càlcul en aquest model, però en el cas de la resta hem pogut aprofitar aquesta propietat per a obtenir-ne circuits capaços de realitzar-la aprofitant circuits de suma ja existents. Un mètode alternatiu per a nombres enters que hem presentat es basa en negar un operand mitjançant els blocs de negació estudiats.

En general, les complexitats espacials i temporals són equivalents a les dels circuits de suma que els composen, les quals es mostren a la taula 15.

Implementació	**Profunditat**	**Qubits addicionals**
Ripple-Carry	$O(n)$	0
Carry-Lookeahed	$O(log\ n)$	$O(n)$
QFT	$O(n)$	0
QFT SUB	$O(n)$	0

Taula 15: *Complexitats espacials i temporals dels blocs SUB.* **Font:** Elaboració pròpia

4.5 Producte de naturals i enters

Mitjançant la descomposició d'un dels operands en la seva representació en base 2 podem escriure el producte com la suma repetida del segon operand condicionada a cada bit d'aquesta. Hem mostrat un esquema bàsic per a obtenir un circuit capaç de calcular el producte de dos registres basat en sumes condicionades i hem donat dues opcions per a condicionar qualsevol circuit basat en portes del nostre conjunt a un qubit addicional mitjançant l'extensió de les **Toffoli**, i hem mostrat un mètode alternatiu basat en anul·lar un dels operands substituint-lo temporalment per un registre inicialitzat a $|\mathbf{0}\rangle_n$.

El cost addicional de condicionar un operador **ADD** amb cadascun dels tres mètodes es mostra a la taula 16, mesurat com l'augment de profunditat aproximada de cada **Toffoli-k**[19], l'augment de la profunditat general del circuit i el nombre de qubits temporals que calen.

Implementació	**Profunditat Toffoli-k**	**Profunditat total**	**Qubits**
Extensió sense qubits	$O(k^2)$	0	0
Extensió amb qubits	$O(1)$	0	1
Anul·lació d'un operand lineal	0	$O(n)$	$O(n)$
Anul·lació d'un operand logarítmica	0	$O(log\ n)$	$O(n)$

Taula 16: *Complexitat del condicionament dels blocs ADD.* **Font:** Elaboració pròpia

Amb aquests mètodes de condicionat de suma, hem mostrat un esquema de circuit de multiplicació de naturals de n qubits basat en incrementar el registre resultat amb tots els termes del producte. Aquest circuit usa n blocs **CADD** seqüencials, de manera que la seva complexitat temporal serà igual a n vegades la complexitat del bloc de suma condicionada.

També hem introduït l'aritmètica Carry-Save per a reduir n sumes a 2 en $O(log\ n)$ passos, el qual pot ser usat per a sumar els n termes del producte en temps logarítmic.

[19] No cal modificar totes les **Toffoli** i **CNOT**, però serveix com a una bona mesura de l'efecte d'aquest mètode de condicionat dels operadors.

Aquests circuits tan sols actuen sobre un registre addicional ja que cal informació dels dos operands per a calcular cada terme. Gràcies al circuit de divisió, però, podem obtenir un disseny per a multiplicar sobre un dels operands mitjançant la inversió d'aquest amb un residu de 0.

La taula 17 mostra el nombre de qubits temporals i la profunditat de cadascun dels dissenys, assumint condicionat sense qubits addicionals.

Implementació		**Profunditat**	**Qubits addicionals**
Producte bàsic	Ripple-Carry	$O(n^2)$	0
	Carry-Lookahead	$O(n\ log\ n)$	$O(n)$
Carry-Save		$O(log\ n)$	$O(n^2)$
Divisió inversa	Ripple-Carry	$O(n^2)$	$O(n)$
	Carry-Lookahead	$O(n\ log\ n)$	$O(n)$

Taula 17: *Complexitats temporals i espacials de les implementacions de l'operador PROD per a calcular el producte de dos naturals de n qubits.* **Font:** Elaboració pròpia

També s'ha proporcionat un esquema simple per a realitzar aquesta operació sobre enters mitjançant la negació dels operands i del resultat amb els circuits ja estudiats.

4.6 Divisió de naturals i enters

Primerament hem mostrat un circuit capaç de descompondre un natural $\mathbf{x} = \mathbf{a}c + \mathbf{b}$ en $\mathbf{b}$ i c mitjançant una resta seguida d'una suma condicionada al signe del resultat. Aquest esquema permet usar qualsevol dels sumadors estudiats anteriorment per a adaptar el bloc a les necessitats particulars del problema.

Hem presentat un circuit que usa aquest bloc per a implementar l'algorisme clàssic de divisió basat en restes condicionades, les quals permeten calcular cadascun dels bits del quocient, deixant el residu en el registre original del dividend.

La taula 18 mostra el nombre de qubits temporals i la profunditat quan s'usen sumadors Ripple-Carry o Carry-Lookeahead, assumint de nou condicionat sense qubits addicionals.

Implementació	**Profunditat**	**Qubits addicionals**
Ripple-Carry	$O(n^2)$	0
Carry-Lookahead	$O(n\ log\ n)$	$O(n)$

Taula 18: *Complexitats temporals i espacials de les implementacions de l'operador DIV per a calcular el quocient i el residu de la divisió de dos naturals de n qubits.***Font:** Elaboració pròpia

4.7 Exponenciació de naturals

Com hem fet amb la multiplicació, podem descompondre l'exponenciació com el producte d'una sèrie de termes condicionats als bits de l'exponent, el qual ens ha permès dissenyar un esquema bàsic d'exponenciació basat en blocs de producte condicionat o blocs normals de producte i blocs d'anul·lació d'operands condicionals.

Aquest esquema bàsic requereix calcular els m termes del producte en paral·lel ja que és necessari per a desfer la computació posteriorment. És per això que hem presentat una manera alternativa de representar naturals mitjançant la sèrie de Fibonacci i la codificació de Zeckendorf, la qual els permet descompondre com a la suma de termes d'aquesta sèrie. Gràcies a la definició recursiva de la sèrie de Fibonacci, podem calcular tots els termes del producte amb tan sols dos registres, oferint una reducció exponencial de la memòria requerida amb un petit augment de la profunditat causat per la codificació de l'exponenent.

Finalment hem presentat la versió més eficient, basada en els multiplicadors Carry-Save de profunditat logarítmica. Aquest últim disseny opera amb les representacions Carry-Save de tots els nombres, duplicant el nombre de qubits temporals requerits, però permetent accelerar l'operació substancialment respecte els dissenys anteriors.

La taula 19 mostra el nombre de qubits temporals i la profunditat quan s'usen sumadors Ripple-Carry o Carry-Lookeahead, assumint de nou condicionat sense qubits addicionals.

Implementació		**Profunditat**	**Qubits addicionals**
Circuit bàsic	Ripple-Carry	$O(n^2m^3)$	$O(nm^2)$
	Carry-Lookahead	$O(nm^2\ log\ nm)$	$O(nm^2)$
Codificació de Fibonacci	Ripple-Carry i QFT	$O(n^2m^2)$	$O(nm)$
	Carry-Lookahead	$O(m\ log\ m + nm^2\ log\ nm)$	$O(nm)$
Carry-Save		$O(log^2\ m + log\ m\ log\ n)$	$O(n^2m^3)$

Taula 19: *Complexitats temporals i espacials de les implementacions de l'operador POW.* **Font:** Elaboració pròpia

5 Gestió del projecte

Tot i que és difícil acotar amb precisió els resultats esperats en un treball de recerca teòrica, es pot construïr una jerarquia d'objectius progressius que permetin orientar el desenvolpament del projecte.

5.1 Objectius

Recerca de conceptes bàsics: Gran part dels conceptes requerits per un exitós desenvolupament del treball no estan inclosos en el pla d'estudis del grau. Conseqüentment, cal realitzar un estudi intensiu de certs conceptes bàsics d'àlgebra lineal i de computació quàntica per a ser capaç d'obtenir el coneixement necessari per a poder comprendre, i construir coneixement, sobre els documents científics a recercar.

Aquests conceptes bàsics es troben llistats a continuació:

- **Àlgebra lineal per computació quàntica:** La naturalesa matricial dels estats quàntics i dels operadors justifica la necessitat de tenir uns coneixements bàsics d'àlgebra lineal, però també permet definir operadors quàntics a partir dels **productes intern i extern** sense haver d'especificar la matriu, o bé usar la **descomposició en valors propis** per a trobar aquesta representació d'un operador arbitrari. També cal conèixer el concepte de **matriu unitària**, la seva estructura, propietats i efectes en combinació amb els vectors normalitzats que representen estats.
- **Aritmètica en computadors clàssics:** Abans d'investigar sobre la implementació d'aritmètica en computadors quàntics, és vital entendre com s'efectuen aquests càlculs en computadors clàssics.
- **Eines bàsiques de computació quàntica:** Cal investigar i entendre el funcionament d'eines vitals en el desenvolupament d'algorismes quàntics com la **transformada de Fourier quàntica**, l'**operador de densitat** per a representar estats quàntics probabilísticament, la **descomposició de Schmidt** i la **purificació**.

Implementació progressiva de càlculs aritmètics: Seguint el procediment jerarquitzat usat en el disseny de processadors clàssics, idealment es començaria desenvolupant o recercant un circuit quàntic capaç de realitzar la **suma de tres qubits en base 2**. Si això fos possible, es podrien dissenyar circuits per a realitzar la **suma i resta de naturals en base 2 o enters en complement a 2** d'un nombre arbitrari de qubits, o en cas contrari caldria investigar possibles alternatives per a implementar aquesta operació. A partir d'aquestes portes de suma i resta es poden dissenyar circuits de **multiplicació i divisió** de naturals o enters, seguit per un que permeti realitzar **exponenciació**.

5.1.1 Objectius secundaris

Depenent dels resultats anteriors, serà possible o necessari haver de resoldre algun dels següents objectius:

Operacions condicionades Implementar les operacions aritmètiques condicionades a un qubit de control eficientment.

Estructures de dades Investigar la possibilitat d'implementar estructures de dades més complexes (grafs, heaps, etc.) de tamany constant en qubits i la possibilitat d'usar el paral·lelisme quàntic amb aquestes.

Operacions amb decimals Estudiar la possibilitat de representar nombres amb coma fixa o flotant i de realitzar operacions amb aquests.

Blocs iteratius Investigar si és possible implementar algorismes amb un nombre d'iteracions variable sobre un conjunt d'estats en superposició.

5.2 Requisits

Tot i ser un treball principalment d'investigació, el qual requereix de recerca i síntesi extensiva de multitud de documents científics, es poden establir una sèrie de requisits mínims pel que fa als dissenys esperats, sobre els quals es poden construir optimitzacions. Aquests requisits mínims són les versions clàssiques de tots els algorismes que computen operacions aritmètiques sobre naturals implementades mitjançant operadors quàntics. De manera que, com a mínim, el resultat del projecte hauria de ser un conjunt de circuits quàntics capaç de ser usat per a dissenyar qualsevol còmput clàssic sobre tots els estats en superposició.

5.3 Obstacles i riscos

Donada la impredictibilitat dels resultats o el succés del projecte, és convenient analitzar els possibles riscos o obstacles que puguin dificultar el seu desenvolupament i estudiar les seves solucions.

5.3.1 Impossibilitat de la implementació

És possible que implementar algun dels requisits sigui matemàticament impossible o poc factible degut a la necessària reversibilitat de les portes quàntiques o que el fenomen de l'entrellaçament impedeixi realitzar certs còmputs sobre qubits independents. S'ha realitzat una extensa recerca prèvia a la sel·lecció del tema del projecte, de manera que els requeriments mínims són possibles. Si els requeriments o objectius més avançats no ho fossin, tan sols caldria dedicar més temps a la recerca sobre investigacions prèvies dels requeriments bàsics i a la seva anàlisi i optimització.

5.3.2 Manca de recerca existent

En el cas que la recerca existent no sigui suficient per a realitzar un treball investigatiu de suficient qualitat, la responsabilitat de dissenyar i optimitzar els algorismes recaurà sobre el desenvolupador, de manera que el projecte tindrà una vessant pràctica més important.

5.3.3 Problemes temporals

Tot treball o projecte és subjecte a limitacions temporals, l'efecte de les quals tan sols es pot minimitzar emprant una correcta metodologia i elaborant una planificació de tasques que tingui en compte possibles imprevistos. Donada la naturalesa progressiva d'aquest projecte, es pot ajustar l'abast d'aquest dinàmicament si la planificació inicial no s'ajustés a la dificultat del procés d'investigació o si sorgís algun imprevist fora de l'anàlisi anterior.

5.3.4 Problemes tecnològics

Pot ser que un error tecnològic obstaculitzi el desenvolupament del treball o directament n'elimini una porció d'aquest. Com que tota la informació del projecte es troba en el document a redactar o en els prototips de codi que testejen els dissenys, tan sols cal garantir que tots aquests fitxers sensibles es troben dins un repositori git, amb copies en servidors externs per a facilitar la recuperació de dades en cas que sigui necessari.

5.4 Metodologia i rigor

En qualsevol projecte de caire científic cal adherir-se a una metodologia concreta des de l'inici i fer ús de les eines pertinents per a optimitzar el temps invertit i minimitzar l'efecte dels imprevistos o possibles problemes detallats anteriorment. A continuació es desenvolupen aquests punts amb més detall:

5.4.1 Metodologia

Donada la naturalesa purament investigadora del projecte, la metodologia es basa en l'assoliment de fites tant teòriques com pràctiques dissenyades per a assegurar una bona distribució temporal de la feina. Com que és impossible conèixer els resultats abans de l'inici de la investigació, aquesta distribució temporal té en compte la possibilitat d'haver de dissenyar els circuits manualment a causa de la falta de recerca anterior, i contempla casos on la impossibilitat matemàtica d'algun dels objectius justifica la inversió de temps en la prova formal d'aquest fet, o on simplement un error tecnològic imprevisible impedeixi el correcte desenvolupament del projecte.

Tot i que no és ideal, la dificultat teòrica del projecte justifica mantenir oberta la possibilitat d'actualitzar dinàmicament aquestes fites per a garantir que l'integritat i correctesa del treball no són sacrificades a causa d'una incorrecta avaluació inicial de la complexitat de l'estudi. Si es realitzés qualsevol canvi, aquest seria consultat i consensuat amb el director del projecte.

5.4.2 Eines de seguiment

Com que el treball no requereix de desenvolupament de codi (no com a finalitat, tot i que s'usarà per a l' investigació), no cal usar software de control de versions com *Git* per a aquest, tot i que serà emprat igualment per a mantenir còpies dels documents en les seves diferents versions, facilitant així el seguiment del progrés en mostrar les dates d'assoliment de cadascuna de les fites.

Ja que la part inicial del projecte tan sols requereix de recerca i investigació de documents científics, es pot realitzar un seguiment d'aquest procés mitjançant el mètode de *Bullet Journaling* combinat amb les fites temporals globals del projecte per a assegurar que no es dedica temps insuficient a la comparació, summarització i optimització del material estudiat.

Òbviament, es duran a terme reunions periòdiques amb el director del projecte per a discutir els avanços en la investigació, l'estat actual del projecte i les fites, i resoldre dubtes o aclarir conceptes teòrics.

5.5 Descripció de les tasques

Aquest projecte s'ha de realitzar en el marc temporal comprés entre principis de Febrer i finals de Juny del 2022, de manera que és imperatiu definir concretament el conjunt de tasques a realitzar i determinar amb alta precisió els intervals de temps a dedicar per a l'assoliment de cadascuna.

El principal obstacle a l'hora de realitzar aquesta distribució de tasques i posterior estimació de temps és que aquest és un treball majoritàriament de recerca, de manera que no existeix un patró a priori per a guiar la recerca, tal i com passaria si el projecte consistís en desenvolupar software. També cal destacar que, encara que sabéssim prèviament el conjunt de documents a investigar i experiments a realitzar, no hi ha manera de predir amb alta fiabilitat el temps que es requerirà per a obtenir una correcta comprensió dels documents o a realitzar aquests experiments. Cal, per tant, destacar que les tasques (i el temps esperat de cadascuna) en les quals s'ha dividit el projecte són una estimació a priori basada en la experiència prèvia del tutor i del desenvolupador sobre el procés d'investigació, experimentació i redacció de documents científics.

Per a poder determinar les tasques i aproximar els seus temps de desenvolupament, s'ha assumit una dedicació setmanal per part del desenvolupador d'unes 25 hores, tot i que es permet que aquest valor fluctuï entre 15 i 35 en funció de l'estat actual del projecte. Aquesta flexibilitat és necessària a causa de l'alta desviació dels temps esperats per a realitzar les tasques, i permet completar tots els objectius secundaris encara que el desenvolupament normal del treball sigui afectat per problemes imprevistos o alta dificultat teòrica durant la investigació.

A continuació es descriuen les diferents tasques a realitzar durant el projecte, les quals han estat determinades per a representar un *desenvolupament ideal* del projecte, però és possible que les dependències entre aquestes canviïn ja que és impossible conèixer els resultats de la investigació abans de realitzar-la. S'ha omès la descripció de materials o recursos necessitats per a la realització de cada tasca degut al fet que totes són equivalents en aquest àmbit, requerint tan sols d'un ordinador amb accés a internet.

5.5.1 Estudi de conceptes bàsics [*CB*]

La primera tasca del projecte consisteix en complementar els coneixements adquirits durant el grau mitjançant la lectura del llibre *Quantum Computation and Quantum Information* [19], el qual summaritza amb gran detall molts dels conceptes avançats de la computació quàntica, entre ells la **transformada de Fourier quàntica**, l'**operador de densitat**, la **descomposició de Schmidt** i la **purificació**, entre d'altres.

Aquesta tasca inclou el llistat i recerca d'altres documents relacionats que desenvolupin aquests temes més profundament i tinguin potencial de ser usats en les tasques futures. Un exemple seria la recol·lecció de documents que investiguin possibles implementacions de la transformada de Fourier quàntica, permetent-nos comparar les seves complexitats temporals i espacials a l'hora d'analitzar aquells algorismes que l'aprofiten per a realitzar computació clàssica.

Previsió temporal 5 hores per a la lectura de les introduccions a totes les seccions i sel·lecció posterior d'aquelles a mirar en profunditat i 35 hores per a l'estudi d'aquestes i documents derivats.

Dependències Aquesta és una tasca d'estudi inicial que té com a finalitat obtenir les eines bàsiques per a enfocar la resta del projecte. Conseqüentment, la seva realització no depèn d'altres tasques.

5.5.2 Llògica booleana sobre qubits [*LB*]

La primera tasca d'investigació consisteix en dissenyar circuits quàntics capaços d'implementar portes clàssiques arbitràries de fins a 2 bits. Per a realitzar-la cal primerament cercar documents científics amb investigacions prèvies d'aquestes portes, comparar els seus resultats i, en el cas que sigui possible, obtenir un conjunt de dissenys capaços d'efectuar aquests càlculs, cadascun amb els seus avantatges i inconvenients.

Cal destacar que, si fos possible obtenir una implementació eficient d'aquestes portes, facilitaria substancialment l'assoliment de les tasques posteriors, ja que tota porta clàssica de qualsevol nombre de bits es pot implementar amb portes que operen sobre 2 o menys d'aquests.

Previsió temporal S'espera que l'estudi o possible disseny propi d'aquest conjunt de circuits portes requereixi no més de 15 hores.

Dependències En consistir en la implementació més bàsica de circuits, aquesta tasca només depèn de la tasca inicial d'estudi bàsic: **5.5.1** [*CB*].

5.5.3 Aritmètica bàsica de naturals [*AN*]

Aquesta tasca consisteix en el disseny d'un circuit capaç de realitzar la suma i resta de dos naturals codificats en base 2 emmagatzemats en dos registres d'un nombre arbitrari de qubits. De la mateixa manera que en la tasca anterior, cal realitzar una recerca extensiva d'estudis i dissenys previs i escollir-ne un o diversos i analitzar el seu funcionament i complexitat.

Cal destacar que si la tasca **5.5.2** [*LB*] és exitosa, es pot obtenir un disseny per a sumar naturals trivialment.

Previsió temporal S'estima que la implementació de la suma en base 2 bàsica requereixi no més de 10 hores i la investigació i anàlisi de models alternatius o optimitzacions en requereixi unes 30.

Dependències Com que el procediment per a realitzar aquesta tasca varia en funció de si es poden implementar portes lògiques en qubits, aquesta depèn de la tasca **5.5.2** [*LB*] .També calen els coneixements sobre la transformada de Fourier i altres algorismes obtinguts en la tasca **5.5.1** [*CB*] .

5.5.4 Aritmètica bàsica d'enters [*AI*]

A continuació cal adaptar els dissenys de la tasca anterior per a operar amb nombres enters en comptes de naturals, de manera que cal implementar noves versions de la suma i resta, a part del canvi de signe de registres de nombre arbitrari de qubits. És possible que existeixin optimitzacions a la versió trivial obtinguda modificant els circuits de naturals, de manera que també caldrà investigar possibles documents o dissenys publicats prèviament.

Previsió temporal S'estimen 10 hores per a la modificació dels dissenys de naturals i 10 més a la recerca de dissenys alternatius publicats per a enters.

Dependències Aquesta tasca requereix els dissenys que realitzen suma i resta de naturals, i per tant depèn de la tasca **5.5.3** [*AN*] .

5.5.5 Aritmètica bàsica condicionada [*AC*]

Per a poder implementar algorismes més avançats cal tenir l'habilitat d'executar certes operacions tan sols en el subconjunt dels estats en superposició que compleixin una certa condició. Aquesta tasca consisteix en investigar i dissenyar circuits quàntics capaços d'executar sumes i restes de naturals i enters condicionades a l'estat d'un qubit.

Previsió temporal S'espera haver de dedicar un total de 20 hores a investigar i dissenyar modificacions condicionades a un o més qubits de control dels circuits anteriors.

Dependències Cal tenir circuits capaços de realitzar operacions lògiques, suma i resta de naturals i enters abans de dissenyar les seves versions condicionades. Per tant les dependències són les tasques **5.5.2** [*LB*] , **5.5.3** [*AN*] , **5.5.4** [*AI*] .

5.5.6 Producte i divisió per potències de 2 [*P2*]

Abans de dissenyar algorismes de divisió i multiplicació generals, és útil obtenir circuits per a realitzar aquestes operacions amb potències de dos, ja que poden ser implementades amb shiftat de bits, tot i que no és trivial ja que les operacions han de ser reversibles, i el shiftat no ho és.

Previsió temporal Tot i l'aparent simplicitat, obtenir circuits reversibles és una tasca que requerirà d'extensiva recerca, de manera que se li assigna una estimació conservadora de 20 hores.

Dependències Aquesta tasca tan sols requereix la base teòrica de **5.5.1** [*CB*] .

5.5.7 Producte de naturals [*PN*]

Usant les operacions de shiftat i sumes condicionades es pot implementar l'algorisme de multiplicació de naturals. Cal investigar implementacions alternatives i comparar-ne les seves complexitats temporals i el nombre de qubits addicionals que requereixen.

Previsió temporal S'espera que la implementació bàsica requereixi 10 hores, però se'n reserven 20 per a l'estudi de dissenys alternatius i recerca de possibles optimitzacions de l'algorisme bàsic en el model quàntic.

Dependències L'algorisme bàsic depèn del shiftat de bits (**5.5.6** [*P2*]) i d'operacions aritmètiques sobre naturals condicionades (**5.5.5** [*AC*]). Les optimitzacions requereixen els coneixements bàsics de la tasca **5.5.1** [*CB*].

5.5.8 Producte d'enters [*PI*]

Aquesta tasca tracta d'adaptar els circuits de producte de naturals per a operar amb enters amb signe, i investigar si existeixen alternatives més eficients a la trivial.

Previsió temporal S'estimen unes 10 hores per a modificar l'algorisme de multiplicació de naturals i 10 més per a la recerca de dissenys o optimitzacions alternatives.

Dependències Depèn dels circuits de producte de naturals (**5.5.7** [*PN*]) i de conceptes avançats de computació quàntica per a possibles optimitzacions(**5.5.1** [*CB*]).

5.5.9 Divisió de naturals [*DN*]

Seguidament cal dissenyar un circuit que obté el quocient i el residu de la divisió de dos naturals, el qual es pot obtenir a partir de les portes quàntiques de shiftat i aritmètica condicional, però cal també investigar les implementacions publicades prèviament i analitzar els seus funcionaments i complexitats.

Previsió temporal S'espera un total de 10 hores per a implementar la versió bàsica mitjançant restes condicionades, i 20 més per a la recerca d'alternatives més eficients.

Dependències Igual que en el producte de naturals, l'algorisme bàsic depèn del shiftat de bits (**5.5.6** [*P2*]) i d'operacions aritmètiques sobre naturals condicionades (**5.5.5** [*AC*]) i les optimitzacions i circuits alternatius requereixen els coneixements bàsics de la tasca **5.5.1** [*CB*].

5.5.10 Divisió d'enters [*DE*]

Tal i com s'ha fet amb el producte d'enters, cal modificar el circuit de divisió de naturals per a poder operar amb enters, i investigar l'existència de dissenys publicats més eficients que la versió bàsica.

Previsió temporal S'estimen unes 10 hores per a convertir tots els dissenys de naturals a enters, i 10 més per a investigar-ne alternatives publicades.

Dependències Depèn dels circuits de divisió de naturals (**5.5.9** [*DN*]) i de conceptes avançats de computació quàntica per a possibles optimitzacions respecte el circuit bàsic (**5.5.1** [*CB*]).

5.5.11 Exponenciació de naturals [*EN*]

Cal també obtenir el disseny d'un circuit quàntic capaç de computar el resultat d'elevar un natural a un altre, possiblement modular, o investigar l'existència de publicacions amb implementacions prèvies.

Previsió temporal S'estimen unes 10 hores per a implementar la versió bàsica usant exponenciació ràpida, i unes 20 per a investigar l'existència de dissenys publicats i optimitzacions.

Dependències Depèn dels circuits de producte de naturals (**5.5.7** [*PN*]) i de conceptes avançats de computació quàntica per a possibles optimitzacions respecte el circuit bàsic (**5.5.1** [*CB*]).

5.5.12 Blocs iteratius [*BI*]

Tot i que es pot iterar una certa porta quàntica simplement aplicant-la un nombre fixat de vegades, cal investigar la possibilitat d'implementar un bucle amb una condició de finalització que depèn de l'estat actual del sistema.

Previsió temporal S'estimen unes 20 hores per a investigar la possibilitat d'implementar circuits iteratius no fitats.

Dependències Depèn dels blocs condicionals (**5.5.5** [*AC*]).

5.5.13 Aritmètica de reals [*AR*]

Tasques de simulació de sistemes quàntics i algorismes avançats requereixen representar decimals per a operar, de manera que cal estudiar la possibilitat d'implementar operacions aritmètiques sobre algun estàndard de coma flotant tenint en compte que els circuits han de ser reversibles tot i la precisió finita i la pèrdua d'informació de la representació.

Previsió temporal S'assumeix que l'estudi i implementació de l'aritmètica en coma flotant(o fixa) sobre qubits requereixi unes 40 hores.

Dependències Depèn de la base teòrica (**5.5.1** [*CB*]), totes les operacions aritmètiques (**5.5.4** [*AI*], **5.5.8** [*PI*], **5.5.14** [*DI*]) i lògiques disponibles (**5.5.2** [*LB*]). Estàndards de coma flotant també requereixen blocs condicionals (**5.5.5** [*AC*]) i iteratius (**5.5.12** [*IT*]).

5.5.14 Estructures de dades [*ED*]

Aquesta tasca correspon al conjunt d'objectius secundaris que tracten d'investigar la possibilitat d'implementar estructures de dades avançades com grafs o *heaps* de tamany finit en qubits, i analitzar la capacitat dels ordinadors quàntics de realitzar operacions sobre aquestes estructures de dades per a veure si aquest model de computació pot ser usat per a representar informació clàssica més enllà de nombres.

Previsió temporal Donada la dificultat de la tasca i la manca de literatura existent, s'estima que caldran no menys de 40 hores per a determinar, a un nivell bàsic, les possibilitats i limitacions d'usar qubits per a representar i operar amb informació complexa.

Dependències Depèn de la base teòrica (**5.5.1** [*CB*]), totes les operacions aritmètiques (**5.5.4** [*AI*], **5.5.8** [*PI*], **5.5.14** [*DI*]) i lògiques disponibles (**5.5.2** [*LB*]). Algorismes avançats també requereixen blocs condicionals (**5.5.5** [*AC*]) i iteratius (**5.5.12** [*IT*]).

5.5.15 Seguiment

El seguiment de l'estat del projecte és vital per a un treball investigatiu d'aquesta magnitud, de manera que cal establir sessions de seguiment tant individuals com tutoritzades pel director.

Previsió temporal S'assigna 1 hora setmanal de seguiment individual i tutoritzat, ja sigui mitjançant correu electrònic o amb visites personals. Les reunions amb el director serien idealment setmanals però es permet realitzar-les cada dues setmanes si els horaris dels implicats ho requereixen. No considerem viable designar tasques explícites de seguiment amb dependències fixes donada l'alta variabilitat dels temps i relacions entre aquestes tasques. El seguiment serà, doncs, una tasca paral·lela a les d'investigació i documentació, amb període variable però mai superior a dues setmanes, per a garantitzar una efectiva comunicació i facilitar la supervisió i seguiment per part del director.

5.5.16 Documentació

En ser un treball majoritàriament teòric, és imperatiu invertir una quantitat superior de dedicació a l'elaboració del document final. Aquest document serà redactat paral·lelament amb la investigació, de manera que s'afegeix una subtasca pertinent en totes les anteriors per a garantitzar un correcte desenvolupament de l'informe. La nomenclatura d'aquestes subtasques ve donada pel sufix ***d***. Per exemple, la subtasca de documentació de [*CB*] és [*CBd*].

Previsió temporal S'assigna un total de 80 hores a l'elaboració del document a causa de la forta càrrega teòrica del projecte, les quals es dediquen paral·lelament a la recerca. També es reserva un bloc final de 50 hores que té com a objectiu finalitzar el document i preparar la defensa del treball.

Dependències Cada subtasca de documentació depèn de la resta de subtasques de la tasca en qüestió. La tasca de documentació final depén de totes le subtasques de documentació [***d***].

5.6 Taula resum

A continuació es mostra la taula 20, on es pot veure el conjunt de tasques, subtasques i les seves dependències corresponents de manera summaritzada. Cal destacar que mencions a tasques en negreta sense el sufixe en les dependències finals fan referència a totes les subtasques a excepció de les de documentació per a facilitar la lectura.

De manera que, amb aquesta assignació de tasques, obtenim un total d'hores esperat de 515, lleugerament superior a les 450 esperades pels 15 crèdits ECTS del projecte, cosa que es considera justificada donada l'alta càrrega teòrica del treball. Assumint una dedicació constant durant les 14 setmanes del projecte, caldria dedicar unes 35 hores setmanals, el qual correspon amb el límit superior que hem determinat adient, tot i que pot ser menor si els temps assignats a les tasques eren superiors als reals.

Tasca	Nom	Hores	Dependències
CB	Estudi de conceptes bàsics	40	-
CBd	Redacció de la base teòrica	15	CB
LB	Llògica booleana: **Recerca i implementació**	15	CB
LBd	Llògica booleana: **Documentació**	5	LB
AN1	Aritmètica bàsica de naturals: **Implementació bàsica**	10	CB, LB
AN2	Aritmètica bàsica de naturals: **Recerca**	30	CB
ANd	Aritmètica bàsica de naturals: **Documentació**	8	AN1, AN2
AI1	Aritmètica bàsica d'enters: **Modificació bàsica**	10	AN1, AN2
AI2	Aritmètica bàsica d'enters: **Recerca**	10	CB, AN2
AId	Aritmètica bàsica d'enters: **Documentació**	2	AI1, AI2
AC	Aritmètica bàsica condicionada: **Recerca**	20	CB, AN1, AN2, AI1, AI2
ACd	Aritmètica bàsica condicionada: **Documentació**	5	AC
P2	Producte per potències de 2 **Recerca**	20	CB
P2d	Producte per potències de 2 **Documentació**	5	P2
PN1	Producte de naturals: **Algorisme bàsic**	10	AC, P2
PN2	Producte de naturals: **Recerca**	20	CB
PNd	Producte de naturals: **Documentació**	8	PN1, PN2
PI1	Producte d'enters: **Modificació bàsica**	10	PN1
PI2	Producte d'enters: **Recerca**	10	CB, PN2
PId	Producte d'enters: **Documentació**	2	PI1, PI2
DN1	Divisió de naturals: **Algorisme bàsic**	10	AC, P2
DN2	Divisió de naturals: **Recerca**	20	CB
DNd	Divisió de naturals: **Documentació**	8	DN1, DN2
DI1	Divisió d'enters: **Modificació bàsica**	10	DN1
DI2	Divisió d'enters: **Recerca**	10	CB, DN2
DId	Divisió d'enters: **Documentació**	2	DI1, DI2
EN1	Exponenciació de naturals: **Algorisme bàsic**	10	PN1, PN2
EN2	Exponenciació de naturals: **Recerca**	20	CB
ENd	Exponenciació de naturals: **Documentació**	5	EN1, EN2
BI	Blocs iteratius: **Recerca**	20	CB
BId	Blocs iteratius: **Documentació**	5	BI
AR	Aritmètica de reals: **Recerca**	40	**CB, LB, AI, AC PI DI, BI**
ARd	Aritmètica de reals: **Documentació**	5	**AR**
ED	Estructures de dades: **Recerca**	40	**CB, LB, AI, AC PI DI, BI**
EDd	Estructures de dades: **Documentació**	5	**ED**
Fd	Finalització del document	30	***d** (CBd, LBd, ANd, ...)
Fp	Preparació de la defensa	20	Fd
-	Total	515	-

Taula 20: *Resum de tasques.* **Font**: elaboració pròpia

5.7 Diagrama de Gantt

A la figura 59 es mostra el diagrama de Gantt corresponent, on es representen les tasques i les seves dependències, ordenades horitzontalment en funció del dia respecte l'inicial en el qual seria òptim començar-les, assumint un nombre il·limitat de desenvolupadors que realitzen unes 15 hores al dia de feina. El treball, però, és desenvolupat per una única persona i per tant totes les 515 hores de feina recauen sobre ell, tot i que cal destacar la importància del disseny paral·lelitzable de les tasques del projecte, el qual permet avançar diferents *fronts* del treball alhora, en el cas que un d'ells requereixi de més temps o esforç, i conseqüentment de la preuada ajuda del director.

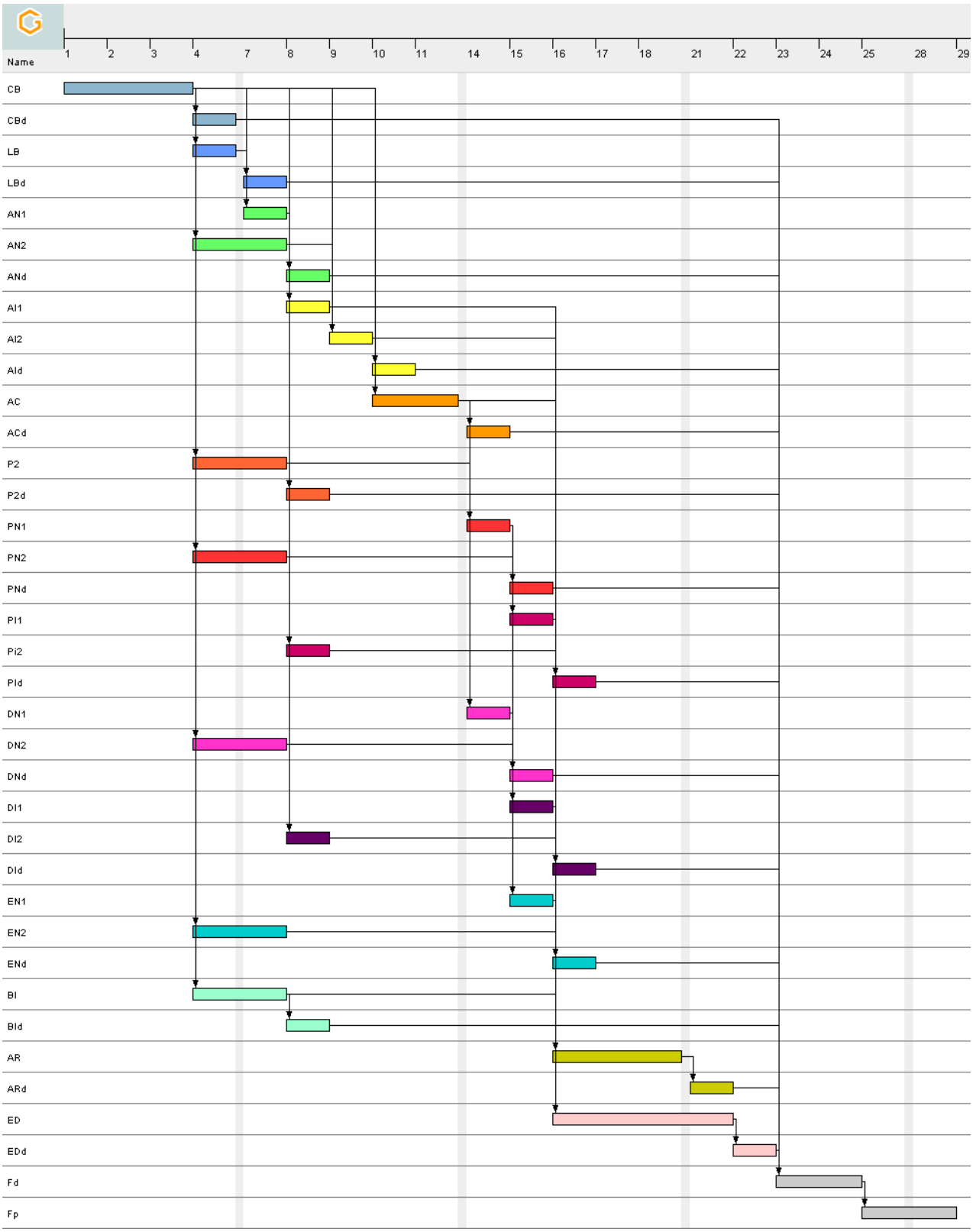


Figura 59: *Diagrama de Gantt de les tasques i dependències.* **Font**: Elaboració pròpia mitjançant [38]

5.8 Planificació d'alternatives i gestió d'imprevistos

L' estimació lleugerament arbitrària dels temps de les tasques, tot i ser el més informat possible, pot no resultar ser del tot acurada un cop s'arribi a les darreres tasques. Aquesta desviació depèn del tipus de tasca, i s'estudia quantitativament en la propera secció, juntament amb l'impacte econòmic d'aquesta possible desviació. Qualitativament, però, ens trobem en una situació on la incertesa d'aquests requeriments temporals poden implicar la impossibilitat de realitzar un subconjunt de les tasques o de completar-les en la seva totalitat, potencialment aturant el procés de desenvolupament del projecte.

És per aquest motius que cal detallar i elaborar possibles alternatives o plans de resposta davant d'imprevistos. Gràcies a la distribució de la feina en tasques majoritàriament independents, el fet de trobar-nos amb qualsevol dificultat pot ser fàcilment remeiat eliminant o bé la part pràctica de qualsevol de les tasques (i realitzar tan sols la cerca de l'estat de l'art) o reduir el temps invertit en la recerca de documents científics, i centrar-se en desenvolupar la implementació, tot indicant a l'informe aquesta decisió.

Tot canvi de pla serà consensuat amb el tutor durant una de les reunions, de manera que caldrà congelar l'estat de la tasca i seguir desenvolupant-ne d'altres gràcies al paral·lelisme del disseny, permetent-nos no perdre temps a causa d'aquests imprevistos. Altres problemes poden consistir en errors tecnològics, com una pèrdua dels documents o corrupció dels arxius. Per a evitar aquests es mantindrà un repositori amb tot el projecte a GitHub, de manera que es minimitza l'efecte d'aquests en permetre obtenir còpies de totes les versions anteriors del treball.

5.9 Gestió econòmica del projecte

En aquesta secció s'avaluarà el cost econòmic del projecte. Com que es tracta d'un treball de caràcter purament investigador, no hi ha un producte pel qual calgui calcular-ne el cost de fabricació o comercialització, de manera que tan sols es tractaran els costos de disseny, planificació, estudi, implementació, recursos humans i materials derivats de la recerca.

5.9.1 Hardware

L'únic requeriment de hardware per a la realització del treball és la d'un ordinador de sobretaula o portàtil. En el nostre cas s'usarà una configuració valorada en 1050€. Assumint una duració del projecte de 515 hores i un temps de vida mitjà de l'equipament de 4 anys de 220 dies laborals i un ús diari de 8 hores, el cost d'amortització del hardware usat és:

$$1050€ \cdot \frac{515 \text{ hores usades en el projecte}}{4 \cdot 220 \cdot 8 \text{ hores de vida útil}} = 76.81€$$

5.9.2 Software

Tot el software usat en aquest projecte és gratuït o de codi obert, o ha estat adquirit mitjançant les llicències d'estudiants facilitades per la facultat. A la taula 21 es llista el programari usat, juntament amb el seu preu usual i el cost que aporta al projecte.

Software	Preu	Cost
Windows 10	139.00€	0.00€
ProjectGantt	0.00€	0.00€
Git	0.00€	0.00€
GitKraken	107.40€	0.00€
LaTeX	0.00€	0.00€
Visual Studio Code	0.00€	0.00€
Qiskit[39]	0.00€	0.00€
Gmail	0.00€	0.00€
Total	246.40€	0.00€

Taula 21: *Costs del software.* **Font:** Elaboració pròpia

5.9.3 Costos de personal per activitat

En ser un projecte individual, el cost dels recursos humans el calcularem com la quantitat de diners que caldria invertir per a contractar tercers per a realitzar cada tasca.

Com que totes les tasques requereixen el mateix conjunt d'habilitats i experiència, no podem classificar-les en funció del tipus d'empleat al qual caldria contractar per a realitzar-les, de manera que assumirem la contractació d'un *Computer Scientist* per a realitzar les tasques, amb un sou per hora de 48.53 € segons *Indeed*[40], el qual implica un cost horari de $48.53 \cdot 1.35 = 65.52$ € amb seguretat social.

La taula 22 summaritza el cost de la realització de cada tasca.

5.9.4 Imprevistos

Tot i que s'ha dissenyat el graf de dependències de tasques de manera que sempre existeixi alguna alternativa en cas que ens trobem amb algun imprevist, és probable que la coordinació o la dificultat de la tasca requereixi una dedicació te temps fora de la previsió inicial. També és possible que errors tecnològics causin una pèrdua important de temps i afegeixin un cost addicional al pressupost donada la necessitat de renovar el hardware.

Per a minimitzar l'efecte d'imprevistos temporals, s'assigna un increment màxim de temps del 30% d'hores a les tasques d'estudi i recerca, i un 10% a les de documentació, amb costos derivats mostrats a la taula 23.

No s'espera haver d'usar totes les hores addicionals estipulades, però és útil tenir un marge per cada tasca per a evitar que imprevistos en una part del projecte es propaguin a la resta de tasques, realentitzant el desenvolupament general.

També cal designar una partida per a averies tècniques o imprevistos tecnològics en general. Tot i que es podria fer un estudi probabilístic i arribar a un valor del cost esperat, considerem que el baix cost del hardware usat i la seva importància vital en el desenvolupament del projecte justifiquen l'adopció d'una quantitat de diners suficient per a reemplaçar tot el material com a partida per a imprevistos tecnològics. Aquesta haurà de ser de no menys de 1050 €per a reemplaçar el hardware i 150 per a possibles despeses de recuperació de disc, en el cas que el repositori *Git* falli.

Tasca	**Nom**	**Hores**	**Cost + SS**
CB	Estudi de conceptes bàsics	40	2,620.80 €
CBd	Redacció de la base teòrica	15	982.80 €
LB	Llògica booleana: **Recerca i implementació**	15	982.80 €
LBd	Llògica booleana: **Documentació**	5	327.60 €
AN1	Aritmètica bàsica de naturals: **Implementació bàsica**	10	655.20 €
AN2	Aritmètica bàsica de naturals: **Recerca**	30	1,965.60 €
ANd	Aritmètica bàsica de naturals: **Documentació**	8	524.16 €
AI1	Aritmètica bàsica d'enters: **Modificació bàsica**	10	655.20 €
AI2	Aritmètica bàsica d'enters: **Recerca**	10	655.20 €
AId	Aritmètica bàsica d'enters: **Documentació**	2	131.04 €
AC	Aritmètica bàsica condicionada: **Recerca**	20	1,310.40 €
ACd	Aritmètica bàsica condicionada: **Documentació**	5	327.60 €
P2	Producte per potències de 2 **Recerca**	20	1,310.40 €
P2d	Producte per potències de 2 **Documentació**	5	327.60 €
PN1	Producte de naturals: **Algorisme bàsic**	10	655.20 €
PN2	Producte de naturals: **Recerca**	20	1,310.40 €
PNd	Producte de naturals: **Documentació**	8	524.16 €
PI1	Producte d'enters: **Modificació bàsica**	10	655.20 €
PI2	Producte d'enters: **Recerca**	10	655.20 €
PId	Producte d'enters: **Documentació**	2	131.04 €
DN1	Divisió de naturals: **Algorisme bàsic**	10	655.20 €
DN2	Divisió de naturals: **Recerca**	20	1,310.40 €
DNd	Divisió de naturals: **Documentació**	8	524.16 €
DI1	Divisió d'enters: **Modificació bàsica**	10	655.20 €
DI2	Divisió d'enters: **Recerca**	10	655.20 €
DId	Divisió d'enters: **Documentació**	2	131.04 €
EN1	Exponenciació de naturals: **Algorisme bàsic**	10	655.20 €
EN2	Exponenciació de naturals: **Recerca**	20	1,310.40 €
ENd	Exponenciació de naturals: **Documentació**	5	327.60 €
BI	Blocs iteratius: **Recerca**	20	1,310.40 €
BId	Blocs iteratius: **Documentació**	5	327.60 €
AR	Aritmètica de reals: **Recerca**	40	2,620.80 €
ARd	Aritmètica de reals: **Documentació**	5	327.60 €
ED	Estructures de dades: **Recerca**	40	2,620.80 €
EDd	Estructures de dades: **Documentació**	5	327.60 €
Fd	Finalització del document	30	1,965.60 €
Dp	Preparació de la defensa	20	0 €
-	Total	515	32,432.40 €

Taula 22: *Cost recursos humans desglossat.* **Font:** Elaboració pròpia.

Tasca	Nom	Hores	Cost + SS	Imprevistos
CB	Estudi de conceptes bàsics	40	2,620.80€	786.24€
CBd	Redacció de la base teòrica	15	982.80 €	98.28€
LB	Llògica booleana: **Recerca i implementació**	15	982.80 €	294.84€
LBd	Llògica booleana: **Documentació**	5	327.60 €	32.76€
AN1	Aritmètica bàsica de naturals: **Implementació bàsica**	10	982.80 €	196.56€
AN2	Aritmètica bàsica de naturals: **Recerca**	30	1,965.60 €	589.68€
ANd	Aritmètica bàsica de naturals: **Documentació**	8	524.16 €	52.42€
AI1	Aritmètica bàsica d'enters: **Modificació bàsica**	10	655.20 €	196.56€
AI2	Aritmètica bàsica d'enters: **Recerca**	10	655.20 €	196.56€
AId	Aritmètica bàsica d'enters: **Documentació**	2	131.04 €	13.11€
AC	Aritmètica bàsica condicionada: **Recerca**	20	1,310.40 €	393.12€
ACd	Aritmètica bàsica condicionada: **Documentació**	5	327.60 €	32.76€
P2	Producte per potències de 2 **Recerca**	20	1,310.40 €	393.12€
P2d	Producte per potències de 2 **Documentació**	5	327.60 €	32.76€
PN1	Producte de naturals: **Algorisme bàsic**	10	655.20 €	196.56€
PN2	Producte de naturals: **Recerca**	20	1,310.40 €	393.12€
PNd	Producte de naturals: **Documentació**	8	524.16 €	52.42€
PI1	Producte d'enters: **Modificació bàsica**	10	655.20 €	196.56€
PI2	Producte d'enters: **Recerca**	10	655.20 €	196.56€
PId	Producte d'enters: **Documentació**	2	131.04 €	13.10€
DN1	Divisió de naturals: **Algorisme bàsic**	10	982.80 €	196.56€
DN2	Divisió de naturals: **Recerca**	20	1,310.40 €	393.12€
DNd	Divisió de naturals: **Documentació**	8	524.16 €	52.42€
DI1	Divisió d'enters: **Modificació bàsica**	10	655.20 €	196.56€
DI2	Divisió d'enters: **Recerca**	10	655.20 €	196.56€
DId	Divisió d'enters: **Documentació**	2	131.04 €	13.10€
EN1	Exponenciació de naturals: **Algorisme bàsic**	10	655.20 €	196.56€
EN2	Exponenciació de naturals: **Recerca**	20	1,310.40 €	393.12€
ENd	Exponenciació de naturals: **Documentació**	5	327.60 €	32.76€
BI	Blocs iteratius: **Recerca**	20	1,310.40 €	393.12€
BId	Blocs iteratius: **Documentació**	5	327.60 €	32.76€
AR	Aritmètica de reals: **Recerca**	40	2,620.80 €	786.24€
ARd	Aritmètica de reals: **Documentació**	5	327.60 €	32.76€
ED	Estructures de dades: **Recerca**	40	2,620.80 €	786.24€
EDd	Estructures de dades: **Documentació**	5	327.60 €	32.76€
Fd	Finalització del document	30	1,965.60 €	196.50€
Fp	Preparació de la defensa	20	0 €	0€
-	Total	515	32,432.40 €	8,681.40€

Taula 23: *Efecte d'imprevistos sobre el cost de recursos humans.* **Font:** Elaboració pròpia

5.9.5 Costs genèrics

El principal cost indirecte del projecte és el cost de l'electricitat consumida per l'ordinador i els perifèrics, el qual pot ser molt variable donada la situació geopolítica actual. És imperatiu, per tant, calcular el consum elèctric esperat durant la duració del treball i fitar el cost monetari d'aquest.

El consum conjunt de l'ordinador de sobretaula i el monitor s'ha estimat en uns 400 Watts, de manera que podem calcular l'energia consumida durant les 515 hores esperades de treball:

$$515\text{h} \cdot \frac{400\text{Wh}}{1\text{h}} \cdot \frac{1\text{MWh}}{1000000\text{Wh}} = 0.216\text{MWh}$$

Tot i l'alta fluctuació actual del preu del MWh[41], assumirem una mitjana conservativa de 300 €per MWh, de manera que el preu total esperat causat pel consum energètic és:

$$0.216\text{MWh} \cdot \frac{300€}{1\text{MWh}} = 64.8€$$

5.9.6 Contingència

Es reserva una part dels diners per a qualsevol altre imprevist dels descrits, per a garantintzar un desenvolupament del projecte ininterromput per inconvenients monetaris. Aquesta reserva consisteix en uns diferents percentatges depenent de l'origen del cost, tal i com es mostra a la taula 24.

Origen	**Cost(€)**	**Percentatge**	**Cost contingència**
Hardware	76.81€	15%	11.52€
Software	0.000€	5%	0€
Recursos Humans	32,432.40€	5%	1,621.62€
Imprevistos RRHH	8,681.40€	5%	434.07€
Imprevistos de hardware	1200.00€	15%	180€
Costs genèrics	64.80€	30%	19.44€
Total	42,455.41€	-	2,266.65€

Taula 24: *Pressupost de contingència.* **Font:** Elaboració pròpia

5.9.7 Pressupost final

Finalment, summaritzem en la taula 25 els diferents costos monetaris del projecte i les partides dedicades a imprevistos, tot computant el pressupost final requerit per a realitzar el treball.

Origen	Cost(€)	Percentatge	Cost contingència	Cost final
Hardware	76.81€	15%	11.52€	88.33€
Software	0.000€	5%	0€	0€
Recursos Humans	32,432.40€	5%	1,621.62€	34,054.02€
Imprevistos RRHH	8,681.40€	5%	434.07€	9,115.47€
Imprevistos de hardware	1200.00€	15%	180€	1,380€
Costs genèrics	64.80€	30%	19.44€	84.24€
Total	42,455.41€	-	2,266.65€	44,722.06

Taula 25: *Pressupost final.* **Font:** Elaboració pròpia

5.9.8 Control de gestió

Per a control·lar constantment el correcte ús del pressupost econòmic i temporal, definim indicadors que representen les desviacions d'aquestes dues magnituds respecte els seus valors ideals, assignats en les seccions anteriors. Més concretament, aquests indicadors calculen la diferència entre els costos esperats i reals tant de temps com de diners:

$$\begin{aligned} \text{Desviació de cost} &= C_r - C_i \\ \text{Desviació temporal} &= T_r - T_i \end{aligned}$$

Podem ponderar aquests dos indicators per a obtenir magnituds monetàries, de manera que podem analitzar i comparar els efectes de les desviacions al pressupost:

$$\begin{aligned} \text{Desviació de cost} &= (C_r - C_i) \cdot T_r \\ \text{Desviació temporal} &= (T_r - T_i) \cdot C_r \end{aligned}$$

On C_x correspon al cost en $\frac{€}{hora}$, i T_x correspon al temps en hores, on x és r o i per a referir-se al valor real o l'ideal, respectivament.

Aquests indicadors s'usaran per a mantenir un control constant sobre l'estat del treball general o una certa tasca individual. Quan aquests valors prenguin valors alts caldrà replantejar les assignacions de temps i pressupost per a evitar que els inconvenients es propaguin a tasques successores.

5.9.9 Viabilitat econòmica

Com que aquest és un projecte d'investigació sense ànim de lucre, la seva viabilitat econòmica depèn de la subvencions per part dels actors implicats. En aquest cas la càrrega econòmica recaurà sobre el desenvolupador, ja que és ell qui realitzarà les tasques en les quals s'ha subdividit el treball, de manera que el cost de recursos humans es pot ignorar.

A la pràctica, els únics costos a tenir en compte són aquells genèrics derivats del cost de l'electricitat i els costos de hardware, tant d'amortització com imprevistos(en el rar cas que sigui necessari). Això redueix el cost del projecte a un valor in l'interval [172.57 €, 1,552.57 €] el qual és completament assequible per a un estudiant.

És impossible predir tots els obstacles que poden impactar negativament el cost del projecte, però gràcies a l'anàlisi realitzat en les seccions anteriors i al marge generós per a imprevistos podem afirmar amb alta confiança que la probabilitat de requerir una quantitat de diners major a la dedicada és negligible, de manera que el pressupost obtingut és adient.

6 Informe de Sostenibilitat

Poc usualment es considera la sostenibilitat a l'hora de desenvolupar un treball de recerca, però a dia d'avui les tecnologies permeen tots els àmbits de la societat, i les conseqüències d'ignorar un pilar tan bàsic com aquest poden ser devastadores.

Des de genètica a física nuclear, l'innocent desig de conèixer i aprendre sovint eclipsa el necessari debat sobre la sostenibilitat del projecte o idea en qüestió. L'ús indecent o immoral de tecnologies no és pas un concepte aliè, de manera que considerem vital l'anàlisi dels motius i conseqüències de qualsevol treball d'investigació, i creiem necessària l'estudi de les seves repercussions en tots els àmbits possibles per a assegurar un desenvolupament i un producte sostenibles.

El testament del nostre compromís amb la sostenibilitat té poc valor si no és acompanyat d'una voluntat d'aprenentatge sobre el tema i una clara declaració d'intencions sobre el *com* es garantitzarà aquesta sostenibilitat del desenvolupament del projecte i del producte final. A continuació, desenvolupem els aspectes *econòmics*, *ambientals*, i *socials* de la sostenibilitat en relació al nostre projecte. Analitzar aquesta *matriu de sostenibilitat* ens permetrà, entre altres, obtenir una visió clara dels requeriments i efectes del nostre treball més enllà de la investigació, i ens donarà la oportunitat de presentar la nostra clara intenció de desenvolupar el projecte amb una visió i objectius basats en la sostenibilitat.

6.1 Dimensions de la sostenibilitat

6.1.1 Dimensió econòmica

Tal i com s'ha mencionat anteriorment a la secció 5.9.9, el cost econòmic real del projecte és de, com a màxim, 1,552.57€. Pel que fa als beneficis, cal destacar que el treball no té ànim de lucre, és a dir, l'únic i insubstituïble objectiu de la investigació és la recopil·lació i summarització de coneixements publicats per altres investigadors, construir coneixement a sobre i publicar-lo gratuïtament a la comunitat científica, per a ajudar i motivar futura recerca sobre els tòpics tractats i facilitar el desenvolupament d'aquest àmbit de recerca.

Més concretament, aquest projecte pretén elaborar un document que recopil·li comprensivament les millors tècniques per a implementar còmputs clàssics sobre estats quàntics en superposició eficientment. Tot i que, com s'ha esmentat anteriorment, no es té ànim de lucre, cal destacar que el projecte pot tenir com a conseqüència un gran estalvi monetari i energètic per a empreses o individus, en el cas que la computació quàntica esdevingui més prominent en un futur.

6.1.2 Dimensió ambiental

Els costos ambientals causats per la realització del projecte són inferibles a partir de les dades presentades en la secció 5.9.5. Com que tot el que cal per al desenvolupament del treball és un ordinador i els seus perifèrics, l'efecte d'aquest desenvolupament en el medi ambient és equivalent al cost energètic en quilowatts hora consumits, d'un total esperat de 206kWh, el qual correspon a l'impacte ambiental durant l'etapa del projecte posat en producció (PPP).

En el pitjor dels casos podem assumir que la totalitat de l'electricitat prové d'una planta tèrmica de crema de combustibles fòssils. Podem calcular una producció màxima de 1.043Kg de CO_2 per kWh consumit[42], de manera que en total es produiran 214.85Kg de CO_2 durant el projecte.

Cal també destacar el principal potencial benefici que pot ser conseqüència del projecte: l'eficiència de càlculs quàntics. Gràcies a aquest treball, tal i com s'ha comentat anteriorment, es pot reduir l'energia requerida per a realitzar còmputs en ordinadors quàntics, reduint així la petjada de carboni futura d'aquests computadors.

Per acabar, recordem que la computació quàntica no suposa un risc ambiental important actualment ni es preveu que ho sigui en el futur, de manera que no existeixen perills pel medi ambient derivats d'aquest estudi.

6.1.3 Dimensió social

Individualment, la realització del projecte oferirà al desenvolupador la possibilitat d'aprendre conceptes avançats d'uns camps amb tant futur com en són la física i computació quàntiques, les quals han revolucionat el món de la computació i tenen unes perspectives de futur encara més optimistes.

Pel que fa a l'àmbit social, aquest projecte no pretén ser trencador pel que fa a l'eficiència dels dissenys de càlcul clàssic publicats, però sí que tracta d'oferir de manera summaritzada però extensiva les diverses tècniques i circuits dissenyats i publicats que realitzen aquests càlculs. El principal efecte d'aquest treball és, doncs, facilitar l'accés a aquesta informació tan negligida en la literatura quàntica, oferint a la comunitat científica un punt de partida de cara a dissenyar i implementar circuits o algorismes més avançats, accelerant el desenvolupament i innovació d'aquest camp substancialment.

De la mateixa manera que en l'apartat anterior, la possibilitat que el projecte impacti de manera negativa la societat és ínfim, ja que tan sols s'està facilitant l'accés a informació ja publicada, informació que, tal i com s'ha explicat anteriorment, no es considera perillosa ni amb potencial de causar mal.

6.2 Matriu de sostenibilitat

Per a resumir els conceptes anteriors presentem seguidament la matriu de sostenibilitat a la taula 26.

	PPP	**Vida útil**	**Riscs**
Econòmica	1 338.85€	No beneficis	Probabilitat negligible
Ambiental	214.85KgCO_2	Eficiència energètica	Cap
Social	Adquisició de coneixements	Accés a informació	Cap

Taula 26: *Matriu de sostenibilitat.* **Font:** Elaboració pròpia.

7 Referències

[1] G. Florio and D. Picca, "Implementation of analytic functions with quantum gates," 2004.

[2] V. Vedral, A. Barenco, and A. Ekert, "Quantum networks for elementary arithmetic operations," *Physical Review A*, vol. 54, p. 147–153, Jul 1996.

[3] T. Haener, M. Soeken, M. Roetteler, and K. M. Svore, "Quantum circuits for floating-point arithmetic," in *RC*, 2018.

[4] T. G. Draper, "Addition on a quantum computer," 2000.

[5] E. Şahin, "Quantum arithmetic operations based on quantum fourier transform on signed integers," *International Journal of Quantum Information*, vol. 18, p. 2050035, Sep 2020.

[6] P. Shor, "Foundations of computer science, 1994 proceedings, 35th annual symposium on," 1994.

[7] V. Vedral, A. Barenco, and A. Ekert, "Quantum networks for elementary arithmetic operations," *Physical Review A*, vol. 54, no. 1, p. 147, 1996.

[8] D. Beckman, A. N. Chari, S. Devabhaktuni, and J. Preskill, "Efficient networks for quantum factoring," *Physical Review A*, vol. 54, no. 2, p. 1034, 1996.

[9] T. G. Draper, S. A. Kutin, E. M. Rains, and K. M. Svore, "A logarithmic-depth quantum carry-lookahead adder," *arXiv preprint quant-ph/0406142*, 2004.

[10] Y. Takahashi and N. Kunihiro, "A fast quantum circuit for addition with few qubits," *Quantum Information & Computation*, vol. 8, no. 6, pp. 636–649, 2008.

[11] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, "A new quantum ripple-carry addition circuit," *arXiv preprint quant-ph/0410184*, 2004.

[12] F. Wang, M. Luo, H. Li, *et al.*, "Improved quantum ripple-carry addition circuit," *Sci China Inf Sci*.

[13] Y. Takahashi and N. Kunihiro, "A linear-size quantum circuit for addition with no ancillary qubits," *Quantum Information & Computation*, vol. 5, no. 6, pp. 440–448, 2005.

[14] L. Ruiz-Perez and J. C. Garcia-Escartin, "Quantum arithmetic with the quantum fourier transform," *Quantum Information Processing*, vol. 16, no. 6, pp. 1–14, 2017.

[15] E. Şahin, "Quantum arithmetic operations based on quantum fourier transform on signed integers," *International Journal of Quantum Information*, vol. 18, no. 06, p. 2050035, 2020.

[16] C. Zalka, "Fast versions of shor's quantum factoring algorithm," *arXiv preprint quant-ph/9806084*, 1998.

[17] Y. Takahashi, S. Tani, and N. Kunihiro, "Quantum addition circuits and unbounded fan-out," *arXiv preprint arXiv:0910.2530*, 2009.

[18] R. N. Bracewell and R. N. Bracewell, *The Fourier transform and its applications*, vol. 31999. McGraw-hill New York, 1986.

[19] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[20] J. Daboul, X. Wang, and B. C. Sanders, "Quantum gates on hybrid qudits," *Journal of Physics A: Mathematical and General*, vol. 36, no. 10, p. 2525, 2003.

[21] I. Fushman, D. Englund, A. Faraon, N. Stoltz, P. Petroff, and J. Vuckovic, "Controlled phase shifts with a single quantum dot," *science*, vol. 320, no. 5877, pp. 769–772, 2008.

[22] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Physical review A*, vol. 52, no. 5, p. 3457, 1995.

[23] M. Saeedi and M. Pedram, "Linear-depth quantum circuits for n-qubit toffoli gates with no ancilla," *Physical Review A*, vol. 87, no. 6, p. 062318, 2013.

[24] P. Gossett, "Quantum carry-save arithmetic," *arXiv preprint quant-ph/9808061*, 1998.

[25] R. Van Meter and K. M. Itoh, "Fast quantum modular exponentiation," *Physical Review A*, vol. 71, no. 5, p. 052320, 2005.

[26] P. Pham, *Low-depth quantum architectures for factoring*. PhD thesis, 2014.

[27] J. Álvarez-Sánchez, J. Álvarez-Bravo, and L. Nieto, "A quantum architecture for multiplying signed integers," in *Journal of Physics: Conference Series*, vol. 128, p. 012013, IOP Publishing, 2008.

[28] A. Pavlidis and D. Gizopoulos, "Fast quantum modular exponentiation architecture for shor's factorization algorithm," *arXiv preprint arXiv:1207.0511*, 2012.

[29] A. Khosropour, H. Aghababa, and B. Forouzandeh, "Quantum division circuit based on restoring division algorithm. 2011 eighth int," in *Conf. Inf. Technol. New Gener*, pp. 3–6.

[30] H. Thapliyal, E. Munoz-Coreas, T. Varun, and T. S. Humble, "Quantum circuit designs of integer division optimizing t-count and t-depth," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 2, pp. 1045–1056, 2019.

[31] S. Beauregard, "Circuit for shor's algorithm using 2n+ 3 qubits," *arXiv preprint quant-ph/0205095*, 2002.

[32] T. Häner, M. Roetteler, and K. M. Svore, "Factoring using 2n+ 2 qubits with toffoli based modular multiplication," *arXiv preprint arXiv:1611.07995*, 2016.

[33] J. A. Miszczak, "Efficient quantum algorithm for factorization," *Archiwum Informatyki Teoretycznej i Stosowanej*, vol. 2, p. 14, 2002.

[34] P. Pham and K. M. Svore, "A 2d nearest-neighbor quantum architecture for factoring in polylogarithmic depth," *arXiv preprint arXiv:1207.6655*, 2012.

[35] I. L. Markov and M. Saeedi, "Constant-optimized quantum circuits for modular multiplication and exponentiation," *arXiv preprint arXiv:1202.6614*, 2012.

[36] H. Jayashree, H. Thapliyal, and V. K. Agrawal, "Design of dedicated reversible quantum circuitry for square computation," in *2014 27th international conference on VLSI design and 2014 13th international conference on embedded systems*, pp. 551–556, IEEE, 2014.

[37] H. Jayashree, H. Thapliyal, and V. K. Agrawal, "Efficient circuit design of reversible square," in *Transactions on Computational Science XXIX*, pp. 33–46, Springer, 2017.

[38] GanttProject, "Free desktop project scheduling app — GanttProject." `https://www.ganttproject.biz/`, 2022.

[39] Quiskit, "Open-Source Quantum Development — Quiskit." `https://qiskit.org/`, 2022.

[40] Indeed, "Comparación de sueldos — Indeed.com." `https://www.indeed.com/salaries`.

[41] OMIE, "Operador de mercado eléctrico designado — OMIE." `https://www.omie.es/es/`, 2022.

[42] coaltrans, "Carbon intensity of coal per KWh — Coaltrans." `https://www.coaltrans.com/insights/article/ecocarbon-august-2021`, 2022.

A Base matemàtica per a computació quàntica

Els estats d'un o un conjunt de qubits poden ser representats mitjançant vectors, i els operadors quàntics es poden definir usant matrius, de manera que és imperatiu tenir uns coneixements bàsics de càlcul vectorial per a poder entendre el funcionament d'aquest model de computació. Aquesta secció pretén aportar aquests coneixements sobre espais vectorials necessaris per a la lectura del document.

A.1 Nombres complexos

Els estats i operadors quàntics es defineixen usant espais vectorials sobre el conjunt de nombres complexos, de manera que cal conèixer les propietats d'aquests nombres per a poder entendre el seu funcionament.

Un nombre complex és aquell que té una part real i una imaginària, sovint representat com:

$$z = a + bi$$

On i és la unitat imaginaria definida per:

$$i^2 = -1$$

La suma i resta d'aquests nombres és fàcilment deduïble:

$$(a + bi) \pm (c + di) = (a \pm c) + (b \pm d)i$$

El producte es pot obtenir mitjançant la definició de i:

$$(a + bi) \cdot (c + di) = ac + adi + cbi + bdi^2 = (ac - bd) + (ad + cb)i$$

I la divisió es pot obtenir invertint el producte i resolent el sistema:

$$\frac{a + bi}{c + di} = x + yi \implies a + bi = (c + di) \cdot (x + yi) \implies a + bi = (cx - dy) + (cy + dy)i$$

$$\left.\begin{array}{l} a = cx - dy \\ b = cy + dx \end{array}\right\} \begin{array}{l} x = \frac{ac+bd}{c^2+b^2} \\ y = \frac{bc-ad}{c^2+b^2} \end{array}$$

De manera que:

$$\frac{a + bi}{c + di} = \frac{(ac + bd) + (bc - ad)i}{c^2 + b^2}$$

Una manera alternativa de representar nombres complexos és mitjançant la formula d'Euler:

$$e^{i\theta} = cos(\theta) + sin(\theta)i$$

Permetent-nos expressar un nombre complex arbitrari mitjançant el seu mòdul i el seu angle respecte l'eix real en el pla complex:

$$z = a + bi = re^{i\theta}$$

on

$$\begin{array}{c} r^2 = |z|^2 = a^2 + b^2 \\ \theta = tan^{-1}(\frac{b}{a}) \end{array}$$

Aquesta és la forma *polar* del nombre complex z, i podem obtenir la forma anterior, anomenada forma *cartesiana*, amb trigonometria bàsica:

$$\left.\begin{array}{l} a = r \cdot cos(\theta) \\ b = r \cdot sin(\theta) \end{array}\right\} z = r \cdot cos(\theta) + r \cdot sin(\theta)i$$

Aquesta representació permet redefinir el producte i quocient entre complexos:

$$r_1 e^{i\theta_1} \cdot r_2 e^{i\theta_2} = (r_1 r_2) e^{i(\theta_1+\theta_2)}$$

$$\frac{r_1 e^{i\theta_1}}{r_2 e^{i\theta_2}} = \frac{r_1}{r_2} e^{i(\theta_1-\theta_2)}$$

Sobre els nombres complexos també es defineix la operació de conjugat, que consisteix en canviar el signe de la part imaginària:

$$z^* = (a+bi)^* = a - bi$$

Aquesta operació permet redefinir el quadrat del mòdul d'un nombre complex:

$$|z|^2 = |a+bi|^2 = a^2 + b^2 = z \cdot z^*$$

A.2 Àlgebra lineal

Com que els estats de sistemes quàntics es comporten com a vectors de nombres complexos, cal tenir uns fonaments d'àlgebra lineal per a poder comprendre el funcionament del model quàntic de computació.

A.2.1 Espais vectorials

De manera general, anomenem vectors als elements d'un *espai vectorial* i els representem amb una lletra minúscula com v o u. La principal propietat d'aquests espais vectorials és que són tancats respecte la suma i el producte per escalars:

$$v = w + u$$

$$v = a \cdot u$$

És a dir, si u i w són elements de l'espai vectorial, v també ho és.

El terme a fa referència a un element d'un cos, el qual és un conjunt tancat per suma, resta, producte i divisió (excepte per 0). Aquest cos sol ser el conjunt de nombres reals $\mathbb{R}$ o el conjunt de nombres complexos $\mathbb{C}$.

Conseqüentment, una combinació lineal de vectors pertany també a l'espai vectorial:

$$v = a_1 u_1 \ + \ a_2 u_2 \ + \ \ldots \ + \ a_n u_n$$

Un conjunt de vectors s'anomena *linealment independent* si cap dels vectors del conjunt pot ser expressat com a una combinació lineal dels altres:

$$u_i \neq a_1 u_1 \ + \ \ldots \ + \ u_{i-1} a_{i-1} \ + \ u_{i+1} a_{i+1} \ + \ \ldots \ + \ a_n u_n$$

Aquesta expressió se sol reescriure en la seva forma més general:

$$0 \neq a_1 u_1 \ + \ a_2 u_2 \ + \ \ldots \ + \ a_n u_n$$

És a dir, un conjunt de vectors és linealment independent si i només si és impossible obtenir l'element neutre 0 mitjançant una combinació lineal no trivial ($a_i \neq 0$) dels seus elements.

Anomenem *base* a un conjunt de vectors linealment independents capaços de representar qualsevol vector de l'espai mitjançant una combinació lineal:

$$v = a_1 u_1 \ + \ a_2 u_2 \ + \ \ldots \ + \ a_n u_n$$

On n és el tamany d'aquest conjunt de vectors i és igual a la *dimensió* de l'espai vectorial.

Gràcies al fet que qualsevol vector pot ser expressat en termes d'aquesta base, podem representar-lo indicant tan sols els coeficients a_i de la forma:

$$v = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$$

Aquesta representació sempre ha d'anar relacionada a una base per a poder reconstruir l'estat com a combinació lineal dels coeficients amb els vectors bàsics.

Sobre aquests vectors definim les anomenades *transformacions lineals*, les quals són funcions que compleixen:

$$f(a \cdot u + b \cdot v) = a \cdot f(u) + b \cdot f(v)$$

De manera que el resultat d'una transformació lineal sobre un vector pot ser definit en funció dels resultats de la funció aplicada sobre els vectors bàsics:

$$\begin{aligned} f(v) &= f(a_1 u_1 \ + \ a_2 u_2 \ + \ \ldots \ + \ a_n u_n) \\ &= a_1 f(u_1) \ + \ a_2 f(u_1) \ + \ \ldots \ + \ a_n f(u_n) \end{aligned}$$

I sabem, per la definició de vector, que cadascun dels termes $f(u_1)$ és un vector, de manera que el resultat d'aplicar una transformació lineal f sobre un vector v és una combinació lineal dels productes de f sobre la base, el qual se sol representar mitjançant el producte matricial:

$$\begin{aligned} f(v) &= a_1 f(u_1) \ + \ a_2 f(u_1) \ + \ \ldots \ + \ a_n f(u_n) \\ &= \begin{bmatrix} | & | & \ldots & | \\ f(u_1) & f(u_2) & \ldots & f(u_n) \\ | & | & \ldots & | \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \end{aligned}$$

El qual ens permet definir un operador mitjançant la seva notació matricial, on cada columna és el seu efecte sobre cada vector de la base.

Si definim $f(u_j)_i$ com el coeficient del vector bàsic u_i del vector resultant d'aplicar f sobre el vector bàsic u_j, podem completar la matriu:

$$f = \begin{bmatrix} f(u_0)_0 & \ldots & f(u_n)_0 \\ \vdots & \ddots & \vdots \\ f(u_0)_n & \ldots & f(u_n)_n \end{bmatrix}$$

Sobre aquestes matrius es defineix l'operació de transposició, el qual consisteix en canviar les files per les columnes:

$$f = \begin{bmatrix} f_{00} & \ldots & f_{n0} \\ \vdots & \ddots & \vdots \\ f_{0n} & \ldots & f_{nn} \end{bmatrix}^T = \begin{bmatrix} f_{00} & \ldots & f_{0n} \\ \vdots & \ddots & \vdots \\ f_{n0} & \ldots & f_{nn} \end{bmatrix}$$

A.2.2 Espai de Hilbert complex

Un espai de Hilbert és un espai vectorial equipat amb una operació entre vectors de l'espai anomenada producte escalar, o *inner product* en anglès. Aquest producte escalar entre dos vectors v i u es denota $< v, u >$ i retorna un escalar.

Els espais de Hilbert solen ser definits sobre un espai vectorial de n dimensions de complexos, o $\mathbb{C}^n$, de manera que qualsevol vector té la forma:

$$v = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}$$

Una base òbvia d'aquest espai vectorial seria:

$$\left\{ \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \cdots, \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \right\}$$

La qual és anomenada base canònica i, com qualsevol altra base, permet representar un vector arbitrari de l'espai v com a una combinació lineal dels elements d'aquesta:

$$v = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} = z_1 \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + z_2 \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} + \cdots + z_n \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

De manera que podem definir les operacions de suma i producte escalar característiques de tot espai vectorial:

$$v + u = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} + \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} v_1 + u_1 \\ v_2 + v_2 \\ \vdots \\ v_n + u_n \end{bmatrix}$$

$$z \cdot v = z \cdot \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} z \cdot v_1 \\ z \cdot v_2 \\ \vdots \\ z \cdot v_n \end{bmatrix}$$

On el cos d'escalars és el conjunt de complexos $\mathbb{C}$.

Els vectors i matrius d'aquest espai compleixen totes les propietats esperades d'un espai vectorial, i es defineix la transposa conjugada d'una matriu com:

$$f = \begin{bmatrix} f_{00} & \cdots & f_{n0} \\ \vdots & \ddots & \vdots \\ f_{0n} & \cdots & f_{nn} \end{bmatrix}^\dagger = \begin{bmatrix} f_{00}* & \cdots & f_{0n}* \\ \vdots & \ddots & \vdots \\ f_{n0}* & \cdots & f_{nn}* \end{bmatrix}$$

El qual consisteix simplement en una transposició seguida del complex conjugat de cada element de la matriu.

El producte escalar de dos vectors d'aquest espai es defineix de la manera següent:

$$< v, u >= v^{\dagger}u = \begin{bmatrix} v_1* & v_2* \dots v_n* \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}$$

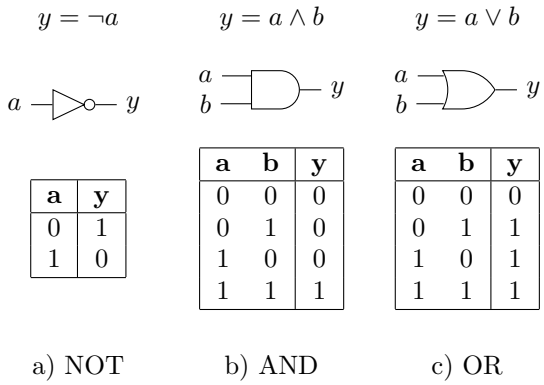
El qual és 0 si i només si els vectors v i u són ortogonals.

B Àlgebra booleana i computació clàssica

Aquesta secció descriu la lògica booleana i el seu ús en la representació d'informació i computació clàssica. La definició és menys formal de la habitual en la presentació de lògiques matemàtiques donada la simplicitat i intuïtivitat d'aquesta.

B.1 Àlgebra booleana

L'àlgebra booleana és una lògica en la que les variables poden prendre tan sols dos valors, anomenats **0** (o **fals**) i **1** (o **cert**). Sobre aquestes variables es defineix la operació unària de negació(**NOT**) i les operacions binàries de conjunció (**AND**) i disjunció(**NOT**), amb notacions i **taules de la veritat** mostrades en la figura 60.



a	y
0	1
1	0

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

a	b	y
0	0	0
0	1	1
1	0	1
1	1	1

a) NOT b) AND c) OR

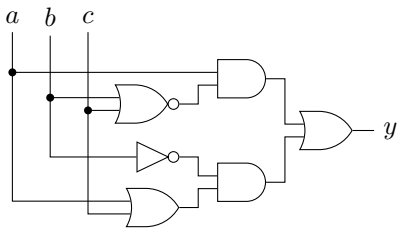
Figura 60: *Les tres operacions bàsiques en àlgebra booleana: negació(a), conjunció(b) i disjunció(c), amb els seus símbols, digrames i taules de la veritat.* **Font**: Elaboració pròpia mitjançant Tikz.

Aquestes operacions es poden aplicar sobre expressions booleanes també, fent possible la creació de predicats amb taules de la veritat més complexes com el mostrat en la figura 61.

B.2 Representació de nombres

La unitat d'informació clàssica bàsica és el **bit**, el qual és equivalent a una variable booleana en el fet que tan sols pot adoptar els valors **0** (o **fals**) i **1** (o **cert**). El nostre objectiu és representar informació arbitrària mitjançant un conjunt d'aquests bits, el qual es pot aconseguir dissenyant una bijecció entre $\{0,1\}^n$ i el conjunt d'elements del que sigui que volem representar.

$$y = a \wedge \neg(b \vee c) \vee \neg b \wedge (a \vee c)$$



a	b	c	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Figura 61: *Formula, diagrama i taula de la veritat d'un circuit lògic format per 1 porta NOT, 2 portes OR, 2 portes AND i 1 porta NOR (OR+NOT).* **Font**: Elaboració pròpia mitjançant Tikz.

B.2.1 Naturals

L'exemple més senzill és el de la representació de nombres naturals, ja que la codificació de la informació en bits ve definida per un canvi de base:

$$\{0,1\}^n \to \mathbb{N} \cap [0, 2^n - 1]$$

$$f(x) = \sum_{i=0}^{n-1} x_i 2^i$$

On x_i és l'íessim element del vector booleà $x \in \{0,1\}^n$: Per a $n = 8$, la codificació del nombre 151 seria:

Valor							
x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0
1	0	0	1	0	1	1	1

Gràcies a aquesta bijecció, podem representar qualsevol natural en el rang $[0, 2^n - 1]$ inequívocament mitjançant un conjunt de n bits, de manera que hem assolit el nostre objectiu de codificar informació en un conjunt finit de bits.

B.2.2 Enters

Un altre tipus d'informació que ens podria interessar representar és nombres enters, els quals es podrien codificar trivialment com una combinació de natural + signe($0 = +$, $1 = -$) en n+1 bits. Aquesta codificació pel nombre -87 és:

Signe	**Valor**							
s	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0
1	0	1	0	1	0	1	1	1

A la pràctica, però, s'usa una altra codificació anomenada *complement a 2*. Aquesta es basa en la operació del complement a 2 d'un nombre de n bits x, la qual consisteix en trobar el nombre y tal que $x + y = 2^n$. Per exemple el complement a dos del nombre $3 = 011_2$ és $5 = 101_2$ ja que $011_2 + 101_2 = 1000_2 \equiv 000_2 \pmod{2^n}$. Aquesta definició és similar a la del negat dels nombres reals, on la suma dels oposats és zero, de manera que es defineix la negació de nombres de n bits mitjançant aquest complement a dos. A continuació es mostren els valors de tots els nombres de tres bits possibles si aquests codifiquen enters en complement a dos.

bits	**Valor**	**Ca2**	**Valor**
000	0	000	0
001	1	111	-1
010	2	110	-2
011	3	101	-3
100	-4	100	-4
101	-3	011	3
110	-2	010	2
111	-1	001	1

L'utilitat d'aquesta representació s'explicarà a continuació, però una de les característiques més importants és la capacitat de realitzar sumes sense haver de tenir en compte el cas especial del canvi de signe, per exemple, si comencem amb el nombre -3 codificat amb signe+valor i complement a dos i anem sumant 1, obtenim:

Valor	-3	-2	-1	0	1	2	3
Signe+Valor	111	110	101	100 o 000	001	010	011
Ca2	101	110	111	000	001	010	011

Es pot veure que en complement a dos, cada valor s'obté amb la suma en base dos normal, mentre que la codificació de Signe+Valor té dues representacions del 0. Amb Ca2 podem codificar en n bits qualsevol valor del rang $[-2^{n-1}, 2^{n-1} - 1]$.

B.2.3 Reals

És impossible representar tots els elements de qualsevol interval $[a, b]$ de nombres reals amb un nombre finit de bits, de manera que caldrà definir una codificació amb precisió limitada.

La codificació més obvia és l'anomenada *coma fixa*, la qual consisteix en afegir una coma a un enter codificat amb Signe+Valor. Per exemple, podem codificar decimals amb 8 bits afegint una coma fixa entre qualsevol parell de bits:

Signe	**Valor enter**				**Valor decimal**			
s	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0
1	0	1	0	1	0	1	1	1

Aquest conjunt de bits es pot descodificar de manera anàloga a Signe+Valor:

$$(-1)^s \cdot (x_7 2^3 + x_6 2^2 + x_5 2^1 + x_4 2^0 + x_3 2^{-1} + x_2^{-2} + x_1 2^{-3} + x_0 2^{-4}) = -5.4375$$

O equivalentment:

$$(-1)^s \cdot 2^{-4} \cdot \mathbf{x}_2 = -\frac{87}{2^4} = -5.4375$$

Cal notar que x_2 es refereix al tercer element del vector de bits $\mathbf{x}$, mentre que $\mathbf{x}_2$ es refereix al valor del vector de bits $\mathbf{x}$ si aquest s'interpreta com un natural en base 2.

El problema d'aquesta representació és que es poc flexible respecte l'interval que pot representar amb alta precisió. A la pràctica s'usa la codificació amb *coma flotant*, on la coma que separa les parts decimals i enteres no és fixa. Aquesta codificació té la forma:

Signe	**Mantissa**				**Exponent**			
s	m_3	m_2	m_1	m_0	e_3	e_2	e_1	e_0
1	0	1	0	1	0	0	1	1

$$(-1)^s \cdot 1.m_3m_2m_1m_0 \cdot 2^{\mathbf{e}_2}$$

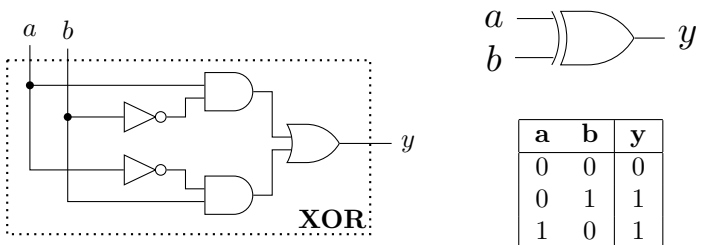
$$(-1) \cdot 1.0101 \cdot 2^3 = -1.3125 \cdot 2^3 = -10.5$$

On $1.m_3m_2m_1m_0$ és un decimal expressat amb coma fixa en base 2 i $\mathbf{e}_2$ és el valor del vector $\mathbf{e}$ si s'interpreta com un enter en Ca2. Aquesta codificació lliga la precisió al nombre de bits de la mantissa i l'interval de reals representables al nombre de bits de l'exponent.

B.3 Aritmètica sobre bits

Hem vist com representar informació sobre un nombre finit de bits, però el que cal és tenir la capacitat de realitzar modificacions i transformacions d'aquesta informació en la forma d'operacions aritmètiques.

En aquesta secció es presenten dissenys de portes compostes que realitzen una certa funció. Com a exemple, la figura 62 mostra el contingut de la porta XOR i el seu diagrama equivalent.



a	**b**	**y**
0	0	0
0	1	1
1	0	1
1	1	0

Figura 62: *Implementació interna, diagrama i taula de la veritat de la porta XOR.* **Font**: Elaboració pròpia mitjançant Tikz.

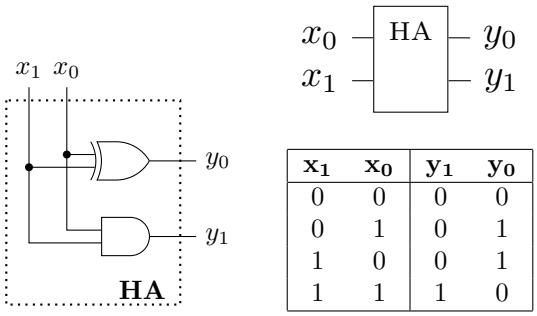
B.3.1 Suma i resta de naturals

Començarem dissenyant un circuit per a realitzar la suma de dos registres de dos bits. Aquest circuit s'anomena *Half-Adder* i es mostra a la figura 63.

Amb aquest bloc, podem implementar la suma de tres bits en l'anomenat *Full Adder*, detallat en la figura 64. Aquests dos blocs permeten comptar el nombre de bits d'entrada que es troben a 1, independentment del seu ordre.

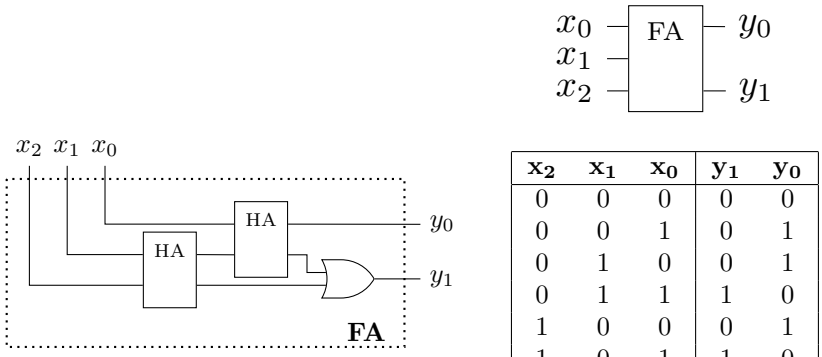
Amb Full Adders podem dissenyar un circuit per a realitzar la suma de dos naturals codificats en dos registres d'un nombre arbitrari de bits. El funcionament d'aquest bloc *ADD* es basa en sumar cada parell de bits dels registres originals amb elròssec del parell anterior, tal i com es mostra en la figura 65.

Es pot dissenyar una jerarquia de circuits similar que permeti calcular la resta de dos naturals de manera simple, però pel que fa a aquest treball podem assumir que generalment treballarem amb enters així que no en donarem els dissenys específics.



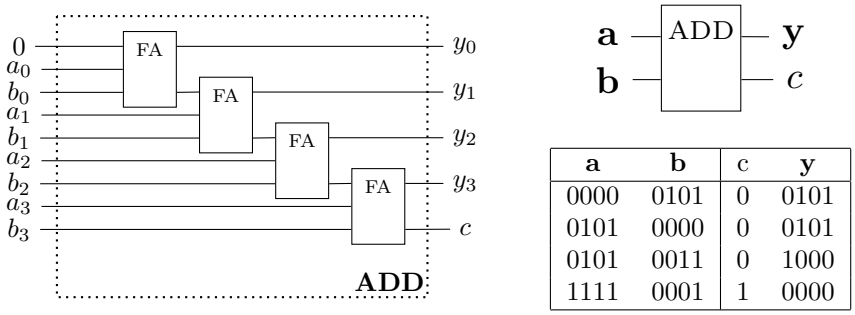
x_1	x_0	y_1	y_0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Figura 63: *Implementació interna, diagrama i taula de la veritat del bloc Half Adder.* **Font**: Elaboració pròpia mitjançant Tikz.



x_2	x_1	x_0	y_1	y_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Figura 64: *Implementació interna, diagrama i taula de la veritat del bloc Full Adder.* **Font**: Elaboració pròpia mitjançant Tikz.



a	**b**	c	**y**
0000	0101	0	0101
0101	0000	0	0101
0101	0011	0	1000
1111	0001	1	0000

Figura 65: *Implementació interna, diagrama i taula de la veritat del bloc ADD per a sumar dos naturals de 4 bits.* **Font**: Elaboració pròpia mitjançant Tikz.

B.3.2 Suma i resta d'enters

Es podria usar una modificació d'aquest bloc ADD per a realitzar la resta de naturals i la suma d'enters codificats amb Signe+Valor, però gràcies a la representació amb complement a 2 podem reutilitzar el mateix circuit per a realitzar la suma d'enters.

Aquest fet es pot justificar de manera més o menys formal pels diversos casos de la suma de dos enters $x + y$:

- $\mathbf{x >= 0 \wedge y >= 0}$: Els dos són enters positius, els quals són codificats igual que naturals, de manera que el circuit realitza la suma correctament.
- $\mathbf{x >= 0 \wedge y < 0}$: Per la relació $y+\neg y+1 = 2^n$, sabem que el circuit calcula: $x+(2^n-\neg y-1) = x - \neg y - 1 + 2^n = x - (\neg y + 1) + 2^n$, on 2^n és l'overflow del càlcul en n bits i $(\neg y + 1)$ és el valor absolut de y, de manera que el que ens queda al registre de sortida és efectivament el resultat correcte.
- $\mathbf{x < 0 \wedge y >= 0}$: Anàleg a l'anterior. Per la relació $x + \neg x + 1 = 2^n$, sabem que el circuit calcula: $(2^n - \neg x - 1) + y = (\neg x - 1 + 2^n) + y = y - (\neg x + 1) + 2^n$, on 2^n és l'overflow del càlcul en n bits i $(\neg x + 1)$ és el valor absolut de x, de manera que el que ens queda al registre de sortida és efectivament el resultat correcte.
- $\mathbf{x < 0 \wedge y < 0}$: Anàleg als anterior. Per la relació $x + \neg x + 1 = 2^n$, sabem que el circuit calcula: $(2^n - \neg x - 1) + (2^n - \neg x - 1) = 2^{n+1} - (\neg x + 1 + \neg y + 1)$, on 2^{m+1} és l'overflow del càlcul en n bits i $(\neg x + 1 + \neg y + 1)$ és la suma dels valors absoluts de x i y.

Per a realitzar la resta es pot aplicar una negació del segon operand i una suma de una unitat per a calcular el seu complement a 2. Per a simplificar els diagrames representarem aquesta composició amb el bloc *subtract*(SUB), tal i com es mostra en la figura 66.

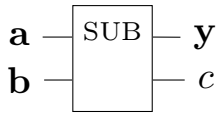


Figura 66: *Diagrama del bloc SUB per a restar dos enters. La sortida c correspon al "borrow"de la resta.* **Font**: Elaboració pròpia mitjançant Tikz.

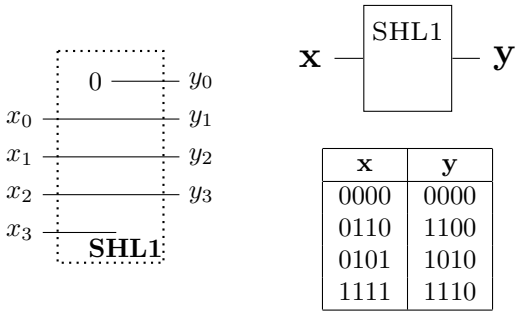
B.3.3 Producte i divisió per potències de 2

El producte i divisió per potències de 2 es realitza mitjançant el shiftat de bits. De la mateixa manera que en base 10 podem realitzar el producte o divisió de qualsevol nombre per 10^k movent el nombre k posicions a l'esquerra i afegint zeros a la dreta quan calgui, podem realitzar el producte d'un natural per 2 mitjançant el circuit Shift Left 1 (SHL1) mostrat a la figura 67.

La divisió es pot implementar amb un bloc oposat a aquest, el qual mogui els bits a la dreta, eliminant aquells que sobrin i afegint zeros a l'esquerra.

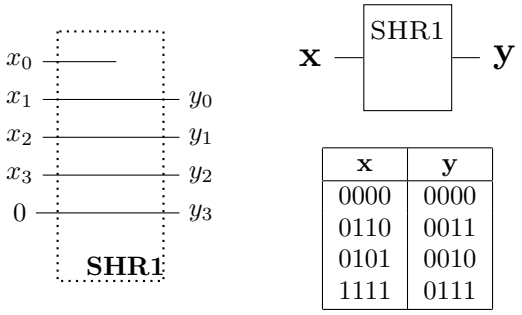
Aquests blocs es poden modificar per a operar amb enters si mantenim el bit de major pes constant, ja que és aquest el que defineix el signe. Aquestes modificacions reben el nom de Shiftats Aritmètics (SAL i SAR). El bloc SAL és idèntic al SHL ja que el bit de major pes no varia quan s'aplica el shiftat lògic a enters dins del rang corresponent. Per a calcular la divisió per 2 s'ha de realitzar una petita modificació al bloc SHR, mostrat a la figura **??**.

Aquests blocs per a multiplicar o dividir per 2 es poden encadenar de manera trivial per a obtenir circuits capaços de multiplicar o dividir qualsevol enter o natural per una potència de dos arbitrària. A partir d'ara anomenarem aquests circuits capaços de multiplicar o dividir per 2^k SHLk, SHRk o SARk, on k és també el nombre de blocs de shiftat individual encadenats.



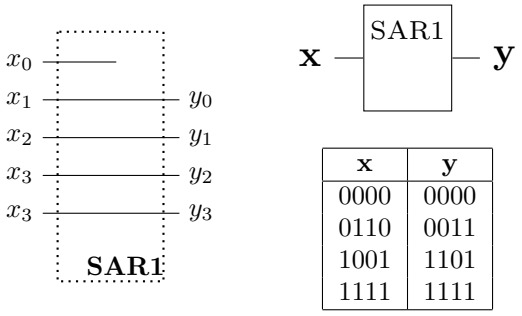
x	y
0000	0000
0110	1100
0101	1010
1111	1110

Figura 67: *Implementació interna, diagrama i taula de la veritat del bloc SHL1 de 4 bits.* **Font**: Elaboració pròpia mitjançant Tikz.



x	y
0000	0000
0110	0011
0101	0010
1111	0111

Figura 68: *Implementació interna, diagrama i taula de la veritat del bloc SHR1 de 4 bits.* **Font**: Elaboració pròpia mitjançant Tikz.



x	y
0000	0000
0110	0011
1001	1101
1111	1111

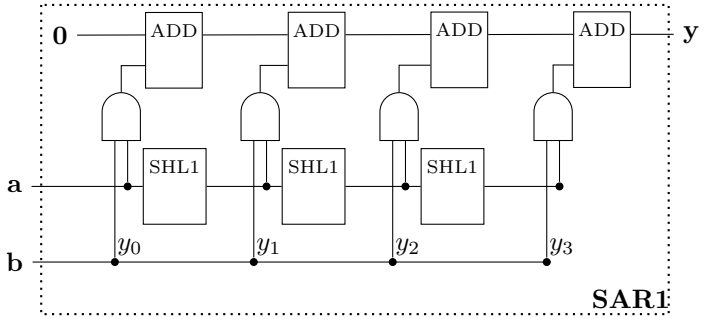
Figura 69: *Implementació interna, diagrama i taula de la veritat del bloc SAR1 de 4 bits.* **Font**: Elaboració pròpia mitjançant Tikz.

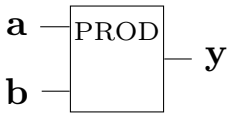
B.3.4 Producte de naturals i enters

Tal i com passa en base 10, podem descompondre el producte de dos naturals de la manera següent:

$$0110 \cdot 1010 = 0110 \cdot (1000 + 10) = 0110 \cdot 1000 + 0110 \cdot 10 = 110000 + 1100 = 111100$$

L'algorisme de multiplicació es basa en calcular tots els shiftats possibles del primer operand i realitzar una suma d'aquests condicionada als bits del segon. Un possible circuit que implementa aquest algorisme es mostra a la figura 70, el qual permet calcular el producte de dos naturals en registres de **a** i **b** de 4 bits i l'emmagatzema en un registre de 8 bits **y**.





a	b	y
0010	0001	00000010
0110	0011	00010010
1001	1101	01110101
1111	1111	11100001

Figura 70: *Implementació interna, diagrama i taula de la veritat del bloc PROD de dos registres de 4 bits. Els busos* **a** *i* **b** *són de 4 bits, els bus* **y** *és de 8, les portes AND s'apliquen a cada element de* **a** *i la seva sortida s'expandeix a un bus de 8 afegint 4 zeros per a realitzar la suma dels blocs ADD.* **Font**: Elaboració pròpia mitjançant Tikz.

Aquest circuit és capaç de realitzar el producte d'enters simplement afegint complementaris condicionals al signe dels registres a i b.

B.3.5 Divisió de naturals i enters

De la mateixa manera que amb la multiplicació, podem descompondre el quocient i el residu de la divisió:

$$\begin{aligned}\mathbf{a} &= \mathbf{b} \cdot q_3 q_2 q_1 q_0 + \mathbf{r} \\ &= \mathbf{b} \cdot (q_3 \cdot 1000 + q_2 \cdot 100 + q_1 \cdot 10 + q_0) + \mathbf{r} \\ &= q_3 \cdot \mathbf{b} \cdot 1000 + q_2 \cdot \mathbf{b} \cdot 100 + q_1 \cdot \mathbf{b} \cdot 10 + q_0 \mathbf{b} + \mathbf{r}\end{aligned}$$

El qual permet usar un disseny similar al del producte, on es realitza la resta successiva dels coeficients de cada q_i per a trobar el seu valor. El circuit que calcula aquest quocient i residu serà representat amb el diagrama mostrat a la figura 71.

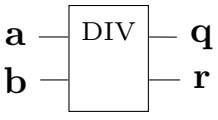


Figura 71: *Diagrama del bloc DIV per a dividir dos naturals o enters.* **Font**: Elaboració pròpia mitjançant Tikz.