NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

PROGRAM OF POSTGRADUATE STUDIES

PhD THESIS

Design and Synthesis of Efficient Circuits for Quantum Computers

Archimedes D. Pavlidis

ATHENS

DECEMBER 2016

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Σχεδίαση και Σύνθεση Αποδοτικών Κυκλωμάτων για Κβαντικούς Υπολογιστές

Αρχιμήδης Δ. Παυλίδης

ΑΘΗΝΑ

ΔΕΚΕΜΒΡΙΟΣ 2016

PhD THESIS

Design and Synthesis of Efficient
Circuits for Quantum Computers

Archimedes D. Pavlidis

SUPERVISOR: Dimitris Gizopoulos, Professor

THREE-MEMBER ADVISORY COMMITTEE:
Dimitris Gizopoulos, Professor
Antonis Paschalis, Professor
Dimitris Syvridis, Professor

SEVEN-MEMBER EXAMINATION COMMITTEE

(Signature) (Signature)

Dimitris Gizopoulos,
Professor NKUA

Antonis Paschalis,
Professor NKUA

(Signature) (Signature)

Dimitris Syvridis,
Professor NKUA

Angela Arapoyanni,
Professor NKUA

(Signature) (Signature)

Manolis Floratos,
Professor Emeritus NKUA

Kiamal Pekmestzi,
Professor NTUA

(Signature)

Dimitris Soudris,
Associate Professor NTUA

Examination Date 20/12/2016

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Σχεδίαση και Σύνθεση Αποδοτικών
Κυκλωμάτων για Κβαντικούς Υπολογιστές

Αρχιμήδης Δ. Παυλίδης

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Δημήτρης Γκιζόπουλος, Καθηγητής ΕΚΠΑ

ΤΡΙΜΕΛΗΣ ΕΠΙΤΡΟΠΗ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ:
Δημήτρης Γκιζόπουλος, Καθηγητής ΕΚΠΑ
Αντώνης Πασχάλης, Καθηγητής ΕΚΠΑ
Δημήτρης Συβρίδης, Καθηγητής ΕΚΠΑ

ΕΠΤΑΜΕΛΗΣ ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ

(Υπογραφή)

Δημήτρης Γκιζόπουλος,
Καθηγητής ΕΚΠΑ

(Υπογραφή)

Δημήτρης Συβρίδης,
Καθηγητής ΕΚΠΑ

(Υπογραφή)

Μανώλης Φλωράτος,
Ομότιμος Καθηγητής ΕΚΠΑ

(Υπογραφή)

Δημήτρης Σούντρης,
Αναπληρωτής Καθηγητής ΕΜΠ

(Υπογραφή)

Αντώνης Πασχάλης,
Καθηγητής ΕΚΠΑ

(Υπογραφή)

Αγγελική Αραπογιάννη,
Καθηγήτρια ΕΚΠΑ

(Υπογραφή)

Κιαμάλ Πεκμεστζή,
Καθηγητής ΕΜΠ

Ημερομηνία Εξέτασης 20/12/2016

ABSTRACT

The recent advances in the field of experimental construction of quantum computers with increased fidelity components shows that large-scale machines based on the principles of quantum physics are likely to be realized in the near future. As the size of the future quantum computers will be increased, efficient quantum circuits and design methods will gradually gain practical interest. The contribution of this thesis towards the design of efficient quantum circuits is two-fold. The first is the design of novel efficient quantum arithmetic circuits based on the Quantum Fourier Transform (QFT), like multiplier-with-constant-and-accumulator (MAC) and divider by constant, both of linear depth (or speed) with respect with the bits number of the integer operands. These circuits are effectively combined so as they can perform modular multiplication by constant in linear depth and space and consequently modular exponentiation in quadratic time and linear space. Modular exponentiation and modular multiplication operations are integral parts of the important quantum factorization algorithm of Shor and other quantum algorithms of the same family, known as Quantum Phase Estimation algorithms. Important implementation problems like the required high accuracy of the employed rotation quantum gates and the local communications between the gates are effectively addressed. The second contribution of this thesis is a generic hierarchical synthesis methodology for arbitrary complex and large quantum and reversible circuits. The methodology can handle more easily larger circuits relative to the flat synthesis methods. The proposed method offers advantages over the standard hierarchical synthesis which uses Bennett's method of "compute-copy-uncompute".

SUBJECT AREA: Quantum Computing

KEYWORDS: Quantum Computer Architectures, Quantum Arithmetic Circuits, Quantum Fourier Transform, Quantum Circuits Synthesis, Reversible Circuits Synthesis

ΠΕΡΙΛΗΨΗ

Οι πρόσφατες εξελίξεις στον τομέα της πειραματικής κατασκευής κβαντικών υπολογιστών με εξαρτήματα αυξημένης αξιοπιστίας δείχνει ότι η κατασκευή τέτοιων μεγάλων μηχανών βασισμένων στις αρχές της κβαντικής φυσικής είναι πιθανή στο κοντινό μέλλον. Καθώς το μέγεθος των μελλοντικών κβαντικών υπολογιστών θα αυξάνεται, η σχεδίαση αποδοτικότερων κβαντικών κυκλωμάτων και μεθόδων σχεδίασης θα αποκτήσει σταδιακά πρακτικό ενδιαφέρον. Η συνεισφορά της διατριβής στην κατεύθυνση της σχεδίασης αποδοτικών κβαντικών κυκλωμάτων είναι διττή: Η πρώτη είναι η σχεδίαση καινοτόμων αποδοτικών αριθμητικών κβαντικών κυκλωμάτων βασισμένων στον Κβαντικό Μετασχηματισμό Fourier (QFT), όπως πολλαπλασιαστής-με-σταθερά-συσσωρευτής (MAC) και διαιρέτης με σταθερά, με γραμμικό βάθος (ή ταχύτητα) ως προς τον αριθμό ψηφίων των ακεραίων. Αυτά τα κυκλώματα συνδυάζονται αποτελεσματικά ώστε να επιτελέσουν την πράξη του modulo πολλαπλασιασμού με σταθερά με γραμμική πολυπλοκότητα χρόνου και χώρου και συνεπώς μπορούν να επιτελέσουν την πράξη της modulo εκθετικοποίησης (modular exponentiation) με τετραγωνική πολυπλοκότητα χρόνου και γραμμική πολυπλοκότητα χώρου. Οι πράξεις της modulo εκθετικοποίησης και του modulo πολλαπλασιασμού είναι αναπόσπαστα μέρη του σημαντικού κβαντικού αλγορίθμου παραγοντοποίησης του Shor, αλλά και άλλων κβαντικών αλγορίθμων της ίδιας οικογένειας, γνωστών ως κβαντική εκτίμηση φάσης (Quantum Phase Estimation). Αντιμετωπίζονται με αποτελεσματικό τρόπο σημαντικά προβλήματα υλοποίησης, που σχετίζονται με την απαίτηση χρήσης κβαντικών πυλών περιστροφής υψηλής ακρίβειας, καθώς και της χρήσης τοπικών επικοινωνιών. Η δεύτερη συνεισφορά της διατριβής είναι μία γενική μεθοδολογία ιεραρχικής σύνθεσης κβαντικών και αντιστρέψιμων κυκλωμάτων αυθαίρετης πολυπλοκότητας και μεγέθους. Η ιεραρχική μέθοδος σύνθεσης χειρίζεται καλύτερα μεγάλα κυκλώματα σε σχέση με τις επίπεδες μεθόδους σύνθεσης. Η προτεινόμενη μέθοδος προσφέρει πλεονεκτήματα σε σχέση με τις συνήθεις ιεραρχικές συνθέσεις που χρησιμοποιούν την μέθοδο ”υπολογισμός-αντιγραφή-αντίστροφος υπολογισμός” του Bennett.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Κβαντική Υπολογιστική
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Αρχιτεκτονικές Κβαντικών Υπολογιστών, Κβαντικά Αριθμητικά Κυκλώματα, Κβαντικός Μετασχηματισμός Fourier, Σύνθεση Κβαντικών Κυκλωμάτων, Σύνθεση Αντιστρέψιμων Κυκλωμάτων

Στην μνήμη των γονιών μου
Δήμου και Ιφιγένειας,
στην Κατερίνα και στους
Δημοσθένη και Φίλιππο.

Ἐὰν ταῖς γλώσσαις τῶν ἀνθρώπων λαλῶ καὶ τῶν ἀγγέλων, ἀγάπην δὲ μὴ ἔχω, γέγονα χαλκὸς ἠχῶν ἢ κύμβαλον ἀλαλάζον. Καὶ ἐὰν ἔχω προφητείαν καὶ εἰδῶ τὰ μυστήρια πάντα καὶ πᾶσαν τὴν γνῶσιν, καὶ ἐὰν ἔχω πᾶσαν τὴν πίστιν, ὥστε ὄρη μεθιστάνειν, ἀγάπην δὲ μὴ ἔχω, οὐδέν εἰμι. καὶ ἐὰν ψωμίσω πάντα τὰ ὑπάρχοντά μου, καὶ ἐὰν παραδῶ τὸ σῶμά μου ἵνα καυθήσομαι, ἀγάπην δὲ μὴ ἔχω, οὐδὲν ὠφελοῦμαι.

ΕΥΧΑΡΙΣΤΙΕΣ

Τη σημερινή εποχή του μεγάλου ανταγωνισμού μεταξύ των ερευνητικών ομάδων για δημοσίευση αποτελεσμάτων αλλά και αναζήτησης χρηματοδοτούμενων ερευνητικών προγραμμάτων, απαιτείται η στόχευση σε εξειδικευμένες περιοχές όπου η ερευνητική ομάδα έχει αποκτήσει σημαντική εμπειρία αλλά και η δημιουργία κρίσιμης μάζας απασχολούμενων ερευνητών. Με αυτήν την έννοια η ερευνητική ενασχόληση με θέματα σχετικά αλλά όχι και στο κύριο ρεύμα ενδιαφερόντων της ομάδας θεωρείται πολλές φορές σπατάλη χρόνου.

Γι'αυτό το λόγο ευχαριστώ τον Καθηγητή του Τμήματος Πληροφορικής και Τηλεπικοινωνιών κο Δημήτρη Γκιζόπουλο για την εμπιστοσύνη που μου έδειξε και δέχθηκε να επιβλέψει την παρούσα διατριβή. Η εμπιστοσύνη αυτή συνεχίστηκε καθ'όλη τη διάρκεια της διατριβής όπου μου άφησε πολλούς βαθμούς ελευθερίας κινήσεων ως προς την κατεύθυνση της διατριβής, ακόμα και όταν υπήρχαν πρόσκαιρα προβλήματα σχετικά με την αποδοχή των δημοσιεύσεων. Η εμπειρία, η επιμονή και η αφιέρωση χρόνου από μέρους του ήταν κρίσιμα στοιχεία για την σωστή παρουσίαση των αποτελεσμάτων ώστε να γίνουν αποδεκτά από τους κριτές των περιοδικών και συνεδρίων όπου αποστάλθηκαν για δημοσίευση. Ο ενθουσιασμός του, συχνά μεγαλύτερος από τον δικό μου, έδινε θάρρος για την συνέχεια της προσπάθειας. Το σημαντικότερο όμως στοιχείο που συνέβαλε στην ολοκλήρωση της διατριβής ήταν η προσωπικότητά του, η οποία επέτρεψε να υπάρχει αγαστή συνεργασία μεταξύ μας και μετέτρεψε την διαδικασία εκπόνησης της διατριβής σε μία ευχάριστη ενασχόληση.

Ευχαριστίες οφείλονται και στον Καθηγητή του Τμήματος Φυσικής κο Μανώλη Φλωράτο. Η συνεργασία που ξεκίνησε μαζί του περίπου στο μέσο της εκπόνησης της διατριβής άνοιξε νέες κατευθύνσεις για την εκμετάλλευση των μεθοδολογιών που αναπτύχθηκαν και συνεισέφερε στην ψυχολογική ενθάρρυνση για την περάτωσή της.

Επίσης, ευχαριστώ τους Επίκουρους Καθηγητές του Πανεπιστημίου Πειραιώς Μιχάλη Ψαράκη και Γιώργο Πιτσέλη για την κριτική ανάγνωση τμημάτων της διατριβής.

Η διατριβή αυτή ολοκληρώθηκε χάρη στην υπομονή και ανοχή που έδειξε η οικογένειά μου, Κατερίνα, Δημοσθένης και Φίλιππος, όλα αυτά τα χρόνια και για αυτό θέλω να τους ευχαριστήσω, ιδιαίτερα όμως την Κατερίνα που επωμίστηκε το μεγαλύτερο βάρος.

PUBLICATIONS LIST

- A. Pavlidis and D. Gizopoulos, ”Fast Quantum Modular Exponentiation Architecture for Shor’s Factoring Algorithm,” *Quantum Information & Computation*, vol. 14, no 7&8, pp. 649-682, May 2014 (also in `https://arxiv.org/abs/1207.0511`)
- A. Pavlidis and D. Gizopoulos, ”Hierarchical Synthesis of Quantum and Reversible Architectures,” *Proc. 12th ACM International Conference on Computing Frontiers (CF’15)*, Ischia, Italy, May 2015, pp. 13:1-13:8 (*Best Paper Award Runner-up*)
- A. Pavlidis and D. Gizopoulos, ”Hierarchical Synthesis of Quantum and Reversible Architectures,” *International Journal of Parallel Programming*, vol. 44, no 5, pp. 1028-1053, October 2016 (*Special Issue; Extended version of the previous work*)

ΣΥΝΟΠΤΙΚΗ ΠΑΡΟΥΣΙΑΣΗ

Η Θεωρία Κβαντικής Πληροφορίας και η Κβαντική Υπολογιστική είναι διακλαδικά ερευνητικά πεδία που συνδυάζουν με διαφορετικά βάρη τις επιστήμες της Φυσική, της Πληροφορική και των Μαθηματικών, ανάλογα με την οπτική γωνία που θέλει κάποιος να τα προσεγγίσει. Η Κβαντική Υπολογιστική θεωρείται σχετικά πρόσφατο ερευνητικό πεδίο, αν και η Θεωρία Κβαντικής Πληροφορίας έχει ξεκινήσει να αναπτύσσεται εδώ και 40 χρόνια, μετά από σημαντικά αποτελέσματα σύνδεσης της κλασικής Θεωρίας Πληροφορίας με την Κβαντική Μηχανική (ανισότητες κβαντικών εντροπιών [1, 2, 3], φράγματα Holevo για χωρητικότητα καναλιών κβαντικής πληροφορίας [4, 5], φράγμα Bekenstein [6], κ.α.).

Η θεωρητική σύνδεση Κβαντικής Μηχανικής με τη Θεωρία Υπολογισμού έγινε τη δεκαετία του 1980 [7, 8], και επιπλέον ώθηση δόθηκε τη δεκαετία του 1990 με την επινόηση αποδοτικών Κβαντικών Αλγορίθμων [9, 10, 11], δηλαδή αλγορίθμων που εκτελούνται σε υπολογιστικές μηχανές (Κβαντικούς Υπολογιστές) που εκμεταλλεύονται βασικές κβαντικές ιδιότητες της φύσης όπως η υπέρθεση (superposition), και ο εναγκαλισμός ή διεμπλοκή (entanglement). Οι αποδοτικοί κβαντικοί αλγόριθμοι μπορούν να επιφέρουν σημαντική μείωση της χρονικής πολυπλοκότητας, ώστε σε αρκετές περιπτώσεις, προβλήματα που δεν επιδέχονται λύση σε πολυωνυμικό χρόνο με τους ως τώρα γνωστούς αλγορίθμους για κλασικούς υπολογιστές, να επιλύονται σε πολυωνυμικό χρόνο από έναν κβαντικό υπολογιστή που εκτελεί έναν κβαντικό αλγόριθμο. Ένα γνωστό παράδειγμα με σημαντικές εφαρμογές στην Κρυπτογραφία είναι η παραγοντοποίηση ενός σύνθετου ακεραίου στους πρώτους παράγοντές του (αλγόριθμος παραγοντοποίησης του Shor) [10]. Άλλα σημαντικά παραδείγματα είναι η αποδοτική προσομοίωση της συμπεριφοράς κβαντικών φυσικών συστημάτων με αρκετούς βαθμούς ελευθερίας (όπως ένα σύνθετό χημικό μόριο), υπολογισμός που είναι πρακτικά ανέφικτος με συμβατικούς υπολογιστές [12].

Η φυσική υλοποίηση ενός κβαντικού υπολογιστή, ενώ θεωρητικά είναι εφικτή, απαιτεί μία πολυσύνθετη τεχνολογική προσπάθεια που οφείλει να αντιμετωπίσει διάφορα πρακτικά προβλήματα. Ένα σημαντικό πρόβλημα είναι ότι οι φορείς της κβαντικής πληροφορίας, τα qubits (quantum bits), είναι πολύ ευαίσθητα στην επίδραση του περιβάλλοντος και είναι πολύ δύσκολο να διατηρηθούν σε μία δεδομένη κατάσταση για ένα αξιοποιήσιμο χρονικό διάστημα. Οι φυσικοί φορείς της πληροφορίας μπορεί να είναι φωτόνια, άτομα, ιόντα, πυρήνες και γενικά μικροσκοπικά θεμελιώδη συστήματα στα οποία μπορούν να παρατηρηθούν κβαντικά φαινόμενα [1]. Το πρόβλημα της ευαισθησίας των φορέων στην επίδραση του περιβάλλοντος είναι γνωστό ως αποσυμφωνία (decoherence) που ουσιαστικά μπορεί να θεωρηθεί σαν το αποτέλεσμα θορύβου του περιβάλλοντος. Το πρόβλημα της αποσυμφωνίας εντείνεται όσο αυξάνεται το πλήθος των qubits της μνήμης του κβαντικού υπολογιστή. Επίσης, οι βασικές μονάδες επεξεργασίας των qubits, γνωστές και ως κβαντικές πύλες (quantum gates), εισάγουν έναν επιπλέον παράγοντα αλλοίωσης της κβαντικής πληροφορίας διότι η λειτουργία τους πολλές φορές προσεγγίζει την ιδανική θεωρητική συμπεριφορά τους με σφάλμα που δεν επιτρέπει την κατασκευή κβαντικών υπολογιστών μεγάλου πλήθους qubits. Το σφάλμα αυτό μπορεί να θεωρηθεί επιπλέον θόρυβος που εισάγεται από το περιβάλλον και μετατρέπει τις ιδανικές πύλες σε «θορυβώδεις». Έτσι, παρ' όλο που από τα τέλη της δεκαετίας του 1990 μέχρι και σήμερα έχουν επιδειχθεί πραγματικοί κβαντικοί υπολογιστές χρησιμοποιώντας διάφορες τεχνολογίες (φωτόνια, παγίδες ιόντων, επαφές Josephson κ.α.), αυτοί περιορίζονται σε περίπου 10 qubits [15, 16, 17, 18].

Το πρόβλημα της αποσυμφωνίας έχει αντιμετωπιστεί θεωρητικά από την δεκαετία του 1990 εφαρμόζοντας και επεκτείνοντας ιδέες από την κλασική Θεωρία Κωδίκων Διόρθω-

[1] Τα πιο ελπιδοφόρα από αυτά είναι οι παγίδες ιόντων (ion traps) [13] και οι μικροσκοπικοί υπεραγωγοί Josephson [14]

σης Σφαλμάτων και αποτέλεσμα την επινόηση Κβαντικών Κωδίκων Διόρθωσης Σφαλμάτων [19, 20, 21]. Τέτοιοι κώδικες μπορούν να χρησιμοποιηθούν έτσι ώστε να συνδυαστούν πολλές «θορυβώδεις» φυσικές κβαντικές πύλες για να συνθέσουν μία ιδανική κβαντική λογική πύλη, δηλαδή να επιτρέψουν τη δημιουργία κβαντικών πυλών ανθεκτικών σε σφάλματα (fault tolerant gates). Αυτό μπορεί να γίνει κάτω από κάποιες προϋποθέσεις, σημαντικότερη από τις οποίες είναι ότι το ποσοστό του θορύβου που εισάγει η λειτουργία της κάθε φυσικής πύλης πρέπει να είναι κάτω από ένα κατώφλι (Κβαντικό Θεώρημα Κατωφλίου – Quantum Threshold Theorem) [22]. Σε μία τέτοια περίπτωση με τη χρήση πλεονασμού χώρου (spatial redundancy), δηλαδή με χρήση πολλών φυσικών κβαντικών πυλών, μπορεί να κατασκευαστεί μία ιδανική λογική κβαντική πύλη. Τα τελευταία χρόνια έχει ενταθεί η προσπάθεια για την κατασκευή κβαντικών πυλών υψηλής αξιοπιστίας, τέτοιας που να επιτρέψει την κατασκευή κβαντικών υπολογιστών ικανοποιητικού μεγέθους στο κοντινό μέλλον. Τα αποτελέσματα αυτών των προσπαθειών είναι πολύ ενθαρρυντικά.

Η διδακτορική συνεισφέρει σε δύο σημαντικά θέματα της Κβαντικής Υπολογιστικής:

- Σχεδίαση καινοτόμων και αποδοτικών κβαντικών κυκλωμάτων (συστοιχίες διασυνδεδεμένων κβαντικών λογικών πυλών), για βασικές αριθμητικές πράξεις ακεραίων καθώς και τον συνδυασμό τους σε υψηλότερο επίπεδο ιεραρχίας για την υλοποίηση κυκλωμάτων πιο σύνθετων αριθμητικών πράξεων. Η καινοτομία των κυκλωμάτων που προτάθηκαν έγκειται στην επεξεργασία των ακεραίων αφού έχει προηγηθεί κατάλληλος κβαντικός μετασχηματισμός Fourier (QFT), επιτυγχάνοντας έτσι βελτιωμένη απόδοση σε όρους ταχύτητας (ή ισοδύναμα βάθους του κυκλώματος). Προβλήματα που σχετίζονται με τη χρήση του QFT σε αριθμητικά κυκλώματα όπως η απαίτηση για πύλες υψηλής ακρίβειας και η έλλειψη τοπικότητας στις επικοινωνίες μεταξύ των qubits, επίσης αντιμετωπίζονται αποτελεσματικά με τις μεθόδους που προτάθηκαν στα πλαίσια της διατριβής.

- Πλήρης και αποδοτική μεθοδολογία ιεραρχικής σύνθεσης κβαντικών και αντιστρέψιμων αρχιτεκτονικών. Η πλειονότητα των μεθόδων αυτοματοποιημένης σύνθεσης είναι επίπεδες, δηλαδή λειτουργούν στο χαμηλότερο επίπεδο των λογικών πυλών και έχουν το μειονέκτημα ότι δεν είναι κατάλληλες για μεγάλα κυκλώματα αφού έχουν εκθετικές απαιτήσεις μνήμης και χρόνου εκτέλεσης της σύνθεσης. Η ενσωμάτωση ιεραρχικής σύνθεσης σε εργαλεία που χρησιμοποιούν επίπεδες μεθόδους σύνθεσης χρησιμοποιεί τη συνήθη μεθοδολογία του Bennett. Η μέθοδος του Bennett εφαρμόζει διαδοχικά κανονικούς υπολογισμούς, αντιγραφή των επιθυμητών αποτελεσμάτων σε ξεχωριστούς καταχωρητές και κατόπιν αντίστροφους υπολογισμούς που επαναφέρουν όλους τους καταχωρητές, εκτός από αυτούς της αντιγραφής, στις αρχικές τους καταστάσεις. Με αυτόν τον τρόπο τα τυχόν ενδιάμεσα αποτελέσματα έχουν μηδενιστεί με αντιστρέψιμο τρόπο. Η μέθοδος του Bennett επιφέρει επιβάρυνση στην χρησιμοποιούμενη μνήμη και την ταχύτητα του τελικού κυκλώματος. Η προτεινόμενη μέθοδος ιεραρχικής σύνθεσης κβαντικών και αντιστρέψιμων κυκλωμάτων προσφέρει πλεονεκτήματα σε όρους ταχύτητας και απαιτούμενης μνήμης του παραγόμενου κυκλώματος, έναντι αυτών της βιβλιογραφίας που χρησιμοποιούν τη μέθοδο Bennett. Σε κάποιες περιπτώσεις το πλεονέκτημα είναι περίπου μισή μνήμη και διπλάσια ταχύτητα.

Στο πλαίσιο της διατριβής, οι πύλες που χρησιμοποιούνται θεωρούνται αξιόπιστες (λογικό επίπεδο) που έχουν προκύψει από βασικές φυσικές πύλες με χρήση οποιασδήποτε τεχνικής διόρθωσης σφαλμάτων. Επομένως, η διατριβή αφορά το λογικό επίπεδο και όχι το

χαμηλότερο φυσικό επίπεδο της κατασκευής των βασικών κβαντικών πυλών σε μία συγκεκριμένη τεχνολογία. Συνεπώς οι μέθοδοι που προτείνονται στη διδακτορική διατριβή μπορούν να εφαρμοστούν σε οποιαδήποτε φυσική τεχνολογία κατασκευής.

Ως βασικό κριτήριο της απόδοσης των μεθόδων που προτάθηκαν στη διδακτορική διατριβή χρησιμοποιείται η ταχύτητα του υπολογισμού, γνωστή και ως βάθος του κυκλώματος (circuit depth), δηλαδή ο αριθμός των βημάτων που απαιτούνται για να ολοκληρωθεί ο κβαντικός υπολογισμός. Αυτό είναι ένα σημαντικό κριτήριο απόδοσης όταν μελλοντικά θα είναι εφικτή η κατασκευή κβαντικών υπολογιστών μεγάλου μεγέθους σε όρους μνήμης. Στην περίπτωση της προτεινόμενης μεθόδου ιεραρχικής σύνθεσης πλεονεκτήματα σε όρους μνήμης, εκτός της ταχύτητας, επίσης επιτυγχάνονται.

Τα προτεινόμενα κβαντικά υποσυστήματα αφορούν βασικές αριθμητικές πράξεις ακεραίων, όπως πολλαπλασιασμό σταθεράς με ακέραιο και συσσώρευση (multiply-and-accumulate – MAC) και διαίρεση ακεραίου με σταθερά (εύρεση πηλίκου και υπολοίπου) που χρησιμοποιούνται σε σημαντικούς κβαντικούς αλγορίθμους. Η υλοποίηση επιτυγχάνεται χρησιμοποιώντας εναλλακτικές αναπαραστάσεις των ακεραίων στο πεδίο Fourier (δηλαδή με χρήση του Κβαντικού μετασχηματισμού Fourier – QFT), αντί της συνήθους αναπαράστασης στην υπολογιστική βάση. Κβαντικά κυκλώματα που χρησιμοποιούν μετασχηματισμό Fourier είναι γνωστά στη βιβλιογραφία, αλλά περιορίζονται μόνο σε αθροιστές διαφόρων τύπων [23], ενώ η προφανής υλοποίηση ενός MAC με αναπαράσταση Fourier χρησιμοποιώντας τέτοιους αθροιστές [24] είχε τετραγωνικό βάθος ως προς το μέγεθος των ακεραίων. Αντίθετα, ο προτεινόμενος MAC έχει μόνο γραμμικό βάθος, ιδιαίτερα σημαντική ιδιότητα για μεγάλους (άρα και πρακτικά χρήσιμους) κβαντικούς αριθμούς. Ως προς τα κυκλώματα διαίρεσης, ελάχιστοι διαιρέτες εμφανίζονται στη βιβλιογραφία και κυρίως είναι περιορισμένοι για ειδικές περιπτώσεις (π.χ. για σώματα Galois $GF(2^m)$, δηλαδή διαιρέτες πολυωνύμων με συντελεστές 0 και 1). Ένας διαιρέτης βασισμένος σε μετασχηματισμό Fourier που είναι γνωστός [25], έχει κυβικό βάθος ενώ αν η διαίρεση γίνει με σταθερά τότε το βάθος ελαττώνεται σε τετραγωνικό. Ο προτεινόμενος στην διδακτορική διατριβή διαιρέτης με σταθερά έχει και πάλι μόνο γραμμικό βάθος.

Τα δύο παραπάνω κυκλώματα, κατάλληλα συνδυασμένα, μπορούν να χρησιμοποιηθούν για κατασκευή άλλων πιο σύνθετων κυκλωμάτων χρήσιμων σε διάφορους σημαντικούς κβαντικούς αλγορίθμους. Στην παρούσα διατριβή καταδεικνύεται ότι είναι δυνατή η κατασκευή πολλαπλασιαστή modulo N ο οποίος είναι βασικό συστατικό για την πράξη της ύψωσης σε δύναμη modulo N ή αλλιώς εκθετικοποίησης modulo N (modular exponentiation). Η ύψωση σε δύναμη είναι η πιο χρονοβόρα πράξη σε έναν από τους πιο σημαντικούς κβαντικούς αλγορίθμους, τον αλγόριθμο παραγοντοποίησης του Shor, αλλά και σε αλγορίθμους της ίδιας οικογένειας. Η προτεινόμενη σχεδίαση επιτυγχάνει βάθος $O(n^2)$ ενώ η πλειοψηφία των κυκλωμάτων της βιβλιογραφίας βρίσκεται στην περιοχή μεταξύ $O(n^2 \log n)$ και $O(n^3)$ και συνεπώς η προτεινόμενη σχεδίαση προσφέρει ιδιαίτερα μεγάλα πλεονεκτήματα ταχύτητας για μεγάλους κβαντικούς αριθμούς. Κάποια κυκλώματα τετραγωνικού βάθους ή μικρότερου έχουν το μειονέκτημα είτε της αύξησης της απαιτούμενης μνήμης κατά τάξη μεγέθους, είτε του υπολογισμού κατά προσέγγιση ενώ η σχεδίαση της διατριβής παρέχει ακριβή υπολογισμό.

Για την εκτίμηση της αποδοτικότητας ενός κυκλώματος (είτε ως προς το χρόνο είτε ως προς το χώρο) πρέπει να λαμβάνονται υπ' όψιν και τα χαρακτηριστικά της φυσικής υλοποίησής του στο χαμηλότερο επίπεδο. Ένα τέτοιο χαρακτηριστικό είναι η ύπαρξη δυνατότητας για καθολική αλληλεπίδραση μεταξύ των qubits που συμμετέχουν στον υπολογισμό ή αντίθετα ο περιορισμός της αλληλεπίδρασης σε γειτονικά qubits μόνο, για παράδειγμα σε υλοποίηση μονοδιάστατης γραμμικής συστοιχίας qubits όπου το κάθε ένα μπορεί να αλλη-

λεπιδράσει μόνο με τα δύο γειτονικά του (1D-LNN,1D Linear Nearest Neighbourhood). Η προτεινόμενη αρχιτεκτονική για τον αλγόριθμο του Shor, παρ' όλο που εκ πρώτης όψεως φαίνεται να απαιτεί καθολικές επικοινωνίες μεταξύ των qubits, αποδεικνύεται ότι μπορεί να προσαρμοστεί σε φυσικές μηχανές τοπικής αλληλεπίδρασης με σταθερή επιβάρυνση στο βάθος, δηλαδή δεν υπάρχει αύξηση του αρχικά υπολογισμένου τετραγωνικού βάθους. Αντίθετα, οι περισσότερες αρχιτεκτονικές χαμηλού βάθους $O(n^2 \log n)$ όταν περιοριστούν σε μηχανή που απαιτεί γειτονικές επικοινωνίες αυξάνουν το βάθος (π.χ. $O(n^3)$ σε 1D-LNN ή $O(n^2\sqrt{n})$ σε 2D-LNN) [26].

Η επεξεργασία στο πεδίο Fourier που εφαρμόζουν τα προτεινόμενα κυκλώματα έχουν σαν αποτέλεσμα τη χρήση ελεγχόμενων κβαντικών πυλών περιστροφής (controlled rotation quantum gates) με συγκεκριμένες γωνίες. Ένα γνωστό μειονέκτημα τέτοιων πυλών είναι ότι δεν ανήκουν στην κατηγορία των πυλών που μπορούν να γίνουν ανθεκτικές στα σφάλματα, εκτός και αν αποδομηθούν σε ακολουθία πυλών που είναι ανθεκτικές σε σφάλματα (π.χ. σε πύλες H, T). Μία τέτοια αποδόμηση όμως θα συνεπαγόταν σημαντική επιβάρυνση στο βάθος του συνολικού κυκλώματος εκθετικοποίησης κατά μία τάξη μεγέθους, δηλαδή από $O(n^2)$ σε $O(n^3)$. Μπορεί όμως, όπως δείχνει η διατριβή, η επιβάρυνση να γίνει πολύ μικρότερη με ένα τελικό βάθος $O(n^2 \log n)$, με τη διαφορά όμως ότι οι υπολογισμοί θα γίνονται προσεγγιστικά, αλλά επιτρέποντας στην εφαρμογή (αλγόριθμος του Shor) να λειτουργήσει με ελάχιστη υποβάθμιση ως προς την πιθανότητα επιτυχίας. Έτσι, η προτεινόμενη αρχιτεκτονική είναι μία από τις πιο ανταγωνιστικές ως προς το βάθος, ιδιαίτερα αν εφαρμοστεί σε 1D-LNN ή 2D-LNN φυσικές μηχανές, που είναι και πιο πιθανό να υλοποιηθούν στο μέλλον.

Η σχεδίαση κβαντικών κυκλωμάτων υιοθετεί ιδέες από τη σχεδίαση κλασικών λογικών κυκλωμάτων που μπορούν να συνδυαστούν μεταξύ τους. Μικρά κυκλώματα ή κυκλώματα που έχουν επαναληπτικές μικρές δομές μπορούν να σχεδιαστούν είτε κατά περίπτωση (ad hoc) είτε με συστηματικές μεθόδους σύνθεσης με βάση τις προδιαγραφές λειτουργίας (π.χ. πίνακες αλήθειας). Στην περίπτωση των κβαντικών κυκλωμάτων υπάρχουν ανάλογες μέθοδοι κατασκευής με βάση προδιαγραφές που στη γενική περίπτωση είναι τετραγωνικοί μοναδιαίοι πίνακες [27, 28]. Σε ειδικές περιπτώσεις όπου ένα κβαντικό κύκλωμα περιγράφεται από πίνακα με στοιχεία αποκλειστικά 0 και 1, μπορούν να χρησιμοποιηθούν μέθοδοι σύνθεσης αντιστρέψιμων[2] (reversible) λογικών κυκλωμάτων [30]. Τέτοιες περιπτώσεις κβαντικών κυκλωμάτων συναντώνται όταν το κύκλωμα υπολογίζει μία αριθμητική ή λογική συνάρτηση στην υπολογιστική βάση (π.χ. πρόσθεση).

Σε όλες αυτές τις περιπτώσεις οι μεθοδολογίες αυτές είναι κατάλληλες για μικρά κυκλώματα μόνο, γιατί η υπολογιστική ισχύς και η μνήμη που απαιτείται για την εφαρμογή τους αυξάνεται εκθετικά με το μέγεθος του προβλήματος. Η προφανής λύση είναι η ιεραρχική σχεδίαση που εφαρμόζεται και σε κλασικά κυκλώματα. Στην ιεραρχική σχεδίαση, εφ' όσον η επιθυμητή λειτουργία μπορεί να περιγραφεί σαν συναρμογή απλούστερων λειτουργιών, η σχεδίαση ξεκινάει από το χαμηλότερο επίπεδο και προχωράει προς το υψηλότερο. Η εφαρμογή μίας τέτοιας μεθόδου σε κβαντικά κυκλώματα είναι δυνατή αλλά απαιτεί ιδιαίτερη μεταχείριση των ενδιάμεσων αποτελεσμάτων του υπολογισμού τα οποία δε χρειάζονται στο τέλος. Η ιδιαιτερότητα οφείλεται στο γεγονός του ότι τα ενδιάμεσα αποτελέσματα δεν μπορούν απλώς να αγνοηθούν στο τέλος διότι γενικά βρίσκονται σε κβαντική διεμπλοκή με τα επιθυμητά αποτελέσματα. Αυτό που πρέπει να γίνει είναι να επανέλθουν στην αρχική τους κατάσταση μέσω αντίστροφου υπολογισμού. Η μέθοδος του Bennett [31] είναι μία γνωστή μέθοδος τέτοιου αντίστροφου υπολογισμού που διατηρεί τα επιθυμητά αποτελέσματα μέσω αντιγραφής. Το κύριο χαρακτηριστικό της είναι ότι διπλασιάζει

[2]Στα αντιστρέψιμα λογικά κυκλώματα από κάθε πιθανή έξοδο είναι δυνατή η εξαγωγή της αντίστοιχης εισόδου, δηλαδή δεν υπάρχει απώλεια πληροφορίας [29].

τα βήματα υπολογισμού (ευθύς υπολογισμός, αντίστροφος υπολογισμός) και απαιτεί επιπλέον μνήμη, όση απαιτούν τα τελικά αποτελέσματα, λόγω της αντιγραφής.

Η μέθοδος ιεραρχικής σχεδίασης που προτάθηκε στη διδακτορική διατριβή προσφέρει πλεονεκτήματα σε σχέση με τη μέθοδο Bennett σε όρους τόσο ταχύτητας όσο και μνήμης του τελικού κυκλώματος. Οι προδιαγραφές του κυκλώματος που πρόκειται να συντεθεί παρέχονται σαν μία ακολουθία συναρτήσεων (αριθμητικών ή λογικών). Οι συναρτήσεις αποτελούν μέρος μία βιβλιοθήκης κβαντικών υπο-κυκλωμάτων. Η βιβλιοθήκη αυτή μπορεί να δημιουργηθεί με χρήση άλλων μεθόδων σύνθεσης χαμηλότερου επίπεδου (πύλης), να περιέχει γνωστά από την βιβλιογραφία παραμετροποιημένα ως προς το μέγεθος κυκλώματα (π.χ. αθροιστές) ή να εμπλουτιστεί με νέα κυκλώματα από την ίδια ιεραρχική μέθοδο. Επίσης, η βιβλιοθήκη περιέχει και τα αντίστροφα κυκλώματα για τις ανάγκες που περιγράφηκαν νωρίτερα. Το τελικό αποτέλεσμα της σύνθεσης σε μορφή κατευθυντικού ακυκλικού γράφου περιγράφει το επιθυμητό κύκλωμα, όπου οι κόμβοι του γράφου αναπαριστούν τα έτοιμα υποκυκλώματα της βιβλιοθήκης και οι ακμές περιγράφουν τις διασυνδέσεις μεταξύ των υποκυκλωμάτων. Η μέθοδος σύνθεσης απαιτεί πολυωνυμικό χρόνο και μνήμη σε σχέση με τον αριθμό των συναρτήσεων που χρησιμοποιούνται και σε κάθε περίπτωση παράγει κυκλώματα καλύτερης ή ίδιας απόδοσης βάθους και μνήμης σε σχέση με τη βασική μέθοδο του Bennett.

Η δομή της διδακτορικής διατριβής είναι η ακόλουθη: Το Κεφάλαιο 2 είναι μία εισαγωγή στην περιοχή της Κβαντικής Υπολογιστικής ενώ το Κεφάλαιο 3 συνοδευόμενο από το Παράρτημα Α είναι η αναλυτική περιγραφή του αλγορίθμου του Shor. Η συνεισφορά της διατριβής ξεκινάει στο Κεφάλαιο 4 όπου περιγράφονται τα γρήγορα κβαντικά αριθμητικά κυκλώματα που βασίζονται σε αναπαράσταση ακεραίων με χρήση του κβαντικού μετασχηματισμού Fourier. Το Κεφάλαιο 5 περιγράφει τη μεθοδολογία προσέγγισης αυτών των κυκλωμάτων με κβάντιση γωνίας καθώς και την προσαρμογή τους σε φυσική μηχανή μονοδιάστατης διάταξης γειτονικών επικοινωνιών. Το Κεφάλαιο 6 περιγράφει την μεθοδολογία ιεραρχικής σχεδίασης κβαντικών κυκλωμάτων. Τέλος, το Κεφάλαιο 7 είναι μία ανασκόπηση συμπερασμάτων και πιθανές μελλοντικές κατευθύνσεις έρευνας στα συγκεκριμένα θέματα της διατριβής.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

This thesis was conducted in the Department of Informatics and Telecommunications of the National and Kapodistrian University of Athens during the period 2011-2016.

1. EXTENDED SUMMARY

Quantum Information Theory and Quantum Computing are interdisciplinary research fields that combine different doses of Physics, Informatics and Mathematics depending on which aspect someone focuses. Quantum Computing is a relatively recent research field, although Quantum Information Theory has already been developed for the last 40 years, after important results which connect classical Information Theory to Quantum Mechanics (quantum entropies inequalities [1, 2, 3], Holevo bounds for capacities of quantum channels [4, 5], Bekenstein bound [6], etc.)

The theoretical connection of Quantum Mechanics to the Theory of Computation achieved in the 80's [7, 8], while more boost came in the 90's with the invention of efficient quantum algorithms [9, 10, 11], which can be executed on computing machines (quantum computers) exploiting fundamental quantum properties of nature, like superposition and entanglement. Such efficient algorithms can achieve important reduction of time complexity, so that in many instances, problems that cannot be solved in polynomial time on a classical computer with the currently known algorithms, can be solved in polynomial time on a quantum computer. A famous example, with important applications in Cryptography, is the factorization of a composite integer into its prime factors (Shor's algorithm)[10]. Another important example is the efficient simulation of quantum physical systems with many degrees of freedom (like a complex chemical molecule), a computation which is not practically achievable in a classical computer [12].

The physical realization of a quantum computer, while in principle is feasible, requires a complex technological effort to overcome practical problems. An important problem is that the carriers of quantum information, the qubits, are very fragile under the influence of their environment and it is very difficult to maintain them in a constant state for a long enough duration so as they can perform a useful computation. The physical carriers of information can be atoms, ions, nuclei and in general any microscopic system on which quantum mechanical effects can be observed[1]. The disturbance effect on the qubits under the environment influence is known as decoherence and can be thought as an environment noise effect. Decoherence problems increase as the number of qubits increases. Additionally, the basic processing elements of qubits, the quantum gates, introduce another factor of disturbance of quantum information, because usually their operation approximates the ideal theoretical operation with errors which don't allow the construction of useful large quantum computers. These introduced errors can be thought as an additional environment induced noise, converting the ideal gates to noisy or erroneous ones. Thus, although real quantum computers have been already developed using various technologies (photons, ion traps, Josephson junctions), they are limited to about 10 qubits [15, 16, 17, 18].

The decoherence problem has been theoretically addressed in the 90's by exploiting and extending results from classical Error Correcting Codes Theory, leading to the invention of Quantum Error Correcting Codes [19, 20, 21]. Such codes can be applied by combining many noisy quantum physical gates so as to build an ideal quantum logical gate, that is they allow the construction of fault tolerant quantum gates. This can be accomplished under some conditions, of which the most important is that the noise percentage introduced by each physical quantum gate is lower than a threshold (Quantum Threshold Theorem) [22]. In such a case, an ideal quantum logical gate can be constructed by using redundancy, that is using many physical gates. During the recent years, the effort to build high reliability quantum gates has been intensified, so as to permit the construction of quantum computers of adequate size in the near future. Results of these efforts are very encourag-

[1] Currently, some of the most promising are ion traps [13] and Josepshon junction superconductors [14]

ing.

This thesis contributes two-fold:

- Design of novel efficient quantum circuits (arrays of interconnected quantum logical gates) for integer arithmetic operations and their combination to a higher hierarchy level to achieve more complex arithmetic operations, like modular exponentiation which is an integral part of Shor's algorithm and important algorithms of the same class. The novelty of the proposed circuits lays in the usage of Quantum Fourier Transform (QFT) on the integers states prior to their processing, resulting in improved efficiency in terms of speed. Problems related to the usage of QFT in arithmetic circuits, such as the requirement for high precision quantum gates and the lack of communications locality between the qubits, are also effectively addressed.
- A generic hierarchical quantum and reversible circuits synthesis methodology. The majority of existing automatic synthesis methods are flat; they operate on the lowest level of gates and while in many cases they lead to optimal or suboptimal results, they have the disadvantage of not being suitable for large circuits as they have exponential requirements in memory usage and run time. The straightforward incorporation of hierarchical synthesis methods into tools of flat methods uses the methodology of Bennett. In contrast, the proposed hierarchical method offers advantages in terms of derived circuit speed and memory, relative to the few hierarchical ones of the literature.

In the context of this thesis, the used gates are assumed to be reliable (logical level) which have been derived from elementary physical quantum gates incorporating any method of error correction. Thus, the thesis concerns the logical level of quantum gates and not the lower level of physical gates. Therefore, the proposed methods of this doctoral thesis can be applied to any technology of physical realization and fault tolerant implementation of logic gates.

We adopt the computation speed, which is known as circuit depth, as the main criterion of efficiency of the proposed methods in this thesis, and it is the number of required steps to complete the computation. This is an important efficiency criterion when construction of large size, in terms of memory, quantum computers become feasible in the future. In the case of the proposed synthesis method, advantage in terms of memory (qubits requirement), except of speed, are also achieved.

The proposed quantum subsystems concern basic arithmetic operations on integers, like multiplication of a constant with an integer and accumulation (MAC) and division by constant (quotient and remainder calculation) which are used in important quantum algorithms. The implementations is accomplished by using alternative representation of integers in the Fourier domain (that is we use the Quantum Fourier Transform) instead of the usual representation in the computational basis. Quantum circuits using QFT exist in the literature, but they are limited to various kind of adders only [23], while the straightforward implementation of a MAC with Fourier representation using such adders [24] has quadratic circuit depth relative to the integer size. In contrast, the proposed MAC offers linear depth, a considerably important property for large (and thus practically useful) quantum numbers. Regarding the division circuits, just a few quantum dividers exist in the literature and they are chiefly limited to special purposes (e.g. for Galois fields $GF(2^m)$, that is dividers of polynomials with coefficients 0 and 1). A known general quantum divider based on QFT [25] has a cubic depth, while if the divisor is constant its depth can be reduced to be quadratic. The proposed constant divider in this thesis offers a linear depth.

The above two circuits, effectively combined, can be used to construct other more complex circuits useful in various important quantum algorithms. In this thesis we show how it is possible to construct a multiplier modulo N, which is a fundamental element for the operation of modular exponentiation. Modular exponentiation is the most time consuming operation in one of the most important quantum algorithms, the factorization algorithm of Shor, and also in other algorithms of the same family. The proposed design achieves a circuit depth of $O(n^2)$, while the majority of the circuits in the literature ranges between $O(n^2 \log n)$ and $O(n^3)$, and consequently the proposed design offers important speed advantage for large numbers. Some of the circuits in the literature offering quadratic or less depth have the disadvantage of increasing excessively the required space (number of qubits) in order or they have the disadvantage of performing approximate calculation.

In the estimation of the circuit efficiency (being in time or space) we must take into account the physical implementation constraints. Such a constraint is the capability of global interactions between the qubits or the limitation of this interaction to neighborhood qubits only, e.g. in a linear one-dimensional array implementation of qubits, where each one can interact only with its two neighbors (1D-LNN, 1D-Linear Nearest Neighborhood). The proposed architecture for Shor's algorithm, while at first sight seems to require global communications between the qubits, it can be adapted in physical machines requiring local interactions with constant overhead in depth, as we show. That is, we don't have any increase in the quadratic order of depth. In contrast, most of the low $O(n^2 \log n)$ depth architectures when applied in a machine that requires local communications increase the depth (e.g. to $O(n^2\sqrt{n})$ in 2D-LNN or to $O(n^3)$ in 1D-LNN) [26].

The Fourier domain processing of the proposed circuits requires the usage of controlled rotation quantum gates with specific angles. A known drawback of such gates is that they do not belong to the category of gates that may constructed fault tolerantly, unless they are decomposed in a sequence of fault tolerant capable gates (e.g. H and T gates). But, such a decomposition implies considerable overhead in the depth of the whole modular exponentiation circuit up to an order, that is to $O(n^3)$ from $O(n^2)$. Yet, it is possible, as it will be shown, to have a much lesser overhead of $O(n^2 \log n)$ by permitting approximate computation which allow the Shor's algorithm to operate with minor degradation concerning the probability of success. Therefore, the proposed architecture is one of the most competitive in terms of depth, especially if it is applied to 1D-LNN or 2D-LNN physical machines, which are the most probable to be implemented in the future.

Design of quantum circuits adopts ideas from classical logical design. Small circuits or circuits with repetitive structure can be designed either ad hoc or with formal synthesis methods based on specifications (e.g. truth tables). In the case of quantum circuits there exist similar synthesis methods based on specifications which in the general case are unitary matrices [27, 28]. In special cases where a quantum circuit is described by a matrix with elements exclusively 0 and 1, then reversible circuits[2] synthesis methods can be exploited [30]. Such quantum circuits cases are met when the circuit computes an arithmetic or logical function in the computational basis (e.g. integer addition).

In such cases, these methodologies are suitable for small circuits only, because the required computation power and memory required for their application increases exponentially with the circuit size. The obvious solution is the hierarchical bottom-up design which is applied in classical circuits. In the hierarchical method, if the desired operation can be described as a splicing of simpler operations, the design starts from the lowest level of

[2] In a reversible circuit, for every possible output, the respective input can be derived, that is no information erasure happens [29].

simpler operations towards the higher level of the more complex operations. The application of the hierarchical method to quantum circuits is possible but requires special handling of the intermediate computation results that are not useful at the end. The particularity is caused due to the fact that these intermediate results cannot be simply discarded at the end because, in general, they are quantum entangled with the desired results. They must be reset to their initial state by inverse computation. Bennett's method is a well known method that keeps the desired results through copying and resets the intermediate results through uncomputation [31]. Its main characteristic and drawback is that it doubles the computation steps (forward computation and the reverse computation) and it also requires more memory space, equal to the space needed by the desired results due to the copying.

The hierarchical design method we propose in the doctoral thesis offers advantages relative to Bennett's method in terms of speed and memory of the target circuit. The specifications of the synthesizable circuit are given as a sequence of arithmetic or logic functions. These functions are supposed to be part of a library of quantum circuits. The library can be constructed by using other lower level synthesis methods, or contain known parametrized circuits of the literature (e.g. adders) or be populated with new circuits of the same hierarchical method. Also, the library contains the inverse circuits due to the necessity described above. The end result of the synthesis in the form of directed acyclic graph describes the target circuit, where the nodes of the graph represent the modules of the library and the arcs of the graph represent the interconnections between the modules. This synthesis method requires polynomial execution time and memory space in relation to the number of the functions of the specifications and in any case it produces circuits of equal or better performance in terms of depth and space in compared to the basic Bennett's method.

The structure of the doctoral thesis is as follows: Chapter 2 is an introduction to Quantum Computing, while Chapter 3 accompanied with Appendix A is a detailed description of Shor's algorithm. The contribution of the thesis begins in Chapter 4 where the fast quantum arithmetic circuits based on the Fourier representation of integers is presented. Chapter 5 describes the approximation methodology of these circuits with angle quantization and their adaptation on a physical machine of one-dimensional linear array of nearest neighborhood interactions. Chapter 6 describes the hierarchical synthesis method of quantum and reversible circuits. Finally, Chapter 7 is a review of conclusions and potential future research directions in the subjects of the thesis.

2. INTRODUCTION TO QUANTUM COMPUTING

An impressive progress in every field of computer science has been observed since the invention of the first digital computers in the 40's: hardware, algorithm design, software applications and networking. Moore's law, introduced in 1965 [32], predicted a doubling in the transistor density every two years with nearly equivalent consequences in computation power, energy decrease per computation step, etc. Moore's law has remained invariant until today, but various technological problems, such as power consumption, process variation, electromigration, etc., raise doubts if such a rate will continue. E.g. a power consumption related law known as Dennard's law [33] stated that, similarly to Moore's law, the computation power per energy unit would raise exponentially as time passed. But this law is no longer valid from about 2006, mainly due to current leakages. Apart from the various technological problems, there are fundamental limits that prohibit the ongoing miniaturization as quantum mechanical effects will come to forefront. E.g. the silicon dioxide gate insulator used in MOS transistor has a thickness of about 5 silicon atoms and the quantum mechanical effect of electron tunneling leads to current leakage towards the channel of the transistor raising the power consumption.

Besides hardware limitations, there are fundamentals software limitations, because there exist problems that classical algorithms cannot solve efficiently (in polynomial time with respect to the problem size). Quantum computing is a fundamentally different approach, both in hardware and algorithm design, which can exploit quantum mechanical effects to solve efficiently problems that are hard to be solved using classical computers.

2.1 Classical Computing

The formal notion of algorithm can be described by the definition of the *Turing machine* (TM) introduced in 1936 and the *Universal Turing Machine* (UTM) which is a general TM that can simulate any TM [34]. Essentially, UTM corresponds to a programmable computer for which any given problem is solved by a different program. Any problem that can be solved by an existing algorithm can be also treated by a Turing machine designed to solve the particular problem (or a UTM that has been programmed to solve this problem). Conversely, for any given problem which can't be solved by an existing TM, no algorithm exists . This is known as *Church-Turing thesis*. A problem that can be solved by a TM is said to be *computable* or *decidable*, otherwise it is said to be *non-computable* or *undecidable*. The Halting problem[1] is a well known example of an undecidable problem [34]. Thus, the Church-Turing thesis partitions the various problems in the two broad classes above.

2.1.1 Complexity Classes

Although TM is an abstract mathematical/logical model of computation, it contains some inevitable physical assumptions such as space, time and power requirements for its operation. Space requirements are related to the tape cells[2] consumption, while time requirements are related to the finite transition time in each step. Power requirements are related to the possible information erasure when writing a cell. Computational complexity deals with partitioning problems into various classes depending on these requirements [35, 34].

Time and space requirements are defined with respect to the input size n of the problem to be solved. A broad class consists of problems that are *efficient* or *tractable* to solve on a TM. These problems need at most polynomial time in n and the set of all these problems

[1] The Halting problem is the problem to decide whether a given algorithm fed with a particular input will terminate or not.

[2] Tape cells in a TM are analogous to the memory in a conventional classical computer

define the P complexity class. E.g., the greatest common divisor of two n-digit integers can be found by the Euclid algorithm in time which is proportional to n^2.

A technique to solve some problems outside the P class, in polynomial time, is to introduce randomness in an algorithm. In the TM model this is accomplished by extending the model so that it becomes indeterministic and the transition function is dependent on a probability distribution function (probabilistic TM). The *Boundary Probability Polynomial* class (BPP) is the class of problems that are solved in polynomial time on such a probabilistic TM. Thus, $P \subseteq BPP$, although it hasn't been proven that the inclusion is not strict. The introduction of the BPP class led to a new definition of the Church-Turing thesis [36].

Definition 2.1. *(Strong Church-Turing thesis) Any reasonable model of computation is efficiently simulated by a probabilistic TM.*

Thus, any problem in the P or in the BPP class is efficiently solvable in any reasonable model of computation as this model is asserted to be polynomially simulated by a probabilistic TM.

Problems for which no efficient algorithm is known to exist are said to be *hard* or *inefficient*. The *Non-deterministic Polynomial* complexity class, or NP, falls in the category of hard problems. The NP problems are defined as the ones that have polynomial time *verifiability*, meaning that even if there is no known polynomial time algorithm to solve them, yet a solution can be verified in polynomial time. An example is the factorization problem of an integer into its prime factors. The best known factorization algorithm is the General Number Field Sieve, which has complexity $O(exp(\sqrt[3]{\frac{64}{9}n})exp(\log n)^{\frac{2}{3}})$, where n is the bits length of the integer. However, a possible factor can be verified in polynomial time using the division algorithm.

An equivalent definition of an NP problem is that it can be solved in polynomial time by a *non-deterministic Turing Machine* (NTM), justifying the name NP (Non-deterministic Polynomial). A NTM (not to be confused with the probabilistic TM) is a modified TM in which each transition may lead to different configurations of the machine by replicating the machine itself. A NTM is believed not to be physically realizable model of computation. It can be proven that any TM cab be simulated on a NTM in polynomial time, thus $P \subseteq NP$, though the relation between NP and BPP is not known.

A subclass of NP problems are the $NP - complete$ problems. These are the "hardest" problems of the NP class in the sense that the solution of any NP problem can be reduced in polynomial time to the solution of an $NP - complete$ problem. Consequently, if any of the $NP - complete$ problems is proven to be in P then every NP would be in P, resulting in $NP = P$.

Concerning the space complexity, an important class is the $PSPACE$. This class consists of problems that can be solved using polynomial space (computer memory or TM cells) with respect to the size of the problem. It can be proven that $P \subseteq PSPACE$ and $NP \subseteq PSPACE$. Thus, the following ordering of classes is known to be valid, without any proof if any inclusion is strict or not.

$$P \subseteq NP \subseteq PSPACE \tag{2.1}$$

In Section 2.4 another complexity class relevant to the quantum computing will be associated with the above defined classes.

The abstract model of TM is usually realized on computers as a digital logic sequential circuit. Moreover, it is proven that a TM is equivalent to combinational digital logic circuits (circuits that have no memory element and feedback) with polynomial cost in space and time [37, 38, 39]. Thus, the efficiency of a problem is preserved if it is solved on a combinational circuit. A similar equivalence holds for the quantum computation case between the Quantum Turing Machine model (subsection 2.2.6) and the Quantum Circuit model (section 2.3) of computation.

2.1.2 Irreversible and Reversible Computing

The energy resources required by a computation are directly related to the information loss that takes place during the computation. Landauer [40] showed that there is a fundamental lower limit of energy loss (*Landauer limit*) when information processing is done in an *irreversible* manner. For each bit of information loss at least $kT\ln 2$ Joules of energy are dissipated, where T is the environment absolute temperature in Kelvin and k is the Boltzmann constant ($\approx 1.38 \cdot 10^{-23}$ Joules/Kelvin). Conventional computing is done irreversibly as it is based on irreversible operations. For example, a NAND gate irreversibly processes information because the information of two inputs is lost as it cannot be recovered by the knowledge of the output. Thus, the power loss due to information loss is irrelevant to implementation technology aspects. The general TM computation model also suffers from information loss because two transitions may lead to the same configuration of the machine, thus it is an irreversible model of computation.

The question whether computations can be done in a *reversible* manner, that is without information loss, was answered by Bennett [31] when he introduce the *Reversible Turing Machine* (RTM). He proved that any irreversible TM can be simulated by a RTM with polynomial cost in space and time. Namely, an irreversible TM with $O(T)$ and $O(S)$ time and space requirements, can be simulated by a RTM with $O(T)$ and $O(ST)$ time and space requirements.

A more practical model for reversible computation is the *reversible circuit* introduced by Toffoli [29]. A reversible circuit is a logic combinational circuit implementing a function $f : \mathcal{B}^n \rightarrow \mathcal{B}^n$, where $\mathcal{B}\{0, 1\}$, n is the number of input and outputs and f is a bijective function or equivalently a permutation. This means that there is an inverse function $f^{-1} : \mathcal{B}^n \rightarrow \mathcal{B}^n$, which for each x giving $f(x) = y$ then $f^{-1}(y) = x$. A reversible circuit is constructed by combining logic *reversible gates* selected from a reversible library. In a reversible circuit no fanout is allowed at the output of each gate (each output line must be connected to only one input of another gate). Also, as in the conventional combinational circuits, no feedback connections are allowed.

Reversible circuit designs find application in a variation of emerging technologies and computation paradigms, such as low power design [29], quantum computation [41], optical computing [42], DNA computing [43]. E.g., optical computing could exploit sooner the zero energy loss of reversible computing compared to computing using conventional electronics. The reason is that the power loss occurring due to other non fundamental reasons (technological imperfections) in optical devices is lower than the respective power loss of semiconductor devices.

The libraries used to build reversible circuits often contain the following reversible gates: NOT,Controlled-NOT (CNOT) and Toffoli. These are one, two and three bit gates, respectively, with truth tables shown in Table 2.1 and symbols depicted in Figure 2.1.

The first input and output line c of the CNOT gate is called the control bit, while the second

Table 2.1: Truth tables for NOT, Controlled-NOT and Toffoli reversible gates.

in	out
0	1
1	0

c^{in}	t^{in}	c^{out}	t^{out}
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

c_1^{in}	c_2^{in}	t^{in}	c_1^{out}	c_2^{out}	t^{in}
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

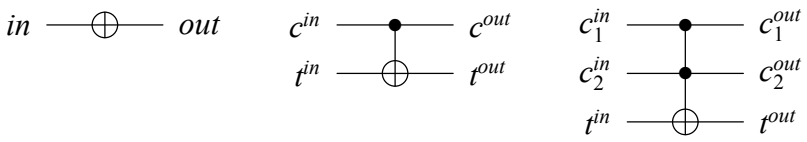


Figure 2.1: Symbols for NOT, CNOT and Toffoli reversible gates

line t is called the target bit. The operation of a CNOT gate is to invert the target bit iff[3] the control bit has logical value 1, while the control bit remains unaffected. The inverse computation, that is to extract the input from the output, is accomplished by connecting a second CNOT gate to the outputs of the first gate as the inverse CNOT is itself.

Toffoli gate is an extension of the CNOT gate to three bits. Now, the first two lines c_1 and c_2 carry the control bits while the last line t is the target bit. Its operation is to invert the target bit iff both control bits have a logic value 1, while both the control bits remain unaffected. The inverse gate of a Toffoli gate is itself. Toffoli gate is a universal gate in the sense that any reversible circuit can be constructed using exclusively Toffoli gates. A Toffoli gate can emulate the irreversible AND gate, if its target input is set to 0. Then the target output will be the logical AND of its two control bits. Similarly, by appending two NOT gates to each control input of a Toffoli gate and setting its target to 1 we can simulate an OR gate.

Sometimes, the function to be emulated in a reversible circuit does not have equal number of inputs and outputs or it is not bijective, thus it is irreversible. E.g. The original irreversible AND gate has two inputs and one output. Its reversible emulation requires the addition of two more input lines and one output line, as the Toffoli gate is a mapping of three inputs to three outputs. The above emulation of an irreversible function by a reversible one can be generalized. In general an irreversible function has the form $f : \mathcal{B}^n \rightarrow \mathcal{B}^m$, where $m < n$ and/or there are $k > 1$ input vectors $x_i \in \mathcal{B}^n, \quad i = 1 \ldots k$ mapped to the same output combination, that is $f(x_1) = \ldots = f(x_k)$. Such an irreversible function can be transformed to a reversible one by embedding it into another constructed reversible function $g : \mathcal{B}^{n+c} \rightarrow \mathcal{B}^{m+g}$ having c additional inputs and g additional outputs relative to the irreversible function f. The reversibility requirement of equal number of inputs and outputs leads to the relation $n + c = m + g$. The n inputs and m outputs are the *primary* ones, while the additional g outputs are the *garbage* ones. The possible addition of c constant variables due to the requirement of the addition of garbage outputs leads to the increment of lines carrying these variables in the implemented circuits. These lines are the *ancillae* of the circuit. The ancilla lines must be reset back to a known constant state, usually the 0 state, in order to be reused later as a constant input to a larger circuit, otherwise they are

[3]"iff" stands for "if and only if"

Table 2.2: Embedding of the irreversible AND function into a reversible function.

a	x	y	z	g_1	g_2
0	**0**	**0**	**0**	0	0
0	**0**	**1**	**0**	0	1
0	**1**	**0**	**0**	1	0
0	**1**	**1**	**1**	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	0
1	1	1	0	1	1

part of the garbage lines set.

A simple embedding example is shown in Table 2.2 for the AND irreversible function, where the embedded AND function subtable is shown in bold. In this case the number of inputs (x and y) is $n = 2$ and the the number of outputs (z) is $m = 1$. Observe that three different input combinations (00,01 and 10) of x and y lead to the same output value $z = 0$. For this reason it is not adequate to add only one output to satisfy the requirement of equal number of inputs and outputs. Instead, two garbage outputs g_1 and g_2 are added to discriminate these three different input combinations. With the additional two garbage outputs an additional constant input a is required to keep the number of inputs equal to the number of outputs. The garbage output values are chosen so as to be different in each input combination that leads to the same output value ($z = 0$) and the rest of the output values are chosen so as no same pattern appears twice in the truth table [29].

Quantum computation is inherently reversible, as it will be discussed in the next section. Thus there is a relation between reversible computation and quantum computation. Reversible computing is a subset of quantum computing and procedures used in reversible design can be used in the design of quantum algorithms. However, a major difference between reversible and quantum computation is that, a possible garbage generation during the quantum computation cannot be simply discarded at the end of the computation as could be done in the reversible case (with the cost of energy loss), as this would affect the outcome of the computation because of the entanglement (see subsection 2.2.5) of quantum bits. For this reason, proper handling of garbage outputs is important in the quantum computation.

2.2 Quantum Mechanics and Computation

Inherent in the assumption of the Strong Church-Turing thesis for the efficient solution of problems is that the operation of a TM (being deterministic or probabilistic) relies on the classical view of the physical laws. This may pose restrictions on the computational power capabilities of the TM model. However, the laws of physics are quantum mechanical in principle. Feynman [44] noted that the simulation of an arbitrary quantum mechanical system by a classical computer (TM) is in general inefficient as it requires exponential resources with respect to the size of the system. Thus, he conjectured that if somehow computers based on the quantum mechanical laws could be build, then they would offer an advantage of computational power relative to classical computers.

The incorporation of the quantum mechanical way that nature works in an abstract manner into the computational processes requires a minimum mathematical framework. This framework will be briefly given in the form of four postulates [41, 45].

2.2.1 State of a closed quantum system

Postulate 2.1. *Every closed physical system is associated to a complex Hilbert space known as* state space *of the system. At any time the system is completely described by its* state vector *of unit norm in the state space.*

Definition 2.2. *(Hilbert Space) A complex Hilbert space is a complete complex vector space with inner product.*

Dirac's *ket* notation $|\psi\rangle$ will be used extensively in this thesis to denote an element (vector) of a Hilbert space. In matrix notation, an element $|\psi\rangle$ of a d-dimensional Hilbert space $\mathcal{H}^d$ is a vector of the form $[a_0, a_1, \ldots, a_{d-1}]^T$ where $a_i \in \mathbb{C}$ ($\mathbb{C}$ is the set of complex numbers) and the symbol ^{T} denotes transposition.

Definition 2.3. *(Computational Basis) The set of vectors in* $\mathcal{H}^d$

$$\mathcal{B} = \left\{ |0\rangle \doteq \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, |1\rangle \doteq \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \ldots, |d\rangle \doteq \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \right\} \tag{2.2}$$

is a basis of $\mathcal{H}^d$ *and it is called the* computational basis*.*

Any state vector of the form $|\psi\rangle = [a_0, a_1, \ldots, a_{d-1}]^T$ can be written as a linear *superposition* of basis states vectors as

$$|\psi\rangle = a_0|0\rangle + a_1|1\rangle + \cdots + a_{d-1}|d-1\rangle \tag{2.3}$$

where the complex coefficients a_i are called the *amplitudes* of the superposition.

Definition 2.4. *(Inner product) An inner product in a complex vector space* $\mathcal{H}^d$ *is a function of two vector arguments from this space which takes complex values, that is* $(\cdot,\cdot) : \mathcal{H}^d \times \mathcal{H}^d \to \mathbb{C}$*, with the following properties:*

Linearity	$(\|\psi\rangle, \sum_i c_i\|\varphi_i\rangle) = \sum_i c_i(\|\psi\rangle, \|\varphi_i\rangle)$
Conjugate Symmetry	$(\|\psi\rangle, \|\varphi\rangle) = (\|\varphi\rangle, \|\psi\rangle)^*$
Positivity	$(\|\psi\rangle, \|\psi\rangle) \geq 0$ *(equality iff* $\|\psi\rangle = 0$*).*

Note that the linearity holds for the second argument only. The first argument has the property of the anti-linearity.

The *dual vector* of $[a_0, a_1, \ldots, a_{d-1}]^T$ is defined as its complex conjugate transpose in matrix notation, that is $[a_0^*, a_1^*, \ldots, a_{d-1}^*]$. The *bra* notation is used for this vector, that is $\langle\psi| \doteq (|\psi\rangle)^\dagger$ where the symbol † denotes conjugate transposition. Using the bra-ket notation for vectors, one can define an inner product of two state vectors $|\psi\rangle$ and $|\varphi\rangle$ as $(|\psi\rangle, |\varphi\rangle) \doteq \langle\psi|\varphi\rangle$, which satisfies the above three properties.

Definition 2.5. *(Norm) The 2-norm of a state vector is defined as* $\|\psi\| \doteq \sqrt{\langle\psi|\psi\rangle}$*.*

The unit norm requirement of Postulate 1 when applied in Eq. (2.3) gives $\sum_i |a_i|^2 = 1$ which is related to the probabilities interpretation of quantum measurement in Postulate 2.3.

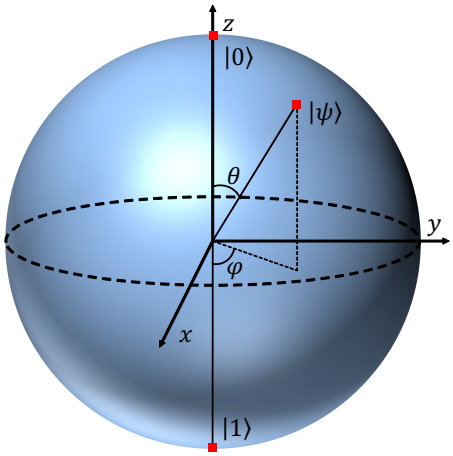


Figure 2.2: Representation of a qubit on the Bloch sphere

The completeness requirement of a set like $\mathcal{H}^d$ means that any sequence of vectors in this space converges to a vector belonging to the same space. A counterexample is the set of rational numbers $\mathbb{Q}$ which is not complete, as the sequence $(1+1/n)^n \in \mathbb{Q}, \quad \forall n \in \mathbb{N}$, while $\lim_{n\to\infty}(1+1/n)^n = e = 2.718\ldots \notin \mathbb{Q}$.

Note that Postulate 2.1 applies to closed systems only; systems that don't interact with their environment. Yet, it is general enough to be used to study open systems if the open system together with its environment is thought to be another closed system.

The simplest quantum physical system is the one associated to a two dimensional Hilbert space $\mathcal{H}^2$. Such a system is called *qubit* or quantum bit and its state can be described as

$$|\psi\rangle = a|0\rangle + b|1\rangle = \begin{bmatrix} a \\ b \end{bmatrix}, \qquad a,b \in \mathbb{C}, \qquad |a|^2 + |b|^2 = 1 \tag{2.4}$$

The constraint $|a|^2 + |b|^2 = 1$ permits the equivalent representation

$$|\psi\rangle = e^{i\omega}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|0\rangle\right), \qquad 0 \le \theta \le \pi, \quad 0 \le \varphi \le 2\pi \tag{2.5}$$

The factor $e^{i\omega}$ is physically unobservable by any measurement and thus the unit state vector of a qubit depends only on the two real parameters θ and φ and consequently lies on the surface of a unit radius sphere called the Bloch sphere [41]. Figure 2.2 depicts the Bloch sphere, the basis vectors $|0\rangle$, $|1\rangle$ (located on the North Pole and the South Pole respectively) and an arbitrary state vector $|\psi\rangle$. Bloch sphere is a useful visualization tool for understanding the operation of various single qubit gates such as the ones discussed in next section.

The physical realization of a qubit can take various forms. E.g. a qubit can be realized as the spin of an electron or the energy level of an atom. In the former case a spin up would correspond to $|0\rangle$ state and a spin down to $|1\rangle$ state. Similarly, the ground state of an atom would correspond to $|0\rangle$, while the first excited state would correspond to $|0\rangle$. Some examples of physical realizations of qubits are discussed in section 2.5 .

2.2.2 Unitary evolution of a closed quantum system

The change of a closed quantum system over time can be stated in its most general form with the following.

Postulate 2.2. *The time evolution of a closed system is defined by a unitary transformation U applied on its state vector as*

$$|\psi_f\rangle = U|\psi_i\rangle \tag{2.6}$$

where $|\psi_i\rangle$ and $|\psi_f\rangle$ are the initial and final states, respectively.

Unitarity of operator U means that $U^\dagger U = U^\dagger U = I$ and this implies that every quantum evolution is reversible as for each operator U its inverse always exists and it holds $U^{-1} = U^\dagger$. The unitary operator U is compatible with the dimensions of the state and for a d-dimensional system it can be expressed by a unitary matrix of dimension $d \times d$.

The above formulation is known as the Schrödinger picture because its continuous time restatement has essentially the solution form of the Schrödinger equation [41]. An alternative formulation is the Heisenberg picture of quantum mechanics where the state is constant and the operator is time dependent, although the two forms are equivalent [46]. The Schrödinger picture is more convenient for the quantum circuit model to be introduced in the next section, as the operator U corresponds to various quantum gates used in the model.

2.2.3 Projective Measurement

The first two postulates deal with closed systems. Except from being an ideal model, a closed system is not useful from the computational perspective. If someone is to obtain computation results then he/she has to interact with the system and perform a *measurement* through establishing correlations of ”his/her own” system to the system under measurement. The third postulate sets the measurement issue in the quantum mechanical view.

Postulate 2.3. *A quantum measurement of a physical system is described by a set of projective matrices $P_m, m = 1 \ldots n$ satisfying the completeness equation*

$$\sum_{m=1}^{n} P_m = I \tag{2.7}$$

If the state of the system before the measurement is $|\psi\rangle$ then the result after the measurement will be m with probability

$$Pr(m) = \langle\psi|P_m|\psi\rangle \tag{2.8}$$

and the post-measurement state of the system will be

$$|\psi_m\rangle = \frac{P_m|\psi\rangle}{\sqrt{Pr(m)}} = \frac{P_m|\psi\rangle}{\sqrt{\langle\psi|P_m|\psi\rangle}} \tag{2.9}$$

The transformation of the state before the measurement state to the post-measurement state is a non unitary transformation and is also called *state collapse* or *state reduction*.

The dimensions of the projective matrices are compatible ($d \times d$) with the dimension d of the system under measurement $\mathcal{H}^d$. The completeness relation (2.7) assures that the sum of probabilities is 1. The projectivity property of P_m means that (i) $P_m^2 = P_m$ and (ii) $P_m^\dagger = P_m$ (Hermitian matrix). The first property assures that a second measurement immediately

after the first gives the same result with probability 1. Two states like $|\psi\rangle$ and $e^{i\varphi}|\psi\rangle$ which differ by a global phase factor $e^{i\varphi}$ are unobservable by any measurement because the measurement probability remains unaltered; $Pr(m) = \langle\psi|e^{-i\varphi}|P_m|e^{i\varphi}|\psi\rangle = \langle\psi|P_m|\psi\rangle$ for any projector P_m.

The above kind of measurements are known as *projective measurement* or *Lüders measurement* [45]. When each projection operator leaves the post-measurement state to a one-dimensional subspace of $\mathcal{H}^d$, meaning that $rank(P_m) = 1$, then we have a *complete* measurement. In this case, the completeness relation (2.7) forces the equality $n = d$, that is the number of different measurement results equals the dimension of the system. This could happen if the projectors were of the form $P_m = |\psi_m\rangle\langle\psi_m|$ where $|\psi_m\rangle$ form an orthonormal basis of $\mathcal{H}^d$ like the one of Eq. (2.2). This special case of projective measurement is called *von Neumann measurement*. Throughout this thesis the measurements are assumed to be von Neumann on the computational basis of Eq. (2.2).

2.2.4 Composition of closed systems

Postulate 2.4. *The state space $\mathcal{H}^d$ of a composite closed system consisting of n subsystems with state spaces $\mathcal{H}_i^{d_i}$, $i = 1 \ldots n$ is their tensor product $\mathcal{H}^d = \mathcal{H}_1^{d_1} \otimes \mathcal{H}_2^{d_2} \otimes \cdots \otimes \mathcal{H}_n^{d_n}$. Moreover, if each subsystem is in the state $|\psi_i\rangle$, $i = 1 \ldots n$, then the state of the composite system is the tensor product $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$.*

The dimension of the composite system is $d = d_1 d_2 \cdots d_n$. The tensor product of two vectors in column format $|a\rangle = [a_1 a_2 \ldots a_p]^T$ and $|b\rangle = [b_1 b_2 \ldots b_q]^T$ is defined through the Kronecker product

$$|a\rangle \otimes |b\rangle \doteq \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} \otimes \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_q \end{bmatrix} = \begin{bmatrix} a_1|b\rangle \\ a_2|b\rangle \\ \vdots \\ a_p|b\rangle \end{bmatrix} = \begin{bmatrix} a_1 b_1 \\ a_1 b_2 \\ \vdots \\ a_1 b_q \\ \vdots \\ a_p b_1 \\ a_p b_2 \\ \vdots \\ a_p b_q \end{bmatrix} \tag{2.10}$$

The generalization to tensor products of more than two vectors is straightforward through the associativity law $|a\rangle \otimes |b\rangle \otimes |c\rangle = |a\rangle \otimes (|b\rangle \otimes |c\rangle) = (|a\rangle \otimes |b\rangle) \otimes |c\rangle$ which holds for tensor products. Equation (2.10) can be used to define a basis for the composite system $\mathcal{H}^d$ from the bases of the subsystems $\mathcal{H}_1^{d_i}$.

The above definition can be extended to tensor products of operators acting on the subsystems. Then, the tensor product is the equivalent operator that acts on the composite system. For example, let A and B be operators acting on subsystems $\mathcal{H}_1^p$ and $\mathcal{H}_2^q$, respectively. Then, U is the equivalent operator acting on the composite system $\mathcal{H}^{pq} = \mathcal{H}_1^p \otimes \mathcal{H}_2^q$ and it is given in matrix block form as

$$A \otimes B \doteq \begin{bmatrix} A_{11}B & A_{12}B & \ldots & A_{1p}B \\ A_{21}B & A_{22}B & \ldots & A_{2p}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{p1}B & A_{p2}B & \ldots & A_{pp}B \end{bmatrix} \tag{2.11}$$

where A_{ij} are the elements of matrix A. The generalization to tensor product of more than two matrices is again straightforward using the associativity law.

Some useful identities that will be used extensively in this thesis are the following, assuming that c is a scalar complex, $|x\rangle, |x_1\rangle, |x_2\rangle \in \mathcal{H}_1^p$, $|y\rangle, |y_1\rangle, |y_2\rangle \in \mathcal{H}_2^q$, A, C operators in $\mathcal{H}_1^p$ and B, D operators in $\mathcal{H}_2^q$

$$c(|x\rangle \otimes |y\rangle) = (c|x\rangle) \otimes |y\rangle = |x\rangle \otimes (c|y\rangle) \tag{2.12}$$
$$(|x_1\rangle + |x_2\rangle) \otimes |y\rangle = |x_1\rangle \otimes |y\rangle + |x_2\rangle \otimes |y\rangle \tag{2.13}$$
$$|x\rangle \otimes (|y_1\rangle + |y_2\rangle) = |x\rangle \otimes |y_1\rangle + |x\rangle \otimes |y_2\rangle \tag{2.14}$$
$$(A \otimes B)(|x\rangle \otimes |y\rangle) = A|x\rangle \otimes B|y\rangle \tag{2.15}$$
$$(A \otimes B)(C \otimes D) = AC \otimes BD \tag{2.16}$$

A composite system consisting of n qubits is essentialy associated to a Hilbert space of 2^n dimensions. The computational basis of this system consists of the tensor products of all the possible combinations of state basis vectors of each qubit. The 2^n computational basis states of the composite system are

$$\begin{aligned}
|0\rangle &= |0\rangle \otimes \cdots \otimes |0\rangle \otimes |0\rangle = |0 \ldots 00\rangle \\
|1\rangle &= |0\rangle \otimes \cdots \otimes |0\rangle \otimes |1\rangle = |0 \ldots 01\rangle \\
|2\rangle &= |0\rangle \otimes \cdots \otimes |1\rangle \otimes |0\rangle = |0 \ldots 10\rangle \\
|3\rangle &= |0\rangle \otimes \cdots \otimes |1\rangle \otimes |1\rangle = |0 \ldots 11\rangle \\
&\vdots \\
|2^n - 2\rangle &= |1\rangle \otimes \cdots \otimes |1\rangle \otimes |0\rangle = |1 \ldots 10\rangle \\
|2^n - 1\rangle &= |1\rangle \otimes \cdots \otimes |1\rangle \otimes |1\rangle = |1 \ldots 11\rangle
\end{aligned} \tag{2.17}$$

Thus the basis state $|k\rangle$ where $k = 0 \ldots 2^n - 1$ of n qubits is the tensor product of the qubit basis states $|0\rangle$ and $|1\rangle$ arranged so as to reflect the binary representation of integer k. In Eq. (2.17) shorthand notations $|\psi_1\rangle|\psi_2\rangle$ and $|\psi_1\psi_2\rangle$ are introduced for tensor product of states, instead of $|\psi_1\rangle \otimes |\psi_2\rangle$. These three notations for tensor product states are used interchangeably in the literature and throughout this thesis text.

The exponential growth 2^n of the composite system dimensions as a function of its size n is the reason why, in general, no efficient simulation of a quantum system can be achieved by a classical computer. It is also one of the reasons for the power of quantum computation as it is shown subsequently.

2.2.5 Entanglement

Postulate 4 states that when the subsystems states $|\psi_i\rangle$ that compose a larger system are known, then the state of this larger system is the tensor product of these states, e.g. $\otimes_{i=1}^{n}|\psi_i\rangle$. The opposite is not true. That is, it is not always possible to write down the state of a composite system as a tensor product state. E.g. the bipartite system state $|\psi\rangle_{AB} = \frac{1}{2}(|00\rangle+|01\rangle+|10\rangle+|11\rangle)$ can be written in product form as $|\psi\rangle_{AB} = \frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$. On the other hand, the state $|\Phi^+\rangle_{AB} \doteq \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ cannot be decomposed in a product state (it is always a sum of product states). This is an example of *entanglement*. The entangled state $|\Phi^+\rangle_{AB}$ is one of the so called four Bell states [41].

Definition 2.6. *(Entangled state) Two subsystems $\mathcal{H}^{d_A}$ and $\mathcal{H}^{d_B}$ composing a larger system $\mathcal{H}^{d_S}$ whose state is $|\psi_S\rangle$ are said to be in entangled state when the state of the larger system*

is not of the product form, that is it cannot be expressed as $|\psi_S\rangle = |\psi_A\rangle \otimes |\psi_B\rangle$ *where* $|\psi_A\rangle \in \mathcal{H}^{d_A}$ *and* $|\psi_B\rangle \in \mathcal{H}^{d_B}$.

Entangled states occur when the subsystems are allowed to interact. The non-separability property of an entangled state means that even if the closed whole system is in a definite state, its subsystems aren't in a definite state in the sense of Postulate 1, and this is understandable because these parts are no longer closed due to their interaction. This phenomenon has no classical counterpart. Entanglement, which is a manifestation of superposition, is believed to be one for the reasons for the power of quantum computation.

A system can always be partitioned in two parts $\mathcal{H}_A^p$ and $\mathcal{H}_B^q$, where p and q are the dimensions of the two subsystems, and its state can be expressed as

$$|\psi\rangle_{AB} = \sum_{i=0}^{p} a_i |\varphi_i\rangle_A |y_i\rangle_B \tag{2.18}$$

where $|\varphi_i\rangle_A$ is an orthonormal basis of $\mathcal{H}_A^p$, $|y_i\rangle_B$ have unit norm and $\sum_{i=0}^{p} |a_i|^2 = 1$. Then, a projective measurement on the orthonormal basis of subsystem A alone would give a result m with probability $|a_m|^2$ and leave the state of the whole system in $|\varphi_m\rangle_A |y_m\rangle_B$. Thus, the state of the system after the measurement is unentangled and both the subsystems are in a definite state, provided the result of the measurement is known, even if only one of the subsystems was measured. The instantaneous influence on one of the subsystems (definite state after measurement) by acting on the other subsystem (measure) was considered first as a thought experiment by Einstein-Podolsky-Rosen [47] and thus pairs of qubits in states such as $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ are referred as EPR pairs.

Entanglement is not only exploited in the design of quantum algorithms but also in quantum cryptography [48], quantum teleportation [49], quantum error correction [50] etc.

2.2.6 Quantum Turing Machine

Deutsch in [7] proposed that Church-Turing thesis should be modified so as to define that computable functions (algorithms) are the ones that can be computed by a real physical system. Then a TM should be modified so as to take into account the quantum mechanical laws of nature. He introduced the Quantum Turing Machine (QTM) by incorporating the above postulates in its operation. He proved that this machine can simulate any TM and that it can also simulate any finite physical system. He also showed that this machine could perform parallel processing exploiting the superposition principle but he pointed out that it couldn't outperform a classical TM from the complexity point of view, as these computations could offer only one result at the end due to the measurement collapse. The QTM model was improved in [36] and it was shown that it could simulate any classical TM with polynomial cost.

2.3 Quantum Circuit Model and Quantum Gates

The universal QTM model of computation, although useful as a theoretical study model, is still cumbersome and can't easily lead to practical implementations of algorithms. Deutsch in [8] introduced the *quantum circuit* model and later [51] its equivalence with the QTM model was shown .

The quantum circuit model of quantum computation is a directed acyclic graph whose nodes correspond to *quantum gates* operating on one or more qubits and the edges correspond to the qubits themselves. A quantum circuit model is a one-to-one mapping of a sequence consisting of unitary evolutions and measurements which operate on qubits

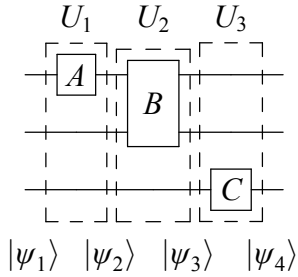


Figure 2.3: An example of quantum circuit model

to a graphical representation like the one shown in the example of Figure 2.3. The initial joint quantum state $|\psi_1\rangle$ of the qubits evolves in time from left to the right of the figure and becomes $|\psi_4\rangle = U|\psi_1\rangle$. The unitary evolution U is a product of unitary operators U_i, where each operator acts on a snapshot of the joint state $|\psi_i\rangle$. In the example shown in Figure 2.3, $U = U_3U_2U_1$. The unitary operators U_i then, are tensor products of the operators acting on each qubit or set of qubits in the respective snapshot. When no such operator acts on some of the qubits (simple wire) the identity operator I is implied. Thus, $U_1 = A \otimes I \otimes I$, $U_2 = B \otimes I$, $U_3 = I \otimes I \otimes C$ and the whole evolution of the quantum circuit in Figure 2.3 can be decomposed as

$$U = (I \otimes I \otimes C)(B \otimes I)(A \otimes I \otimes I) \tag{2.19}$$

A, B and C are quantum gates acting on one,two and one qubits, and can be described by unitary matrices of dimensions 2×2, 4×4 and 2×2, respectively. Below we define a set of quantum gates used frequently in quantum circuits descriptions. These gates are categorized depending on the number of qubits they act on.

2.3.1 Single Qubit Gates

The Pauli gates I, X, Y and Z are defined by their respective matrices as

$$\begin{aligned} I &\doteq \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad X \doteq \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \\ Y &\doteq \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad Z \doteq \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \end{aligned} \tag{2.20}$$

The identity gate I takes no action on a qubit (that is simply a wire that leaves a state intact). The X gate (also called NOT gate) is the quantum analog of a classical NOT gate on the computational basis of one qubit; it evolves $|0\rangle \xrightarrow{X} |1\rangle$ and $|1\rangle \xrightarrow{X} |0\rangle$. The operation of the X gate on a superposition is $a|0\rangle + b|1\rangle \xrightarrow{X} b|0\rangle + a|1\rangle$. The other two gates have

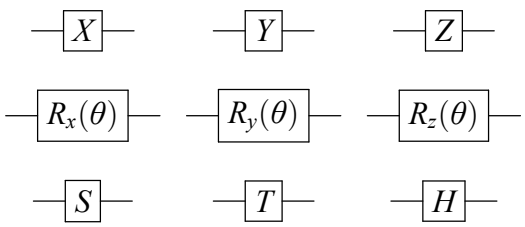


Figure 2.4: Symbols for various single qubit gates

no classical analog and their operation on a superposition is $a|0\rangle + b|1\rangle \xrightarrow{Y} -ib|0\rangle + ia|1\rangle$ and $a|0\rangle + b|1\rangle \xrightarrow{Z} a|0\rangle - b|1\rangle$. The operations of the X,Y and Z gates can be visualized on the Bloch sphere in Figure 2.2 as angle π rotations about the x,y and z axes, respectively, justifying thus their names. The inverses of these gates are the gates themselves, because it holds $X^2 = I$, $Y^2 = I$ and $Z^2 = I$.

A generalization of the above three non trivial Pauli gates X,Y and Z are the single qubit rotation gates $R_x(\theta)$, $R_y(\theta)$ and $R_z(\theta)$ where the angle θ specifies rotation about the x,y and z axes, respectively. They are defined by the matrix exponentials

$$R_x(\theta) \doteq e^{\frac{-i\theta X}{2}} = \begin{bmatrix} \cos(\frac{\theta}{2}) & -i\sin(\frac{\theta}{2}) \\ -i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{bmatrix} \tag{2.21}$$

$$R_y(\theta) \doteq e^{\frac{-i\theta Y}{2}} = \begin{bmatrix} \cos(\frac{\theta}{2}) & -\sin\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{bmatrix} \tag{2.22}$$

$$R_z(\theta) \doteq e^{\frac{-i\theta Z}{2}} = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix} \tag{2.23}$$

The inverse gates of the three rotation gates are the gates themselves but with the opposite sign in their angles, e.g. $R_a^{-1}(\theta) = R_a(-\theta)$, where $a \in \{x, y, z\}$.

Two special cases of the $R_z(\theta)$ gate are the Phase gate and the $\pi/8$ gate, also encountered as S and T gates, respectively. Their matrices are defined as

$$S \doteq \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \quad T \doteq \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix} \tag{2.24}$$

It is obvious that they occur as $R_z(\theta)$ gates with $\theta = \pi/4$ and $\theta = \pi/8$, respectively, if we ommit a global phase of $e^{i\frac{\pi}{2}}$ and $e^{i\frac{\pi}{4}}$, respectively. We have seen that such a global phase on a quantum state is unimportant with respect to measurement.

Finally, the Hadamard gate H is another important single qubit gate used frequently and it is defined as

$$H \doteq \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.25}$$

The Hadamard gate when applied to a computational basis state of a qubit results in an equiprobable superposition, $|0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|1\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. The inverse of H is itself, as $H^2 = I$.

The symbols for the various single qubit gates introduced above are shown in Figure 2.4

2.3.2 Two-Qubit Gates

The two-qubit gates allow interaction between two qubits. The most important two-qubit gate is the CNOT gate (or controlled-NOT gate). The CNOT gate acts on two qubits called control and target. On the computational basis state of both qubits, it inverts (applies the NOT gate) the state of the target qubit iff the control qubit is $|1\rangle$, otherwise it leaves the target intact. In compact form $|c\rangle|t\rangle \xrightarrow{CNOT} |c\rangle|c \oplus t\rangle$ where the $\oplus$ symbol denotes modulo-2 addition and c and t take values of 0 or 1, corresponding to the states of control and target qubit. Linearity of the operator extends this operation to superpositions. E.g. $(a|0\rangle +$

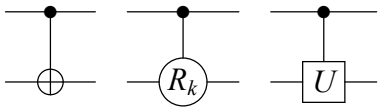


Figure 2.5: Symbols for CNOT gate, controlled rotation gate and general controlled gate.

$b|1\rangle)(c|0\rangle + d|1\rangle) \xrightarrow{CNOT} ac|0\rangle|0\rangle + ad|0\rangle|0\rangle + bc|1\rangle|1\rangle + bd|1\rangle|0\rangle$. The unitary matrix definition for the CNOT gate is

$$CNOT \doteq \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.26}$$

It holds that $CNOT^{-1} = CNOT$.

The notion of a general controlled two-qubit gate can be easily understood if one recognizes that Eq. (2.26) can be written in block structure form as

$$CNOT \doteq \begin{bmatrix} I & 0 \\ 0 & X \end{bmatrix} \tag{2.27}$$

where I is the 2×2, 0 are zero 2×2 matrices and X is the matrix describing the X gate. We can replace X with any unitary operator on a qubit and define any controlled-U gate (c-U). Such a c-U gate would apply the operator U on the target qubit iff the control qubit is in the state $|1\rangle$, otherwise it leaves the target qubit unaffected. In the computational basis of the control qubit, it can be stated that $|c\rangle|\psi\rangle \xrightarrow{CNOT} |c\rangle U^c|\psi\rangle$ where $c = 0, 1$. As in the previous cases, linearity extends this operation to superpositions states.

Unitarity of U and the structure of Eq. (2.27) assure the unitarity of the c-U gate. Its inverse c-U^{-1} has the same form of Eq. (2.27) if we replace U with $U^{-1} = U^\dagger$.

The controlled rotation gates are special cases where the $R_z(\theta)$ unitary matrices take the place of the general U matrix of Eq. (2.27). Namely we define

$$c\text{-}R(\theta) \doteq \begin{bmatrix} I & 0 \\ 0 & R_z(\theta) \end{bmatrix} \tag{2.28}$$

The controlled rotation gates are used in the Quantum Fourier Transform circuit and they are used frequently throughout this thesis text as components for various arithmetic circuits. In such cases, the alternative notation c-$R_k, \quad k \in \mathbb{N}$ is used instead of c-$R(\frac{2\pi}{2^k})$. Symbols for the CNOT gate, the controlled rotation gate c-R_k and the general c-U gate are shown in Figure 2.5.

The SWAP gate is a special gate that exchanges the states of two qubits. It performs the transformation $|\psi_1\rangle|\psi_2\rangle \xrightarrow{SWAP} |\psi_2|\psi_1\rangle$. The SWAP gate symbol is shown in Figure 2.6.

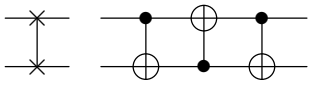


Figure 2.6: SWAP gate and an implementation using three CNOT gates.

Depending on the physical realization, it can be implemented using three CNOT gates (depicted on the right of the Figure) or by physical movements of the qubit carriers (e.g. movement of ions).

2.3.3 Three-Qubit Gates

Extending the notion of the CNOT gate to three qubits (two of them acting as control and one as target qubit) someone can define the Toffoli gate as $|c_1\rangle|c_2\rangle|t\rangle \xrightarrow{Toffoli} |c_1\rangle|c_2\rangle|c_1c_2 \oplus t\rangle$. The target qubit is inverted iff both the control qubits are in state $|1\rangle$. Toffoli gate is also called CCNOT. In matrix form its definition is given by

$$CCNOT \doteq \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.29}$$

or in block form

$$CCNOT \doteq \begin{bmatrix} I & 0 \\ 0 & X \end{bmatrix} \tag{2.30}$$

where I is the 6×6 identity matrix and 0 are zero matrices of appropriate dimensions. Like in the case of two qubits, this block form leads naturally to the definition of a double controlled-U gate (c^2-U) if X is replaced by any unitary 2×2 matrix U. The definition of a double controlled rotation gate is

$$c^2R(\theta) \doteq \begin{bmatrix} I & 0 \\ 0 & R_z(\theta) \end{bmatrix} \tag{2.31}$$

Finally, the Fredkin gate (or controlled-SWAP gate) acts on one control and two target qubits by swapping the states of the two target qubits iff the control qubit is in the $|1\rangle$ state. In the computation basis of the control qubit it can be described with the transformations $|0\rangle|\psi_1\rangle|\psi_2\rangle \xrightarrow{Fredkin} |0\rangle|\psi_1\rangle|\psi_2\rangle$ and $|1\rangle|\psi_1\rangle|\psi_2\rangle \xrightarrow{Fredkin} |1\rangle|\psi_2\rangle|\psi_1\rangle$. The Fredkin gate can be decomposed into three Toffoli gates, like the SWAP gate is decomposed into three CNOT gates.

Symbols for the Toffoli gate, the general c^2-U gate and the Fredkin gate are shown in Figure 2.7.

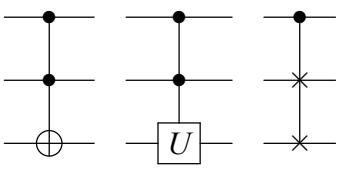


Figure 2.7: Symbols for Toffoli gate, general c^2-U gate and Fredkin (controlled SWAP) gate

2.3.4 Measurement

The quantum circuit model contains special measurement gates to extract classical results of the quantum computation. The symbol of the one qubit measurement gate for the complete projective measurement on the computational basis $\{|0\rangle, |1\rangle\}$ is shown in Figure 2.8. Classical results of the measurement are denoted with the double line wire after the measurement gate and can be used when they classically control a quantum gate after the measurement.

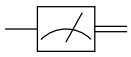


Figure 2.8: Symbol for the computational basis measurement gate.

2.3.5 Universal Gates and Synthesis of Quantum Circuits

The quantum circuit model, being equivalent to the QTM model, is adequate for any quantum computation task. The next question is which are the required gates to be used in the composition of any quantum circuit. These gates form a set called *universal set of gates*, the same way as the NAND or the NOR gate is a universal gate for the classical logic circuits.

Definition 2.7. *A set of quantum gates $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ is called universal set if it can approximate with arbitrary low error any quantum circuit.*

The notion of the above definition is that a sequence of gates drawn from this set can be used to approximate a quantum circuit represented by a unitary matrix U so as the derived circuit $\hat{U}$ is as close to U, e.g. it has the property $\forall \varepsilon > 0, \exists \hat{U} : \max_{|\psi\rangle} \| \langle \psi | (U - \hat{U}) | \psi \rangle \| < \varepsilon$.

The introduction of the quantum circuit model in [8] was accompanied with the proof of the existence of a universal gate, called Deutsch gate. This is a three-qubit parametrized gate defined in block matrix form by

$$Q(a) \doteq \begin{bmatrix} I & 0 \\ 0 & D(a) \end{bmatrix} \tag{2.32}$$

where

$$D(a) \doteq \begin{bmatrix} i\cos(\frac{\pi a}{2}) & \sin(\frac{\pi a}{2}) \\ \sin(\frac{\pi a}{2}) & i\cos(\frac{\pi a}{2}) \end{bmatrix} \tag{2.33}$$

and a is an irrational number. Thus, this universal set is $\mathcal{G}_3 = \{Q(a)\}$ for some $a \in \mathbb{R} - \mathbb{Q}$.

DiVincenzo in [52] improved this result proving that a family of two-qubit gates are universal. This universal set is $\mathcal{G}_2 = \{X, c\text{-}R_x(\theta), c\text{-}R_y(\theta), c\text{-}R_z(\theta) \,|\, \theta \in \mathbb{R}\}$. In this set, $c\text{-}R_x(\theta)$, $c\text{-}R_y(\theta)$ and $c\text{-}R_z(\theta)$ are controlled rotation gates of the form 2.31 with the substitutions $R_x(\theta)$, $R_y(\theta)$, $R_z(\theta)$ in place of the lower right block, respectively. This is an important result from the implementation perspective, because quantum gates with more than two qubits usually involve the interaction of more than two physical subsystems at a time instance, something which is hard to achieve in a controlled manner.

A result that narrowed down a possible universal set proved by Barenco et al. in [53] states that the set $\mathcal{G}_{1,CNOT} = \{CNOT, R_x(\theta), R_y(\theta), R_z(\theta) | \theta \in \mathbb{R}\}$ is universal. This set consists of single-qubit rotation gates of Eq. (2.21), (2.22), (2.23) and the CNOT gate. This result

contributes to practical realizations of quantum circuits because only one kind of two-qubit gate is required and the single qubit gates are relatively easily implemented.

The set $\mathcal{G}_{1,CNOT}$ is uncountable (there is no one-to-one mapping with any subset of the natural numbers). This may pose implementation problems as in a particular technology only a finite set of gates may be realizable. Moreover, fault tolerance requirements compels the usage of a finite discrete universal set. For this reason, it would be desirable to approximate with arbitrary low error any quantum circuit with a sequence of discrete set of gates. Indeed, it has been shown [54, 55] that almost any two-qubit gate is universal (one member set). Yet, this universality comes at a large cost in the number of the required gates, .e.g. exponential with respect to the required accuracy $\log(1/\varepsilon)$.

Solovay-Kitaev theorem [56, 57] and improvements [58, 59, 60] come to resolve this situation. Expressly, these results can be used to approximate any single qubit gate of the set $\mathcal{G}_{1,CNOT}$ with a sequence of single qubit gates drawn from a discrete set at a polynomial cost in $\log(1/\varepsilon)$. A discrete set of gates frequently used to synthesize a quantum circuit is the $\mathcal{G}_{Fl} = \{H, T, CNOT\}$ because these three gates can be relatively easily built in a fault tolerant manner [50], although other discrete sets can be used as well. The design of a quantum circuit satisfying a particular specification can exploit these results.

The specification of an n qubits circuit can be in the form of the unitary matrix U with dimensions $2^n \times 2^n$. Then, the design can be accomplished with an automatic *quantum synthesis* procedure taking as input the specification and giving as output a network consisting of quantum gates drawn from a universal set and a connection between them. Usually, such a synthesis is performed in two steps: First, the large matrix U is gradually decomposed in a sequence (products and tensor products) of smaller dimension matrices up to reaching the dimensions covered by the universal set (single and two-qubit gates). This step is an exact low-level synthesis down to the primitive quantum gates of the universal set. If the universal set used is an uncountable one then a second step follows to transform the continuous set of gates to an approximation with a sequence of gates from a discrete set. This kind of synthesis incorporating these two steps can be described as *flat synthesis* and the approach is a top-down one.

The are some other approaches to design a quantum circuit. Quantum circuit synthesis differs from reversible circuit synthesis in the specifications and the set of gates used to synthesize the circuit. Boolean specifications in the computational basis are adequate when the target circuit is an arithmetic one or a logical one due to the linearity and the superposition principle. Thus, reversible circuit methodologies can be used and then gate transformation can be applied to convert the reversible set of gates to a quantum set of gates (if they differ). For example, the design of a quantum adder circuit of two n-bit integers a and b can be specified by the input-output mapping $|a\rangle|b\rangle \rightarrow |a\rangle|b+a\rangle$ when $|a\rangle$ and $|b\rangle$ are states in the computational basis $\{|0\rangle, |1\rangle, \ldots, |2^n-1\rangle\}$ without mentioning its unitary matrix. The quantum adder circuit can be designed using well known results from classical digital circuits (taking into account reversibility requirements) or classical reversible circuits using gates such as CNOT, Toffoli etc. Adders in [61, 62] are examples of such designs.

Other design methods exploit the availability of already designed blocks and combine them ad-hoc or automaticaly to gradually build larger circuits climbing the complexity ladder . These methods fall in *hierarchical synthesis* category. A novel hierarhical synthesis method will be described in Chapter 6.

Important characteristics (defined in next subsection) of a particular synthesis method related to the circuits generated are the quantum cost, the depth of the circuit, the ancilla and

the possible garbage generated. Another important factor to account for is the execution time and memory requirements of the synthesis algorithm itself.

2.3.6 Quantum Circuit Characterization

Elementary gates are the ones that can be easily implemented in a given technology and they must form a universal set. Not all the elementary gates have the same implementation cost in terms of area, speed operation and other resources in a given technology. E.g. single qubit gates are more easily constructed in various technologies. This estimation changes if one takes into account the fault tolerant implementation requirement. In this case single qubit gates X,Y,Z,S,H and $CNOT$ gates can be assumed to have roughly the same cost, while the T gate is thought to be harder to implement, as its fault tolerant version is more complicated.

The performance study of various quantum circuits and architectures involves several metrics. The most important and most frequently used ones are:

Width Total number of qubits used in the circuit, including ancilla ones.

Quantum Cost A weighted sum of the elementary gates used. The weight of each gate is related to its fault tolerance implementation cost. A unit cost is assigned to the less costly gate.

Depth The largest weighted sum of elementary gates acting on any input-output path where the calculated weight is related to the speed of the gate. This metric is related to the latency of the circuit. A unit weight is assigned to the faster gate of a particular physical implementation.

Ancilla The number of qubits that participate in the intermediate steps of the computation but are not part of the input nor of the end results. They are the analogous of the temporary variables in classical computation. A qubit is characterized as ancilla if its state before and after the computation is constant (usually zero) and irrelevant to the input given to the quantum circuit.

2.4 Quantum Algorithms

Computation based on the quantum mechanics laws offers parallelism due to the superposition of basis states (2^n states for n qubits) that are allowed in principle. The evolution principle then, leads the state to a superposition of states, where each state of the superposition is the value of a function corresponding to a different basis state. However, the measurement postulate prevents us from reading all these results. The first evidence that we can exploit this exponentially grown parallelism was shown by the Deutsch algorithm [7] which is a very simple algorithm with no practical application.

2.4.1 Deutsch's algorithm

The statement of the problem that Deutsch's algorithm solves is the following: Given a function $f : \{0,1\} \rightarrow \{0,1\}$ find if f is constant, $f(0) = f(1)$, or balanced, $f(0) = \overline{f(1)}$. The complexity metric will be how many times it is required to call the function evaluator, which is called *oracle*. In a classical computer two such evaluations are needed.

Figure 2.9 shows the quantum algorithm to solve Deutsch's problem in the form of a quantum circuit. Two qubits are used in this circuit, three Hadamard gates, a measurement gate, and a two-qubit quantum oracle U_f whose internal structure doesn't concern us here. The purpose of this oracle is to evaluate the function f in a unitary manner so as to satisfy the

unitarity evolution postulate. The operation of the unitary U_f can be described in the computational basis of the two qubits as $U_f(|x\rangle|y\rangle) = |x\rangle|y \oplus f(x)\rangle$ where $x, y \in 0, 1$ and $\oplus$ is the symbol for addition modulo 2. Clearly, U_f is unitary as $U_f U_f(|x\rangle|y\rangle) = |x\rangle|y \oplus f(x) \oplus f(x)\rangle = |x\rangle|y\rangle$, thus $U_f = U_f^{-1}$. Also, $U_f^* = U_f$ and we conclude that $U_f = U_f^\dagger$.

The initial state for top and bottom qubit is $|0\rangle$ and $|1\rangle$, respectively. The joint initial state is their tensor product $|\psi_0\rangle = |0\rangle|1\rangle$. The evolution through the two left Hadamard gates leads to

$$\begin{aligned} |\psi_1\rangle &= (H \otimes H)|\psi_0\rangle \\ &= H|0\rangle \otimes H|1\rangle \\ &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{aligned} \tag{2.34}$$

The action of U_f when the top qubit is in one of the computational basis states $|x\rangle$, $x = 0, 1$ and the bottom qubit is in the state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ is described by

$$\begin{aligned} U_f\left(|x\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) &= |x\rangle\frac{1}{\sqrt{2}}(|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle) \\ &= |x\rangle\frac{1}{\sqrt{2}}(-1)^{f(x)}(|0\rangle - |1\rangle) \\ &= (-1)^{f(x)}|x\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{aligned} \tag{2.35}$$

In the last equality of Eq. (2.35) the phase factor $(-1)^{f(x)}$ is transfered from the amplitude of the second qubit to the amplitude of the first one, due to the tensor product property of Eq. (2.12). This phase transfer is referred to as *phase kick-back* and will be used repeatedly.

Now the state $|\psi_2\rangle$ on Figure 2.9 can be calculated by combining Eq. (2.34) and (2.35) as

$$\begin{aligned} |\psi_2\rangle &= U_f|\psi_1\rangle \\ &= (-1)^{f(0)}\frac{1}{\sqrt{2}}|0\rangle\left(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) + (-1)^{f(1)}\frac{1}{\sqrt{2}}|1\rangle\left(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) \\ &= \frac{1}{\sqrt{2}}\left((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\right)\frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right) \\ &= (-1)^{f(0)}\frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle\right)\frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right) \end{aligned} \tag{2.36}$$

The last step before the measurement is to apply a Hadamard gate on the top qubit and this gives the joint state

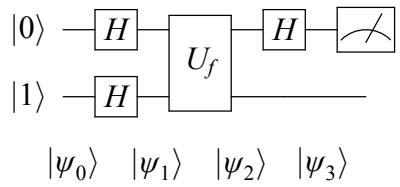


Figure 2.9: Quantum circuit for Deutsch's algorithm

$$
\begin{aligned}
|\psi_3\rangle &= (H \otimes I)|\psi_2\rangle \\
&= (-1)^{f(0)} H \left(\frac{1}{\sqrt{2}}|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle \right) \otimes (|0\rangle - |1\rangle) \\
&= (-1)^{f(0)}|f(0) \oplus f(1)\rangle \frac{1}{\sqrt{2}}|0\rangle - |1\rangle
\end{aligned}
\tag{2.37}
$$

Observe that when the function f is constant, then $f(0) \oplus f(1) = 0$ and $(-1)^{f(0)\oplus f(1)} = 1$, while when the function f is balanced, then $f(0) \oplus f(1) = 1$ and $(-1)^{f(0)\oplus f(1)} = -1$.

The measurement of the top qubit gives the result 0 with probability 1 if $f(0) \oplus f(1) = 0$ which occurs iff f is constant, otherwise it gives the result 1 with probability 1 if $f(0) \oplus f(1) = 1$ which occurs iff f is balanced. (The global phase $(-1)^{f(0)}$ is unobservable in the measurement process). We conclude that Deutsch's algorithm decides with certainty if the function f is constant or balanced using only one evaluation of f whereas a classical algorithm uses two evaluations.

The presented formulation of Deutsch's algorithm is an improvement that appeared in [63]. The original formulation of [7] lacks the Hadamard gate on the lower qubit, thus the state fed to the U_f is $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle$ and then after the application of the U_f two Hadamard gates were used to both qubits before a measurement of both of them, instead of a single qubit measurement. This resulted in a $\frac{1}{2}$ probability to decide if f is constant or balanced and another $\frac{1}{2}$ probability to fail, which is no succesful decision. This result could be probabilistically obtained on a classical computer as well, with only one evaluation of f and so the original Deutsch's formulation offered no advantage relative to a probabilistic classical computation.

2.4.2 Generalizations - Phase Estimation Algorithms

Deutsch's algorithm has no practical application, nor does it offer any significant speed-up (one evaluation of the oracle instead of two). Nevertheless, through its generalization it opened the way to more useful algorithms such as Shor's algorithm. Below, some key aspects of Deutsch's algorithm that remain invariant in a whole family of algorithms, known as *phase estimation algorithms*, are presented [63].

Superposition — In Deutsch's algorithm the state of the top qubit is set to a superposition of the computational basis state $|0\rangle$ and $|1\rangle$ using a Hadamard gate. In general a collection of n control qubits, collected as a control register, are set in a superposition of their computational basis states $\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle$ using n Hadamard gates, each one operating on a qubit of this control register.

Target Register — The bottom qubit can be generalized in a collection of m qubits, called the target register.

Oracle — A quantum subcircuit U_f that operates on both the control and target register is used, and its purpose is to evaluate a function f of the state of the control register $|x\rangle$ by altering the target register accordingly, in general as $U_f|x\rangle|y\rangle = |x\rangle|g(y, f(x))\rangle$. In Deutsch's algorithm $g(\cdot,\cdot) = (\cdot \oplus \cdot)$.

Eigenvalue & Phase Kick-Back — Observe that in Eq. (2.35) the state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ of the target register is an eigenstate of the operator U_f, if this operator is thought as an operator acting on target register only and controlled by the control register. Moreover,

the eigenvalues depend on $f(x)$ and in the particular Deutsch's algorithm they are $(-1)^{f(x)}$. These eigenvalues are kicked-back on the control register as amplitudes on the superposition of the computational basis states $|x\rangle$.

Interference & Phase Estimation

Deutsch's problem can be defined as a problem of estimating amplitude phases of the control register superposition as shown in Eq. (2.36). E.g. the state of the control register can be written as $\frac{1}{\sqrt{2}}\sum_{x=0}^{1} e^{i2\pi\omega x}|x\rangle$ and the phase parameter ω takes values 0 or $1/2$, depending on if the function f is constant or balanced, respectively. Thus, the problem is reduced to the estimation of the phase parameter ω. This estimation is done by interfering the parallel branches of the computation that occur due to the superposition and combining the various amplitudes so as to enhance or cancel them. In Deutsch's algorithm this interference is accomplished with the Hadamard gate on the right, just before the measurement. In a general setting of quantum phase estimation, as it will be shown in Chapter 3, the interference is accomplished using the *Quantum Fourier Transform* (QFT). In Deutsch's special case, the Hadamard gate performs the QFT on just one qubit.

The Deutsch-Josza algorithm [64] is an example that uses a non trivial control register. This problem is described as follows: Given a Boolean function f of n variables $f: \mathcal{B}^n \rightarrow \mathcal{B}$ and the promise that it is constant or balanced (half of its values are 0 and the other half are 1), decide in which one of the two categories it falls in. The quantum algorithm shown in quantum circuit representation is depicted in Figure 2.10. The oracle U_f applies the transformation $|x\rangle|y\rangle \xrightarrow{U_f} |x\rangle|y \oplus f(x)\rangle$ where $x \in \mathcal{B}^n$, that is a string of bits representing the n qubits computational basis or equivalently an integer between 0 and $2^n - 1$. An analysis similar to the previous case of the simple Deutsch's algorithm leads to the conclusion that a measurement of the control register gives the integer 0 with certainty, iff f is constant, while it gives with certainty a value different from 0, iff f is balanced. Thus, with only one invocation of U_f the problem can be solved with certainty.

On a classical computer the same problem would be solved with $2^{n-1}+1$ queries, in the worst case. Yet, it can be shown that there is a classical probabilistic algorithm that solves this problem with $n+1$ queries and $\frac{1}{2^n}$ probability of error. An arbitrary low probability of error can be achieved with linear number of queries, thus this problem is not considered as a hard problem, even if deterministically it needs exponential number of queries as a function of its size n.

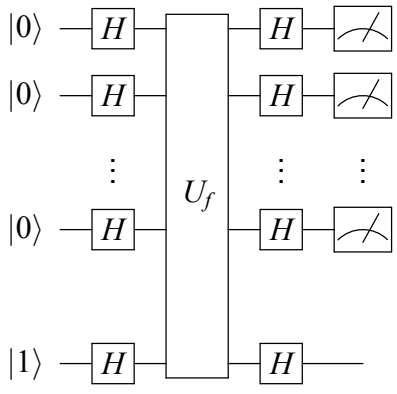


Figure 2.10: Quantum circuit for Deutsch-Josza algorithm

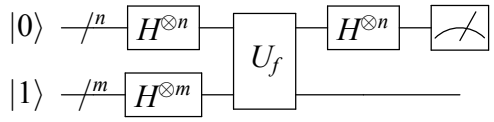


Figure 2.11: Quantum circuit for Simons's algorithm

One of the problems that showed a clear superiority of quantum relative classical computation in terms of complexity was Simon's problem [9]. This problem is described as follows: We are given a function $f : \mathcal{B}^n \rightarrow \mathcal{B}^m$ with $m \geq n$ and the property that f is xor invariant under some mask s, that is $\forall x \neq x', \exists s \in \mathcal{B}^n : f(x) = f(x') \Leftrightarrow x' = x \oplus s$. The $\oplus$ symbol is now the operation of bitwise XOR. The problem is to find the mask s. Note that, in the case of $s = 0$ the function f is 1-1.

Figure 2.11 shows the circuit for the quantum part of Simon's algorithm. Qubit lines of control and target registers are not shown individually, but instead, they are shown collectively as a single line with the width of each register attached to it (n and m for the control and target register, respectively). The notation $H^{\otimes n}$ is the tensor product of n Hadamard gates, each one acting in paallel on one qubit of the respective register. The oracle U_f computes $|x\rangle|y\rangle \xrightarrow{U_f} |x\rangle|y \oplus f(x)\rangle$ where $\oplus$ stands for the bitwise XOR operation.

Simon's algorithm consists of a repeated execution of the quantum part depicted in 2.11 about n times and then classical post processing of the measurements results with $O(n^3)$ complexity. It has been proven that it can solve with success probability at least $2/3$ the problem of bitwise XOR masking as posed above. On the contrast it can be proven that any classical algorithm solves the same problem with success probability at least $2/3$ using $\Omega(2^{n/3})$. Thus, Simon's problem was one of the first problems that showed a superpolynomial superiority of the quantum computation relative to classical computation.

At the same time, Shor announced [10] his famous quantum algorithm which, among others, solves the hard problem of factoring an integer into its prime factors in polynomial time. This algorithm belongs to the same family of quantum phase estimation algorithms and it is extensively discussed in Chapter 3. Variations of this algorithm solve the discrete logarithm problem and the more general hidden subgroup problem.

An important point about these algorithms is that their circuits must be efficient. Until now, the oracles were presented as black boxes without mentioning their inner implementation details. Fortunately, it is proven that such oracles can be constructed efficiently (polynomial circuit cost,depth etc). In Chapter 4 a set of novel efficient quantum arithmetic circuits (multipliers/accumulators, multipliers, dividers,etc) is presented and the way they can be combined to build a fast quantum modular exponentiator circuit which is an integral part of Shor's algorithm and the quantum phase estimation algorithm in general, is shown.

2.4.3 Other quantum algorithms and applications

While Shor's algorithm offered a superpolynomial speed-up in relation to classical algorithms, its applications are restricted mainly in cryptanalysis and algebraic problems. A quantum algorithm that is general enough and applies to a varied field of applications is Grover's algorithm [11]. This is a quantum search algorithm in an unstructured "database". The database consists of a large number N of "objects" indexed by an integer $k = 0 \ldots N-1$. Each object has a property that is satisfied or not. An oracle is available which, given an index k of the object, it can answer whether the object has the satisfying property, e.g. it computes the function $f : \{0 \ldots N-1\} \rightarrow \{0,1\}$ where $f(k) = 1$ if the k indexed object is a solution, otherwise $f(k) = 0$. The problem is to find all the objects (indexes) in the database

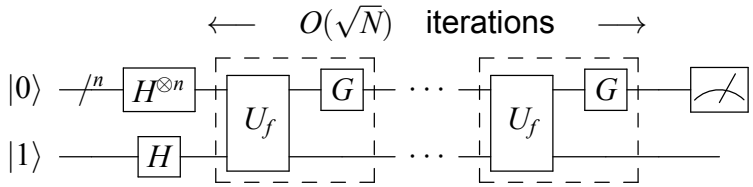


Figure 2.12: Quantum circuit for Grover's algorithm. Dashed boxes are the Grover operator while G is the Grover diffusion operator.

that satisfy the required property.

Figure 2.12 depicts Grover's algorithm in a very abstract level. The top n qubits register, with n chosen such as that $2^n \geq N$, is the index register whose state after the n Hadamard gates becomes $\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n} |k\rangle$. The oracle is fed with this superposition and the state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ om the lower qubit. The operation of the oracle is to compute the function f as follows:

$$|k\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \xrightarrow{U_f} (-1)^{f(k)}|k\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{2.38}$$

Thus, the solution is again encoded in the phases of the first register while the lower qubit remains unaltered. Assuming that s is the only solution, then after the application of U_f on the superposition states of the first register and ignoring the lower qubit as it stays unaltered, we have the following state for the index register

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \left(-|s\rangle + \sum_{j=0, j\neq s}^{N-1} |j\rangle \right) \tag{2.39}$$

The amplitude of the basis state $|s\rangle$ which belongs to the solution is $-\frac{1}{\sqrt{2^n}}$ while the amplitudes of every other basis state are $\frac{1}{\sqrt{2^n}}$. The purpose of the G operators (Grover's diffusion operator) in Figure 2.12 is to gradually increase the absolute amplitude of the solution state's $|s\rangle$ while at the same time to decrease the other amplitudes of the non solution states $|k\rangle$, $k \neq s$. This procedure is called *amplitude amplification*. The detailed operation of the diffusion operator G is not explained in this thesis, but it is enough to give its definition as $G = H^{\otimes n}(2|0\rangle\langle 0| - I)H^{\otimes n}$. Each dashed box in Figure 2.12 which contains U_f and G is called Grover's operator. It can be proven that an iteration of $O(\sqrt{N})$ Grover's operators (and thus the same number of U_f oracle invocations) is adequate to amplify the absolute value of the solution state $|s\rangle$ so as at the end, a measurement gives the solution s with high probability.

A classical algorithm must query the oracle N times to find the desired objects. On the contrast Grovers's algorithm finds the solution by invoking $O(\sqrt{N})$ queries. Although this quadratic speed-up is not as dramatic as the superpolynomial speed-up of Shor's algorithmm, this search algorithm is quite general. E.g. it can be applied to various NP problems having a complexity of $O(2^n)$ to lower it to $O(2^{n/2})$, an improvement which may be adequate for many applications.

Another family of quantum algorithms which are related to Grover's search algorithm are the quantum walks algorithms. These are the quantum analogue of classical random walk algorithms and they find applications in various problems. They offer superpolynomial (welded-tree [65], hidden non-linear structures [66]) or polynomial speed-up (triangle prob-

lem [67], element distinctness [68], verifying matrix products [69], etc.) compared to classical computation. Overwiews of quantum walks algorithms can be found in [70] and [71].

In general, the various quantum algorithms developed for specific problems fall into two broad categories: Quantum Phase Estimation like Shor's algorithm and search based like Grover's and quantum walks algorithms. An updated collection of quantum algorithms with references can be found on-line at [72].

Simulation of quantum systems (quantum physics,quantum chemistry) will probably be the most promising application of quantum computation. Anyway, the simulation of physical systems at the quantum mechanical level was the initial stimulus for the study of whether quantum computers as exposed by Feynman and others are feasible. The purpose of the simulation of a quantum system is to predict its time evolution given some initial state. The time evolution of a closed system as predicted by Eq. (2.6) of Postulate 2 is essentially the solution of the Schrödinger differential equation

$$i\hbar\frac{d}{dt}|\psi(t)\rangle = H|\psi(t)\rangle \tag{2.40}$$

where $|\psi(t)\rangle$ is essentialy the wave function, $\hbar$ is Planck's constant and H is the so called Hamiltionian of the system which represents its energy. The solution given in Eq. (2.6) is valid for the initial condition $|\psi_i\rangle = |\psi(0)\rangle$ and gives the final state $|\psi_f\rangle = |\psi(t)\rangle$ for every t. The unitary matrix U is related to the Hamiltonian as $U = e^{i\hbar Ht}$ for some specific t. The difficulty of such a simulation in a classical computer arises from the fact that the dimension of system's state $|\psi(t)\rangle$ grows exponentially as system's size n, namely 2^n, as implied by Postulate 4. This means that there is an exponential requirement of memory resources. Moreover, the dimension of the Hamiltonian H grows also exponentially ($2^n \times 2^n$) which means that the matrix exponentiation and multiplication needs exponential running time.

The simulation of quantum systems by a quantum computer based on the quantum circuit model can be done efficiently on the conditions that (i) the initial state $|\psi_i\rangle$ can be prepared, (ii) the unitary matrix U corresponding to the particular Hamiltonian H can be efficiently expressed in a polynomial cost and depth quantum circuit, and (iii) useful information can be extracted by an efficient measurement of the final state $|\psi_f\rangle$. Fortunately, due to existence of local interactions in many physical systems it is pointed out that such simulations are feasible [12, 73, 74, 75, 76, 77].

Although the simulation of a variety of quantum systems can be done efficiently on a quantum computer, at the end some measurement must be performed to retrieve useful results, but this measurement lead to the collapse of the state vector. Thus, only part of the information simulated up to the moment of the measurement is available. Yet, this information may be related to properties such as energy gaps, eigenvalues and eigenvectors, correlation functions and spectra. If the full quantum state estimation is required then quantum state tomography [41] methods could be used but they have the disadvantage that they scale exponentially on the problem size.

The applications of quantum simulations lie on a very wide field (nuclear physics, atomic physics, condensed matter physics,quantum field theory, cosmology, quantum chaos, quantum chemistry,etc.). Reviews on the subject of quantum simulation can be found in [78] and [79]. The quantum simulation of physical systems is a very promising application of quantum computation and in fact it will be the first application of quantum computers. The reason is that a few tens of qubits are adequate to simulate physical systems of interest outperforming the current classical computer capabilities. In contrast, to factor a

1000 bits integer using Shor's algorithm would require (accounting the necessary error correction involved) some millions of qubits. In Section 2.5 recent advances on some experimental set-ups related to this promising application are summarized.

2.4.4 Quantum Complexity

The evidence that quantum computation may solve problems that are believed to be intractable for classical quantum computation brings the need to define a new complexity class related to quantum computation. This class is called *Bounded-error Quantum Polynomial time* or BQP [36].

Definition 2.8. *(BQP complexity) The Bounded-error Quantum Polynomial time complexity class consists of all the problems that can be solved in polynomial time on a quantum computer with probability of error at most $\frac{1}{3}$ for all instances of the problem.*

The probability of $\frac{1}{3}$ is arbitrary and in fact every real number $0 < k < \frac{1}{2}$ leaves the BQP set imperishable. The above definition is the quantum analog of the classical BPP class. The proven relations between BQP and other complexity classes are that $BPP \subseteq BQP$ and $BQP \subseteq PSPACE$ [36]. Figure 2.13 shows the relations of BQP class to other classical complexity classes. In this figure all of the inclusions are shown as strict (otherwise all the classes would collapse to only one), although the fact that these inclusions are proper is not yet proven, but it is widely believed to be so. An exception is the conjecture that $P = BPP$.

Some consequences of possible proofs of whether some of the above inclusions are strict or not follow. The most important one is that if the suspected superior power of quantum computers compared to classical ones is proven ($BPP \neq BQP$ and consequently $P \neq BQP$) then $P \neq PSPACE$, which is a problem that remains unsolved until today. The proof that $BQP \neq BPP$ would also invalidate the Strong Church-Turing thesis. On the other hand, if it will be ever proven that $P = PSPACE$ then this would automatically mean that $P = BPP = BQP$ and thus a quantum computer would not offer any computational power advantage over a classical computer.

Observe that it is not known even if $BQP \subseteq NP$. The problem of Recursive Fourier Sampling [36] gives indications that it may not be so and for this reason part of the ellipsis denoting the BQP class lies outside the circle of the NP class. If, on the other hand, it holds that $BQP \subseteq NP$ then the rigorous proof of the quantum computation advantage $BPP \neq BQP$ would lead to the conclusion $P \neq NP$.

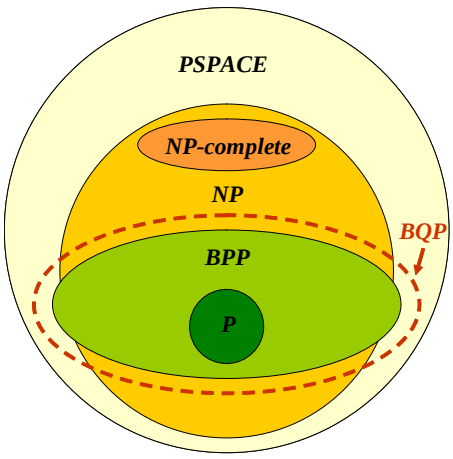


Figure 2.13: The suspected relations among various classical complexity classes and BQP.

Looking for the inverse relation between BQP and NP, it is not known whether every NP problem can be solved efficiently on a quantum computer. In fact none of the efficient quantum algorithms in the literature solve any $NP - complete$ problem. If any efficient algorithm will ever be proposed for any $NP - complete$ problem then this would mean that $NP \subseteq BQP$.

2.5 Physical Implementations

The feasibility of building real devices that perform quantum information processing tasks relies on the choice of a particular quantum mechanical system capable to process and store quantum information. In the last thirty years much progress has been made in experimental realization of quantum information processing devices and even some commercial products have appeared, but they are limited to small scale blocks. E.g. quantum cryptography systems, quantum teleportation demonstrations, quantum gates implementations and quantum computers of a few qubits. Yet, the physical implementation of a large scale quantum computer operating on thousands of qubits is still away from current technological capabilities.

The physical representation must meet some minimum requirements if it is to be considered a candidate for a large scale quantum computer implementation. DiVincenzo [80] introduced five criteria that must be met by any implementation technology. These criteria are briefly exposed below

1. Scalable physical system with well-characterized qubits.

   Usually, the physical system representing the qubit may lie in a Hilbert space which is larger than the desired dimension of the qubit which is two dimensional. E.g. if the energy state of the atom is chosen to represent a qubit, with the mapping of the ground state to $|0\rangle$ and of the first excited state to $|1\rangle$, there is still a much larger space on this system corresponding to the higher energy states. The ”well-characterization” requirement means that this system must not make a transition to any other dimension except the required ones, that is $|0\rangle$ and $|1\rangle$. Also, the selected physical system for the representation of the qubit must be easily expanded to a large number of qubits and additionally each qubit must be easily handled individually and independently.

2. Ability to initialize the state of the qubits to a simple fiducial state, e.g. $|00\ldots0\rangle$.

   This requirement comes due to the quantum circuit model definition which mandates the initialization of a quantum register or storage element to a definite state before any computation takes place. Except from the preparation of a quantum register to a definite state at the initialization of the computation, such a preparation is also required during the whole computation run by the error correcting procedures applied to the circuit.

3. Long relevant decoherence times, much longer than the gate operation time.

   Decoherence or quantum noise is the undesirable degradation of a quantum state, due to non unitary evolution, to a statistical mixture of states. A qubit is a very fragile entity, whatever its implementation may be, as it unavoidably interacts in an uncontrollable way with its environment and other neighboring qubits. The quantum circuit model assumes only unitary evolution (except at the final measurements) and thus it is an idealization of a real quantum circuit. In a real quantum circuit the evolution becomes non-unitary because the system can no longer be assumed a closed system.

The consecutive accumulation of errors in a real quantum circuit during the computation makes the final result useless. Decoherence increases as time evolves, leading to the notion of decoherence time which can be thought as the time that transforms a well defined state to a completely mixed state. This implies that long processing tasks would be impossible due to the accumulation of errors. Thus, it is important that the decoherence time is long enough so as at the end of the computation the accumulated decoherence in each timestep of computation does not degrade the final state excessively.

Fortunately, various quantum error correction codes have been invented which make the restoration of a distorted state of one or more qubits possible, at the expense of using redundancy [19, 21, 20]. Nevertheless, it is not obvious that the usage of error correcting codes can lead to fault-tolerant quantum gates because the entailed redundancy may introduce more errors during the error correction than the errors corrected. The incorporation of fault tolerance to quantum computation is achieved by (i) choosing a quantum error correcting code, (ii) applying directly on the codewords the quantum operations corresponding to the usual elementary gates like X,Z,CNOT, etc, as well as measurements and state preparations and (iii) applying error correction after each operation. The operations on the encoded states must be done without the errors spread on more qubits than the code can handle. As an example, when using the Steane $[[7,1,3]]$ code, one encoded logical qubit (quantum codeword) is represented by seven physical qubits. The basis states of a logical qubit are represented by the seven qubit states

$$\begin{aligned}|0\rangle_L = \frac{1}{\sqrt{8}}(&|0000000\rangle + |1010101\rangle + |0110011\rangle + |1100110\rangle \\ &+ |0001111\rangle + |1011010\rangle + |0111100\rangle + |1101001\rangle)\end{aligned} \tag{2.41}$$

and

$$\begin{aligned}|1\rangle_L = \frac{1}{\sqrt{8}}(&|1111111\rangle + |0101010\rangle + |1001100\rangle + |0011001\rangle + \\ &+ |1110000\rangle + |0100101\rangle + |1000011\rangle + |0010110\rangle),\end{aligned} \tag{2.42}$$

respectively, while a general superposition state of a logical qubit is $a|0\rangle_L + b|1\rangle_L$. The distance of this code is 3 and it can detect and correct any kind of error in any one of its seven qubits, but no more than one error. The correction on a codeword is achieved by suitable syndrome extraction circuits using ancilla qubits and measurements on them so as to orchestrate the recovery of the correct codeword. Special care must be taken so as the ancilla are prepared faithfully and the measurements are applied fault tolerantly without introducing new errors.

The computation is achieved by applying logical gates on each logical qubit on the condition that they are fault tolerant, that is they don't propagate an error appearing on anyone of the seven physical qubits to another physical qubit of the same block code, as this code cannot correct more than one error in a codeword. An example of a fault tolerant gate on this code is the X gate which can be implemented *transversally* by applying seven X gates on each physical qubit as $X_L = X \otimes X \otimes \cdots \otimes X$. Clearly, an error on anyone of the seven physical qubits cannot propagate to another qubit. Similar constructions can be accomplished other elementary gates as well as for the logical CNOT gate where seven CNOT qubit gates are applied qubit-wise between two seven qubit codewords. Not all members of a universal set of gates can

be implemented transversally. An example is the T gate where its fault tolerant implementation requires additional ancilla qubits specially initialized and verified and fault tolerant measurements. Thus, the T gate is counted as a more costly gate than the transversal gates H,X,Z,S and CNOT.

After the application of each logical fault tolerant gate, fault tolerant error correction is applied on each code block being part of the logical gate operation. Also, error correction must be applied periodically on idle logical qubits. The above coarsely described scheme of error correction can be applied recursively at a higher level leading to *concatenated coding*. E.g. seven logical qubits of the first level error correction are combined to offer a second level logical qubit consisting of $7^2 = 49$ physical qubits, etc, until an adequate probability of error can be reached for a particular computation.

An important result known as *Quantum Threshold Theorem* [22] gives the adequate condition for fault tolerant quantum computation. This may be simplified as follows [50]: A quantum computation of arbitrary duration can be done with arbitrary small error probability in the presence of noise and using imperfect quantum gates if the storage of qubits and the physical gates used to perform the computations have error probabilities less than a threshold value P_a known as the accuracy threshold. The higher the lower bound is for the accuracy threshold, the less challenging is the actual physical gate construction relative to induced decoherence . The exact lower value of this threshold depends on various assumptions for the error model, the error control codes used etc. Some improvements on the initially estimated lower bound for this threshold can be found in [81] giving a threshold of $2.73 \cdot 10^{-5}$ for the simple Steane $[[7,1,3]]$ 7 qubits - distance 3 code [21], or in [82] giving a threshold of $1.32 \cdot 10^{-3}$ for a Golay $[[23,1,7]]$ 23 qubits - distance 7 code.

Topological or surface codes suggest another promising method for fault tolerant quantum computation [83, 84]. Recent results [85] suggest an accuracy threshold of about $1.4 \cdot 10^{-2}$ for these kind of codes. An additional advantage of these codes is that they require local interaction only between the qubits in the block code, relaxing the communication cost requirement.

4. Ability to implement a universal set of quantum gates.

   In Section 2.3.5 the notion of a universal set of gates was introduced and it was also noted that a discrete set of gates supporting fault tolerance can be used to approximate any quantum gate with arbitrary accuracy. The chosen physical representation must support the operation of this set of gates and also implement them so as its reliability or *fidelity* (small probability of error) is under the accuracy threshold defined above.

   In general, interaction between two systems, each one representing a qubit, is more difficult to be controlled in a manner corresponding to a two-qubit gate than the control of a single qubit gate. This may pose restrictions to the architecture of the system, such as allowing local interactions only. Another restriction which may have important impact to fault tolerance and speed of the circuit is the ability of performing parallel operations of the gates.

5. A qubit-specific measurement capability.

   At the end of the quantum computation, classical results must be extracted by measuring particular qubits on a particular basis. Except from this, measurement is also

involved in the error correction used in the fault tolerance. For this reason the measurement must be reliable and fast.

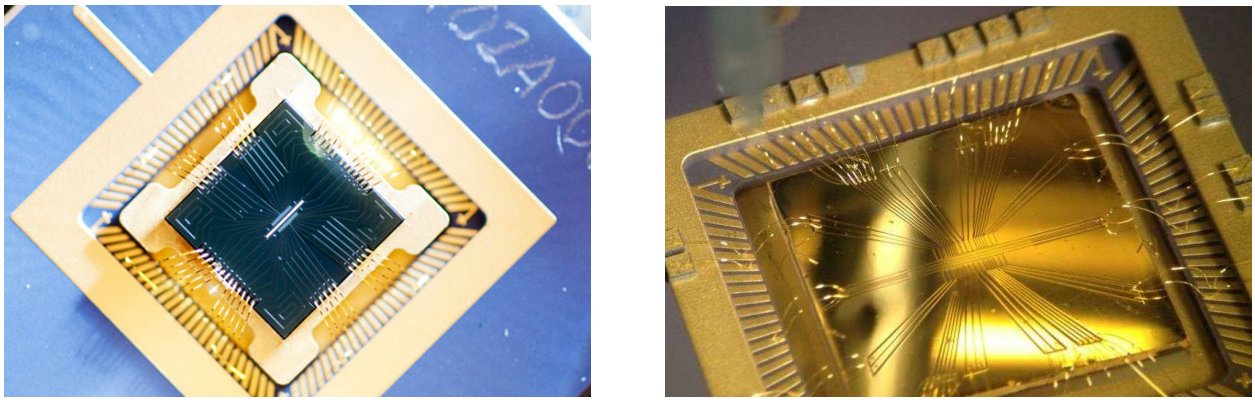
An additional requirement, as noted in [80] is reliable quantum communication from one place of the quantum computer to another. This could be accomplished with real transport of the physical carrier of the qubit or using quantum teleportation.

A particular technology cannot always sufficiently fulfill all the above requirements and in fact some of them may contradict each other. E.g. the requirement to build a two-qubit gate means that these two qubits can easily come to an interaction. On the other hand this means that they can be easily decohered. A striking example is the usage of photons which can very robustly represent a qubit but cannot easily be involved in a two-qubit gate construction.

In general, trade-offs between various architecture parameters must be accounted for in order to study whether the proposed architecture is feasible for a future quantum computer. These parameters may be the number of qubits available, the physical gates operation and decoherence times, error control codes used, the particular quantum algorithm implementation targeted, communication costs, total running time, etc. The interplay of all these parameters is complicated enough so as to make any analytical study on performance vs resource requirements of a candidate quantum computer on a particular physical implementation. For this reason various software tools have been developed to make feasibility studies [86, 87, 88].

A brief presentation of various technologies that have been proposed and tested as implementation candidates on various aspects follows:

Trapped Ions The physical carrier of quantum information is the energy level of charged atoms which are confined in a small area with the help of static electrical potential and radio frequency radiation. The ions are kept in an ultra-vacuum, ultra-cold environment to minimize their kinetic energy and interaction with other atoms. Laser pulses of appropriate wavelength and duration can initialize the state, perform one qubit gate operation and measure a particular ion. The ions can form one dimensional chain and the Coulomb interaction between them combined with laser pulses can be exploited for a two-qubit gate implementation as first proposed in [13]. Recent experiments [91] have shown that ion traps can offer excellent fi-



(i) (ii)

Figure 2.14: Ions Trap microfabricated chips: (i) Sandia National Laboratories. [89], (ii) Department of Physics, Oxford University. [90]

delity of initialization and measurement with probability of error $7 \cdot 10^{-4}$, single qubit gate operation with error probability 10^{-6}, while the two-qubit gate is implemented with an error of about 10^{-2}.

In the last 20 years extensive experimentation with trapped ions has delivered promising results concerning the scalability of these systems. Entanglement of six qubits was observed in [92] and [93]. Trapped ions were demonstrated on semiconductor fabrication MEMS technology in [94]. Universal quantum simulation with ion traps of six qubits was demonstrated in [95]. Shor's algorithm with three qubits (Kitaev's version) experimentally worked on ion traps [96]. Entanglement of 14 qubits was achieved in [97]. Number 15 was factored in its primes using 11 qubits as reported in [15].

Superconducting qubits

A Josephson junction is formed when two superconductors are coupled with a thin insulator. Current can flow through the insulator in the absence of any applied voltage, while microwave current oscillations, whose frequency depends on the voltage applied, appear. Resonant microcircuits (LC) are formed with their inductances (L) implemented as superconductors with Josephson junctions. These microcircuits behave as individual atoms in the sense that their stored energy is quantized representing thus a superconducting qubit [14]. Microwave pulses can alter the quantum state and electrical measurements are used to perform quantum measurements.

Universal set of high fidelity quantum gates based on Josephson junctions reported in [98]. Probability of error $8 \cdot 10^{-4}$ for single qubit gates and $6 \cdot 10^{-3}$ for two-qubit gates were measured. Also, entanglement on five qubits was demonstrated. Together with the ability to prepare and measure a state with high fidelity, these results show that combined with the usage of surface codes, this technology is viable for large scale quantum computation.

In [99], Shor's algorithm was implemented using three Josepshon junctions qubits and succesfully factored integer 15. Simulation of fermionic systems was reported in [16] using four qubits.

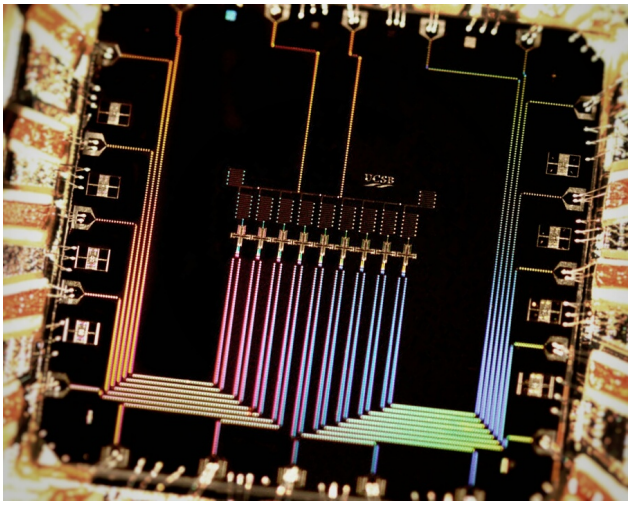


Figure 2.15: Nine superconducting qubits integrated circuit fabricated at University of California - Santa Barbara [100].

Optical Photons

Photons of optical wavelengths can be another qubit carrier. The polarization of the photons or their path can represent the qubit value. Mirrors, beam splitters, phase shifter crystals etc. can be used to implement any single qubit gate. Photons are very robust qubit carriers as they do not have strong interactions between them and with matter. Yet, this is also a drawback, as the construction of two-qubit gates becomes difficult. Indirect interaction between photons could be achieved with Kerr crystals to build two-qubit gates, but this method leads to low fidelity gates. Another problem faced with photon implementations for quantum computation is the requirement for single photon sources for state preparation and single photon detectors with high efficiency for the measurement.

A scheme that uses only linear optics (Kerr crystals are non-linear) and teleportation is presented in [101]. The linear optics implementation proposal requires single photon detection on demand and high efficiency single photon detectors.

Shor's algorihm was implemented to factor integer 15 with linear optics and using four qubits in [102] and [103]. A similar experimental setup was demonstrated in an integrated silicon photonic chip [17]. Integer 21 was factored in [104] using only one qubit and a qutrit (a qutrit operates on three dimensions instead of two).

Nuclear Magnetic Resonance

Nuclear magnetic resonance manipulation is a well established method to direct manipulate nuclear spins using magnetic fields and radio frequency radiation. Due to the tiny strength of the inherent magnetic field produced by an individual nucleon, a sample of many molecules is used, in contrast to the case of trapped ions. Every molecule in the sample is a tiny quantum computer and each average behavior is manipulated. The interactions between the nuclei in each molecule are exploited to define the quantum gates. Effectively, the architecture of the quantum computer is hardwired in a molecule. The drawback here is that this scheme does not scale well. Trivial implementation of Grover's algorithm using NMR was shown in [105]. An implementation of Shor's algorithm using seven qubits was reported in [18].

These were but a few proposals to real quantum computation. Many other proposals are under investigation (cavity quantum electrodynamics, neutral atoms etc. [106, 107]). Only recently a two-qubit CNOT gate was demonstrated using quantum dots in conventional silicon fabrication [108].

Another approach to computation and simulation is the adiabatic quantum computation [109]. It is a special purpose method which solves specific global optimization problems using quantum annealing. Quantum annealing is computationally more efficient than classical simulated annealing.

D-Wave Systems announced in 2011 a product which was claimed to be the first commercial quantum computer. In fact it is a system that performs modest quantum annealing and it is not a programmable quantum circuit model computer. The first computer named D-Wave One was a 128 qubits computer (sold to Lockheed Martin). In 2013, D-Wave Two version was released incorporating 512 qubits (sold to Google) and in 2015 more than 1000 qubits were announced for the D-Wave 2X product. Superconducting qubits are used in all three products but without any fault tolerance incorporated.

Criticism has been raised concerning D-Wave Systems machines, in two aspects, in relation with each other. The first is whether these machines are real quantum computers in the sense that entanglement, which is believed to be a prerequisite for quantum computation, is observed. A study [110] showed entanglement in groups of eight local qubits, but not throughout the machine, and this is due to the low fidelity of the qubit operations and the lack of error correction. The other question is whether any quantum speed-up is offered. While some studies showed that D-Wave outperforms classical solution (but in constant factors) [111], others showed no real quantum speed-up [112, 113]. It is unclear yet if any of the machine's outperformance to solve some problems instances can be attributed to quantum entanglement as this entanglement is not global.

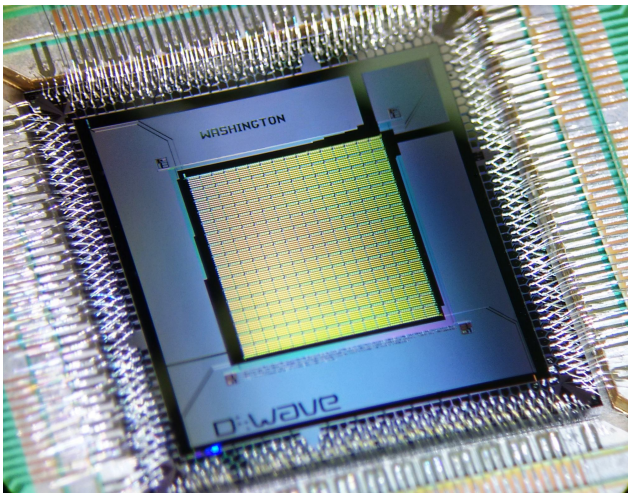


Figure 2.16: D-Wave Systems 1000 qubits quantum annealing processor.

The progress achievements of the last years, both in theory and experimentally, bring closer the possibility of the appearance of practical quantum computers in the near future (at least for simulation purposes), something that a few years ago was not surefooted. Still, much effort remains to be devoted.

3. SHOR'S ALGORITHM

While Simon's problem [9] was the first one that manifested a super-polynomial speed-up of a quantum computer relative to a classical one, it had no practical applications. At the same time, Peter Shor inspired by Simon's work, introduced a quantum algorithm that offered super-polynomial speed-up and had also important applications [10], [114]. Shor's algorithm deals with the problems of integer factorization and discrete logarithm. Both of them find applications in the cryptanalysis domain that is to decrypt an encrypted message without prior knowledge of the decryption key. The security of many public key cryptography systems [115] rely on the assumption that the integer factorization and discrete logarithm problems are intractable for a classical computer. Shor's quantum algorithm, combined with classical pre-processing and post-processing, achieves to solve these problems in polynomial time with respect to the problem size, which is the bits length of the number to be factored. In the following subsections we describe the algorithm in detail.

3.1 Preprocessing: Reduction of Factoring to Period Finding

The factorization problem of a composite odd integer N, which is not a prime power, can be reduced to a period finding problem (see Appendix A.1 for proofs). Namely, given the integer N, a random integer a is chosen such that $1 < a < N$ and the greatest common divisor $\gcd(a, N)$ is calculated. If $\gcd(a, N) \neq 1$ then, by the definition of GCD, a factor $p = \gcd(a, N)$ of N is found and the algorithm finishes. If on the other hand $\gcd(a, N) = 1$ holds, then the smallest period r of the sequence $f(x) = a^x \bmod N$ is calculated. If r happens to be even and also $a^{\frac{r}{2}+1} \bmod N \neq 0$, then the a factor of N is $p = \gcd(a^{\frac{r}{2}} - 1, N)$ and the algorithm finishes with success. If r is not even or $a^{\frac{r}{2}+1} \bmod N = 0$, the previous steps are repeated by choosing another random a.

The above steps are shown compactly in Table 3.1. The subroutines used in the algorithm are $\gcd$, modular exponentiation and the period finding of a sequence $\mathrm{FindPeriod}$. Euclid's algorithm is a polynomial complexity algorithm which can be used for the calculation of the greatest common divisor. The modular exponentiation can also be calculated with polynomial complexity. The algorithm of Table 3.1 successfully finds factors of N with a

Table 3.1: Probabilistic algorithm to factorize an odd, non prime power integer, N by finding the period of sequence $a^x \bmod N$. The output p is a factor of N. The probability of success can be made arbitrary close to 1 with a constant number of iterations.

1:	INPUT N
2:	$a = \mathrm{Random}(1, \ldots, N-1)$
3:	$g = \gcd(a, N)$
4:	IF $g \neq 1$ THEN
5:	$p = g$
6:	ELSE
7:	$r = \mathrm{FindPeriod}(a^x \bmod N)$
8:	IF $r \bmod 2 = 0$ AND $a^{\frac{r}{2}+1} \bmod N \neq 0$ THEN
9:	$p = \gcd(a^{\frac{r}{2}} - 1, N)$
10:	ELSE
11:	GOTO LINE 2
12:	END IF
13:	END IF
14:	OUTPUT p

probability arbitrary close to 1 with constant number of iterations, on the condition N is odd and not a prime power, that is not of the form $N = p^m$ where p is a prime (this case can be solved efficiently using a classical algorithm). Thus, the presented factorization algorithm can factorize an integer efficiently on the condition that the period finding routine is efficient. There is not known efficient classical algorithm for the period finding procedure, yet an efficient quantum algorithm for the period finding can be used instead and thus the problem of factoring can be solved efficiently on a quantum computer. This is the contribution of Shor's algorithm.

3.2 Quantum Fourier Transform

The *Quantum Fourier Transform* (QFT) of size L is a linear transformation from $\mathcal{H}^L$ to itself. It transforms a general superposition state $|\psi\rangle = x_0|0\rangle + x_1|1\rangle + \cdots + x_{L-1}|L-1\rangle$ to another state $|\varphi\rangle = y_0|0\rangle + y_1|1\rangle + \cdots + y_{L-1}|L-1\rangle$ according to

$$|\psi\rangle \xrightarrow{QFT} |\varphi\rangle = \sum_{k=0}^{L-1} \left(\frac{1}{\sqrt{L}} \sum_{j=0}^{L-1} x_j \omega_L^{jk} \right) |k\rangle \tag{3.1}$$

where $\omega_L = e^{i\frac{2\pi}{L}}$ is the L-th primitive root of unity. The parenthesis in Eq. (3.1) is the amplitude y_k of the transformed state $|\varphi\rangle$, that is

$$y_k = \frac{1}{\sqrt{L}} \sum_{j=0}^{L-1} x_j \omega_L^{jk} \tag{3.2}$$

and this is a modified definition[1] of the *Inverse Discrete Fourier Transform* (IDFT) applied on the $\{x_j\}$ sequence; $y_k = \mathrm{IDFT}\{x_j\}$.

The transform in Eq. (3.1) is unitary and its inverse is given by

$$|\varphi\rangle \xrightarrow{QFT^{-1}} |\psi\rangle = \sum_{j=0}^{L-1} \left(\frac{1}{\sqrt{L}} \sum_{k=0}^{L-1} y_k \omega_L^{-jk} \right) |j\rangle \tag{3.3}$$

Similarly, the amplitudes of $|\psi\rangle$ superposition are given by the *Discrete Fourier Transform* (DFT) of the $\{y_k\}$ sequence; $x_j = \mathrm{DFT}\{y_k\}$ which is given by

$$x_j = \frac{1}{\sqrt{L}} \sum_{k=0}^{L-1} y_k \omega_L^{-jk} \tag{3.4}$$

Thus, the QFT and its inverse are essentially IDFT and DFT applied on the amplitudes of a superposition. When working with n qubits, the size L of the QFT is $L = 2^n$. Then, any computational basis state $|j\rangle, j = 0 \ldots 2^n - 1$ can be written as $|j\rangle = |j_1\rangle|j_2\rangle \cdots |j_n\rangle$ where $(j_1 j_2 \ldots j_n)$ is the binary representation of j, that is $j = j_1 2^{n-1} + j_2 2^{n-2} + \cdots + j_n 2^0$. It can be proven (see Appendix A.4) that the application of QFT to any basis state $|j\rangle$ can be described as the n qubits product state [116]

[1] Usually, in Signal Processing literature the DFT is defined by $\mathrm{DFT}\{x_j\} = \frac{1}{L}\sum_{j=0}^{L-1} x_j \omega_L^{-jk}$, while the IDFT is defined by $\mathrm{IDFT}\{y_k\} = \sum_{j=0}^{L-1} y_k \omega_L^{jk}$. In the Quantum Computation context the pairs of Eq. (3.4) and (3.2) are used instead, mainly due to the amplitude normalization requirement; the initial state $|\psi\rangle$ and the transformed state $|\varphi\rangle$ must have unit norm.

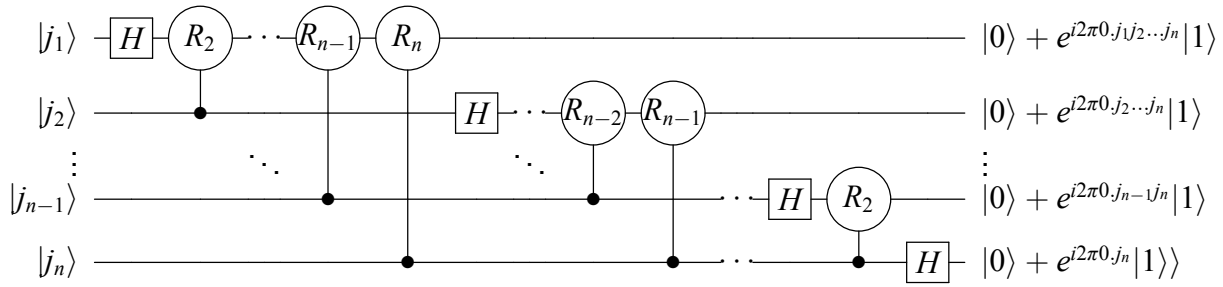


Figure 3.1: Quantum Fourier Transform circuit on n qubits. The normalization factor $\frac{1}{\sqrt{2}}$ is not shown at the output states. The order of the qubits must be reversed at the end.

$$|j\rangle \xrightarrow{QFT} \frac{1}{\sqrt{2^n}} \left(|0\rangle + e^{i2\pi(0.j_n)}|1\rangle\right) \left(|0\rangle + e^{i2\pi(0.j_{n-1}j_n)}|1\rangle\right) \cdots \left(|0\rangle + e^{i2\pi(0.j_1 j_2 \ldots j_n)}|1\rangle\right) \tag{3.5}$$

In Eq. (3.5) the binary notation $(0.j_l j_{l+1} \ldots j_n)$ is used to represent the fractional number $\frac{j_l}{2} + \frac{j_{l+1}}{4} + \cdots + \frac{j_n}{2^{n-l+1}}$.

The above product form representation helps to construct a circuit (see also Appendix A.4) which computes the QFT on n qubits [117] as shown in Figure 3.1. The controlled R_k gates ($k = 2 \ldots n$) depicted are c-$R_z(\theta)$ gates, where the abbreviation $R_k = R_z(\frac{2\pi}{2^k})$ is used. Observe that the output product state of the circuit is given in reverse order, e.g. the state $|0\rangle + e^{i2\pi(0.j_n)}|1\rangle$ results at the bottom qubit instead of the top one. A reordering of the qubits using $\frac{n}{2}$ SWAP gates is not shown in this figure.

The circuit of the inverse QFT can be easily built by mirroring horizontally (flipping about the vertical axis) the whole QFT circuit (including the SWAP gates) and using opposite signs in the c-$R_z(\theta)$ gates.

3.3 Discrete Fourier Transform and Periods

We define the *comb sequence* of finite length L and period r as

$$\mathrm{Comb}_{r,L}(j) \doteq \sum_{n=0}^{\lfloor L/r \rfloor - 1} \delta(j - nr), \quad 0 \leq j \leq L-1 \tag{3.6}$$

where $\delta(j)$ is the impulse sequence defined by

$$\delta(j) = \begin{cases} 1, & j = 0 \\ 0, & j \neq 0 \end{cases} \tag{3.7}$$

Proposition 3.1. *The DFT of a comb sequence of length L and period r, when r divides L, is another comb sequence of length L and period $\frac{L}{r}$. That is,* $\mathrm{DFT}\{Comb_{r,N}(j)\} = Comb_{\frac{N}{r},N}(k)$

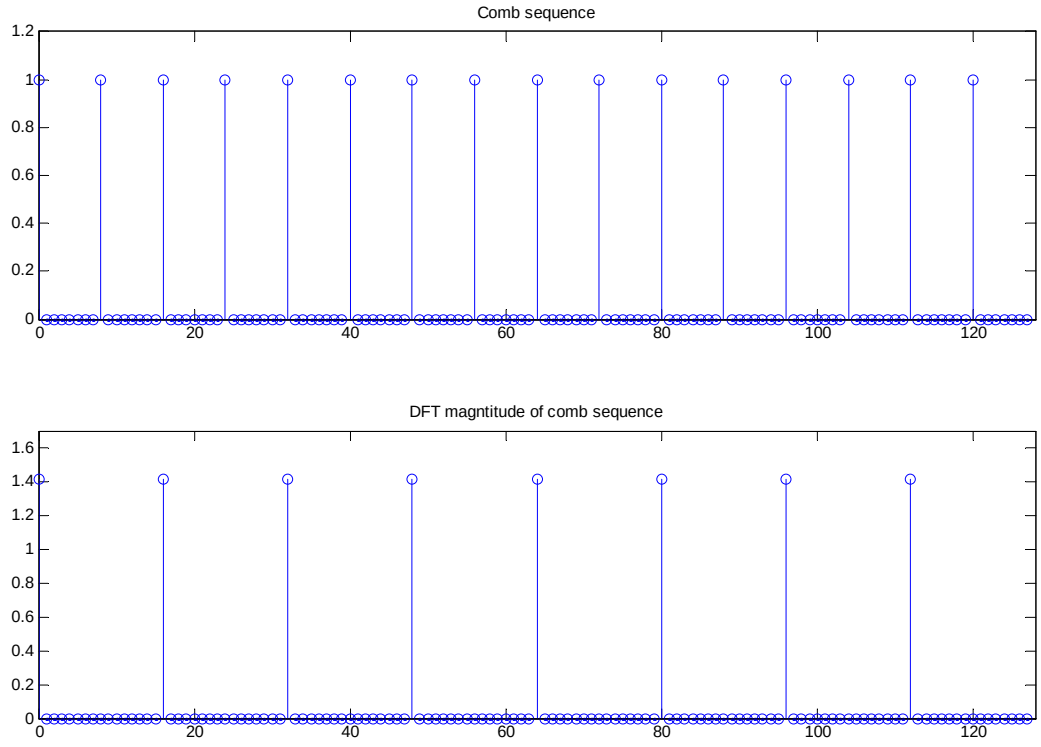


Figure 3.2: Comb sequence of length $L = 128$ and period $r = 8$ (top), magnitude of the corresponding DFT (bottom).

Proof.

$$
\begin{aligned}
y_k &= \mathrm{DFT}\{\mathrm{Comb}_{r,L}(j)\} \\
&= \frac{1}{\sqrt{L}} \sum_{j=0}^{L-1} \sum_{n=0}^{L/r-1} \delta(j - nr)\omega_L^{-jk} \\
&= \frac{1}{\sqrt{L}} \sum_{n=0}^{L/r-1} e^{-i\frac{2\pi rnk}{L}} \\
&= \begin{cases} \frac{1}{\sqrt{L}} \sum_{n=0}^{L/r-1} 1, & rk \bmod L = 0 \\ \frac{1}{\sqrt{L}} \sum_{n=0}^{L/r-1} e^{-i2\pi nl} & rk \bmod L = l \neq 0 \end{cases} \\
&= \begin{cases} \frac{\sqrt{L}}{r}, & rk \bmod L = 0 \\ 0, & rk \bmod L \neq 0 \end{cases} \\
&= \frac{\sqrt{L}}{r} \sum_{n=0}^{r-1} \delta(k - n\frac{L}{r}) \\
&= Comb_{\frac{L}{r},L}(k)
\end{aligned}
\tag{3.8}
$$

□

In other words, the impulses of the DFT appear at integer multiples of L/r, when r divides exactly the length L of the initial comb sequence. These impulses have equal values (as well as absolute values or magnitudes) $\frac{\sqrt{L}}{r}$.

A comb sequence of length $L = 128$ and period $r = 8$ and its respective DFT magnitude is depicted in Figure 3.2. The DFT has a period of $L/r = 16$ and the magnitude of the impulses occuring at integer multiples of 16 is $|\frac{\sqrt{L}}{r}| \approx 1.414$.

If someone knows the index k of any DFT peak, then together with the knowledge of L, he may possibly find the period r. From Eq. (3.8), this index is of the form $k = \frac{L}{r}n$ for $n = 0 \ldots r-1$. Then, $\frac{k}{L} = \frac{n}{r}$. Reducing the fraction $\frac{k}{L}$ into its lower terms p and q so as $\frac{k}{L} = \frac{p}{q}$ leads to $q = \frac{L}{\gcd(k,L)}$ and thus the period is $r = q = \frac{L}{\gcd(k,L)}$. This is valid on the condition that $\frac{n}{r}$ is already expressed into its lower terms or equivalently that $\gcd(n, r) = \gcd(\frac{rk}{L}, r) = 1$. In the example of Figure 3.2 the knowledge of $k = 48$ (fourth peak) leads to the correct estimation $\frac{128}{\gcd(48,128)} = \frac{128}{16} = 8$ for the period because it holds $\gcd(\frac{8 \cdot 48}{128}, 8) = 1$. But when $k = 32$, an incorrect estimation $r = \frac{128}{\gcd(32,128)} = \frac{128}{32} = 4$ is extracted because $\gcd(\frac{8 \cdot 32}{128}, 8) = 2 \neq 1$.

In the more general case where the period r of a comb sequence does not fit exactly its length L (r does not divide L), its DFT still shows peaks at about integer multiples of the ratio L/r, however the overall pattern is somehow distorted. The DFT for this case follows by defining $s = \lfloor L/r \rfloor$

$$\begin{aligned} y_k &= \mathrm{DFT}\{\mathrm{Comb}_{r,L}(j)\} = \\ &= \frac{1}{\sqrt{L}} \sum_{j=0}^{L-1} \sum_{n=0}^{s-1} \delta(j - nr) \omega_L^{-jk} \\ &= \frac{1}{\sqrt{L}} \sum_{n=0}^{s-1} \omega_L^{-nrk} \\ &= \frac{1}{\sqrt{L}} \frac{\omega_L^{-rks} - 1}{\omega_L^{-rk} - 1} \\ &= \frac{1}{\sqrt{L}} \frac{e^{\frac{-i2\pi rk}{L} \lfloor L/r \rfloor} - 1}{e^{\frac{-i2\pi rk}{L}} - 1} \end{aligned} \tag{3.9}$$

It is easy to see that the above DFT reduces to Eq. (3.8) if $r|L$. Figure 3.3 depicts a comb sequence as defined in Eq. (3.6) with length $L = 128$ and period $r = 6$. In the same figure the magnitude of its DFT is shown. Although r does not divide L, large amplitudes are concentrated close to integer multiples of $L/r = 21.33 \ldots$ and this concentration can be exploited to infer the period r as it will be shown later.

It is useful to define a slightly modified version of the comb sequence in Eq. (3.6) so as to contain one more impulse :

$$\begin{aligned} \mathrm{Comb}'_{r,L}(j) &\doteq \mathrm{Comb}_{r,L}(j) + \delta(\lfloor L/r \rfloor r), \quad 0 \leq j \leq L-1 \\ &= \sum_{n=0}^{\lfloor L/r \rfloor} \delta(j - nr), \quad 0 \leq j \leq L-1 \end{aligned} \tag{3.10}$$

Taking into account that $DFT\{\delta(j - \lfloor L/r \rfloor r)\} = \frac{1}{\sqrt{L}} \omega^{-rsk}$, where again $s = \lfloor L/r \rfloor$, and using Eq. (3.9), the DFT of this modified comb sequence can be found

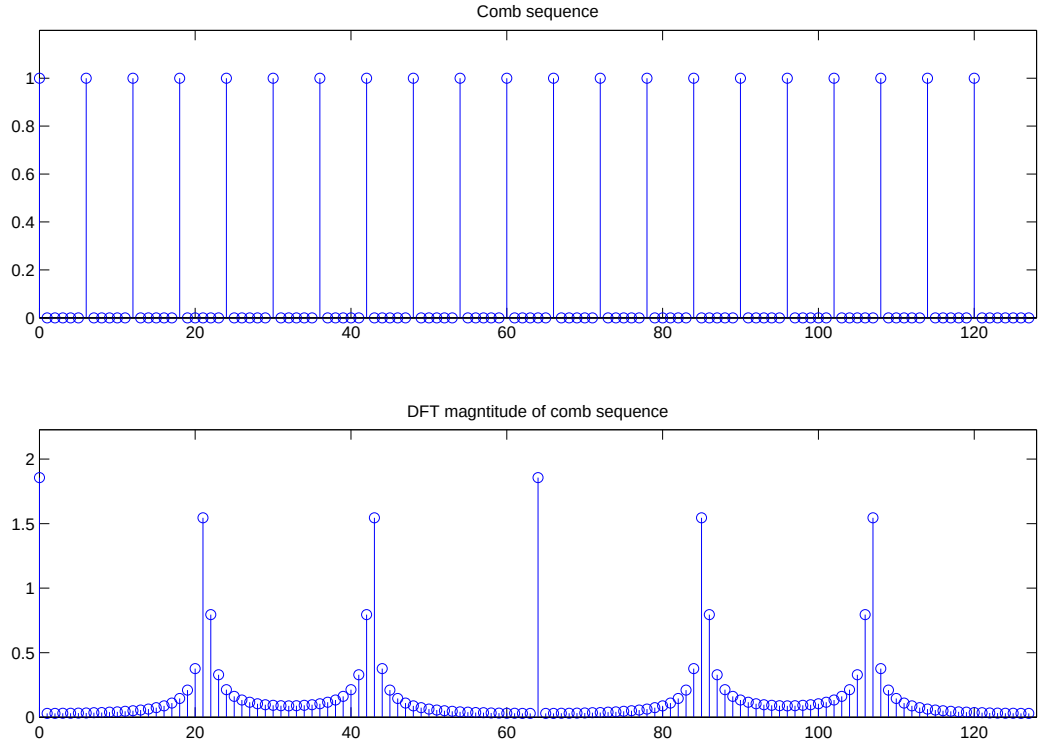


Figure 3.3: Comb sequence of length $L = 128$ and period $r = 6$ (top), DFT of magnitude sequence (bottom).

$$\begin{aligned} y_k &= \mathrm{DFT}\{\mathrm{Comb}'_{r,L}(j)\} \\ &= \frac{1}{\sqrt{L}} \frac{\omega_L^{-rks} - 1}{\omega_L^{-rk} - 1} + \frac{1}{\sqrt{L}} \omega^{-rsk} \\ &= \frac{1}{\sqrt{L}} \frac{\omega^{-rsk} - 1 + \omega^{-rk}\omega^{-rsk} - \omega^{-rsk}}{\omega^{-rk} - 1} \\ &= \frac{1}{\sqrt{L}} \frac{\omega_L^{-r(s+1)k} - 1}{\omega_L^{-rk} - 1} \end{aligned} \tag{3.11}$$

The pair of the modified comb sequence and its DFT is shown in Figure 3.4. The DFT of this sequence is slightly different from the one of Figure 3.3, but the qualitative observations about the location of the peaks still holds. Namely, the peaks appear near integer multiples of the ratio L/r, except that all the terms of the derived DFT sequence are up-level shifted by an amount $\frac{1}{\sqrt{L}} \approx 0.088$ because of the extra term $\delta(\lfloor L/r \rfloor r)$ in Eq. (3.10).

A useful property of the DFT is its magnitude invariance under cyclic shift. If x_j is any sequence of length L, then the same sequence shifted by an amount λ is defined to be $x_{(j-\lambda) \bmod L}$. It can be proven that

$$\mathrm{DFT}\{x_{(j-\lambda) \bmod L}\} = e^{\frac{-i2\pi\lambda}{L}} \mathrm{DFT}\{x_j\} \tag{3.12}$$

and consequently the DFT magnitudes of the original and the shifted sequence are the same; $|DFT\{x_{(j-\lambda) \bmod L}\}| = |DFT\{x_j\}|$

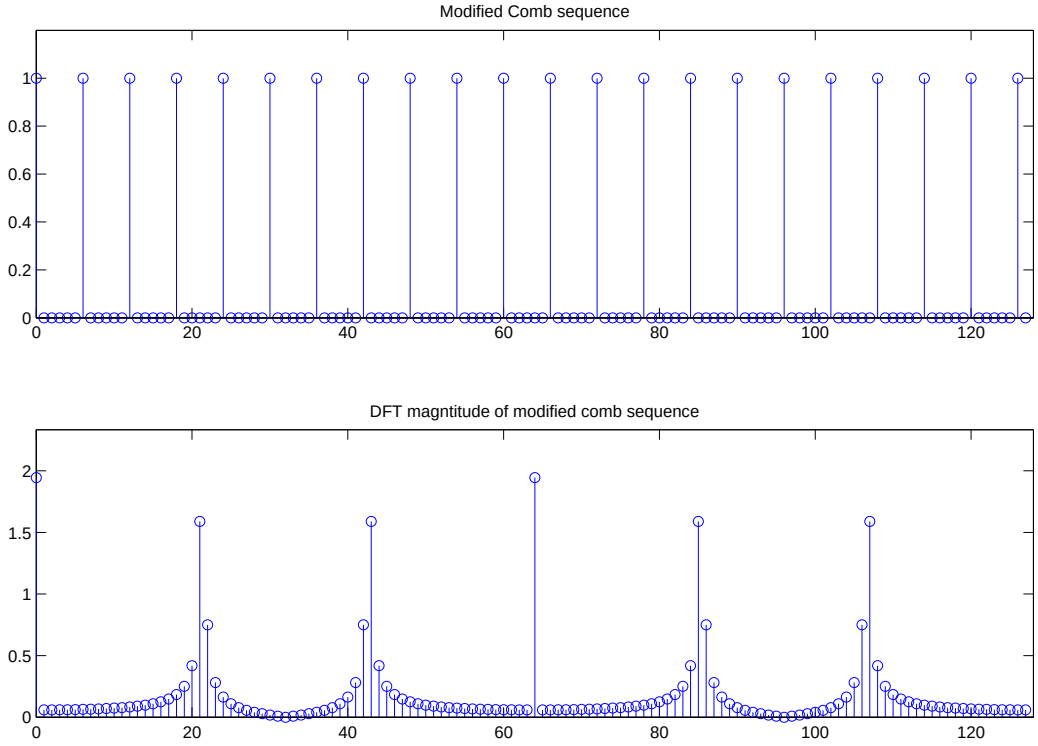


Figure 3.4: Modified Comb sequence of length $L = 128$ and period $r = 6$ (top), DFT of magnitude sequence (bottom).

A more general definition of a comb sequence of length L and period r that takes into account possible shifts by an amount λ in the range $0 \ldots r-1$ is given below where $t = L \bmod r$

$$\begin{aligned} \mathrm{Comb}_{r,L,\lambda}(j) &\doteq \sum_{n=0}^{s-1} \delta(j - nr - \lambda) + \delta(j - nr - \lambda) \\ &= \begin{cases} \mathrm{Comb}'_{r,L}((j-\lambda) \bmod L), & 0 \le \lambda \le t-1 \\ \mathrm{Comb}_{r,L}((j-\lambda) \bmod L), & t \le \lambda \le r-1 \end{cases} \end{aligned} \tag{3.13}$$

Thus, the generalized definition of the comb sequence is the cyclic shift by λ of the comb sequences of Eq. (3.6) or Eq. (3.10), depending on the value of λ. The DFT of the generalized comb of Eq. (3.13) can be obtained by combining Eq. (3.9) and Eq. (3.11) with the DFT shift property of Eq. (3.12)

$$y_k = \mathrm{DFT}\{\mathrm{Comb}_{r,L.\lambda}(j)\} = \begin{cases} \frac{1}{\sqrt{L}} \omega^{-\lambda k} \frac{\omega_L^{-r(s+1)k} - 1}{\omega_L^{-rk} - 1}, & 0 \le \lambda \le t-1 \\ \frac{1}{\sqrt{L}} \omega^{-\lambda k} \frac{\omega_L^{-rks} - 1}{\omega_L^{-rk} - 1}, & t \le \lambda \le r-1 \end{cases} \tag{3.14}$$

Thus, the magnitude of the DFT of a generalized comb sequence of length $L = 128$ and period $r = 6$ for a shift of $\lambda = 0, 1$ is the one depicted in Figure 3.4, while for a shift of $\lambda = 2 \ldots 5$ is depicted in Figure 3.3.

The reason for the definition of the generalized comb sequence of Eq. (3.13) is that its DFT of Eq.(3.14) corresponds to superposition amplitudes of a quantum register just before the measurement in the quantum part of the order finding algorithm and this fact will be used in the next section.

3.4 Quantum Period Estimation

The purpose of the quantum part of Shor's algorithm is to find efficiently the period r of the modular exponentiation sequence $f(x) = a^x \bmod N$, $x \in \mathbb{N}$, for given integers N and a. Quantum superposition is exploited to compute $f(x)$ in a superposition of x values, and then interference using QFT is used to find the period with a constant probability.

A high level quantum circuit for the period finding problem is shown in Figure 3.5. The bottom (target) register $|\psi_B\rangle$ consists of n qubits, while the top (control) register $|\psi_T\rangle$ consists of $2n$ qubits, where $n = \lceil \log_2 N \rceil$. The U_f quantum subcircuit acts on both registers and computes the modular exponentiation function as follows

$$U_f(|x\rangle|z\rangle) = |x\rangle|za^x \bmod N\rangle \tag{3.15}$$

Initially, the two registers are in the states $|0\rangle$ and $|1\rangle$. The first step is to apply $2n$ Hadamard gates to each qubit of the top register and obtain the superposition

$$|\psi_T\rangle_1 = \frac{1}{\sqrt{L}} \sum_{x=0}^{L-1} |x\rangle \tag{3.16}$$

for $L = 2^{2n}$. The same state could be obtained by applying the forward QFT because of the particular initial state $|0\rangle$. Thus, a symmetry on Figure 3.5 could be observed. The state of the bottom register remains $|\psi_B\rangle_1 = |1\rangle$ and the application of the U_f block on the joint state of both registers results in

$$\begin{aligned} |\psi_T\rangle_2|\psi_B\rangle_2 &= U_f\left(\frac{1}{\sqrt{L}} \sum_{x=0}^{L-1} |x\rangle|1\rangle\right) \\ &= \frac{1}{\sqrt{L}} \sum_{x=0}^{L-1} |x\rangle|f(x)\rangle \\ &= \frac{1}{\sqrt{L}} \sum_{x=0}^{L-1} |x\rangle|a^x \bmod N\rangle \end{aligned} \tag{3.17}$$

The above equation can be reformulated, exploiting the periodicity property of f, in a form which is more convenient for the application of the QFT at next step. The running index $0 \le x \le L-1$ can substituted by two indices $m = 0 \ldots r-1$ and $l = 0 \ldots s-1$ as $x = lr + m$, where r is the period of f, $s = \lfloor L/r \rfloor$ and $t = L \bmod r$. Then Eq. (3.15) becomes

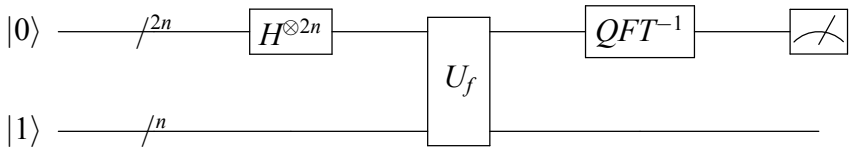


Figure 3.5: Quantum circuit for period finding algorithm

$$
\begin{aligned}
|\psi_u\rangle_2|\psi_l\rangle_2 =& \frac{1}{\sqrt{L}}\left(\sum_{m=0}^{r-1}\sum_{l=0}^{s-1}|lr+m\rangle|f(lr+m)\rangle + \sum_{m=0}^{t-1}|sr+m\rangle|f(sr+m)\rangle\right) = \\
& \frac{1}{\sqrt{L}}\left(\sum_{m=0}^{r-1}\sum_{l=0}^{s-1}|lr+m\rangle|f(m)\rangle + \sum_{m=0}^{t-1}|sr+m\rangle|f(m)\rangle\right) = \\
& \frac{1}{\sqrt{L}}\sum_{m=0}^{r-1}\sum_{x=0}^{L-1}\left(\sum_{l=0}^{s-1}\delta(x-lr-m)+\delta(x-sr-m)\right)|x\rangle|f(m)\rangle = \\
& \frac{1}{\sqrt{L}}\sum_{m=0}^{r-1}\sum_{x=0}^{L-1}\mathrm{Comb}(x)_{r,L,m}|x\rangle|f(m)\rangle
\end{aligned}
\tag{3.18}
$$

The second line of the above equation is due to the periodicity of f. The third line is derived because the superposition amplitudes of the top register are zero except for the states $|lr+m\rangle$ and $|sr+m\rangle$. The last line results by the definition of the generalized comb sequence given in Eq. (3.13).

The next step, which is the applications of the inverse QFT on the top register, can be derived by exploiting Eq. (3.14) which gives the DFT of the generalized comb sequence. Thus, the joint state of both registers after this step is

$$
\begin{aligned}
|\psi_T\rangle_3|\psi_B\rangle_3 =& (QFT^{-1}\otimes I)\frac{1}{\sqrt{L}}\sum_{m=0}^{r-1}\sum_{x=0}^{L-1}\mathrm{Comb}(x)_{r,L,m}|x\rangle|f(m)\rangle = \\
& \frac{1}{\sqrt{L}}\sum_{m=0}^{r-1}QFT^{-1}\left(\sum_{x=0}^{L-1}\mathrm{Comb}(x)_{r,L,m}|x\rangle\right)|f(m)\rangle = \\
& \frac{1}{L}\sum_{k=0}^{L-1}\left(\sum_{m=0}^{t-1}\omega^{-mk}\frac{\omega_L^{-r(s+1)k}-1}{\omega_L^{-rk}-1}+\sum_{m=t}^{r-1}\omega^{-mk}\frac{\omega_L^{-rks}-1}{\omega_L^{-rk}-1}\right)|k\rangle|f(m)\rangle
\end{aligned}
\tag{3.19}
$$

The last step is the measurement of the top register in the computation basis $\{|k\rangle,\, k = 0\ldots L-1\}$. The superposition amplitudes $c_{k,m}$ of both registers in Eq. (3.19) are

$$
c_{k,m} = \begin{cases} \frac{1}{L}\omega^{-mk}\frac{\omega_L^{-r(s+1)k}-1}{\omega_L^{-rk}-1}, & m = 0\ldots t-1 \\ \frac{1}{L}\omega^{-mk}\frac{\omega_L^{-rks}-1}{\omega_L^{-rk}-1}, & m = t\ldots r-1 \end{cases}
\tag{3.20}
$$

and consequently the probability to obtain the result k is $\mathrm{Prob}\,[k] = \sum_{m=0}^{r-1}|c_{k,m}|^2$. This probability can be calculated analytically by taking into account the trigonometric identity $|e^{i\theta}-1| = 4\sin^2(\frac{\theta}{2})$

$$
\begin{aligned}
\mathrm{Prob}\,[k] =& \sum_{m=0}^{t-1}\left|\frac{1}{L}\omega^{-mk}\frac{\omega_L^{-r(s+1)k}-1}{\omega_L^{-rk}-1}\right|^2+\sum_{m=t}^{r-1}\left|\frac{1}{L}\omega^{-mk}\frac{\omega_L^{-rks}-1}{\omega_L^{-rk}-1}\right|^2 = \\
& \frac{t\sin^2\left(\frac{\pi rk}{L}(s+1)\right)+(r-t)\sin^2\left(\frac{\pi rks}{L}\right)}{L^2\sin^2\left(\frac{\pi rk}{L}\right)}
\end{aligned}
\tag{3.21}
$$

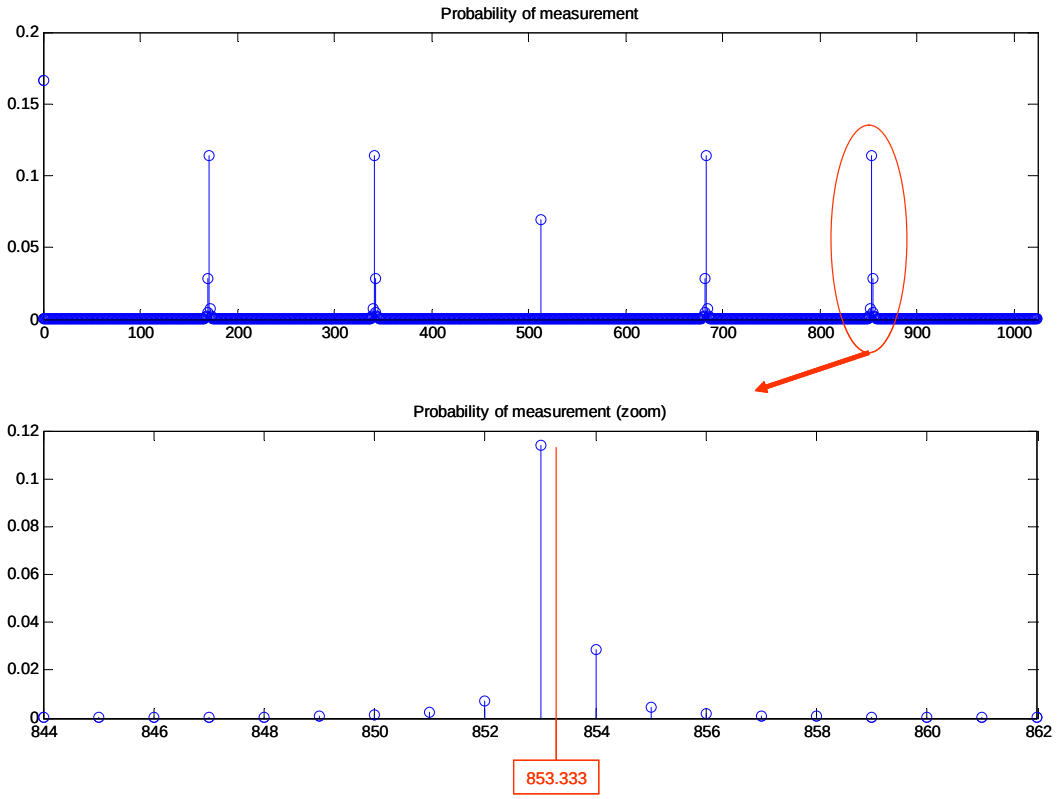


Figure 3.6: Measurement probabilities of period finding algorithm for $L = 1024$ and period $r = 6$ (top), zoom in the range $k = 844 \ldots 862$ (bottom).

As it was in the case of the DFT amplitudes of modified comb sequences, the probability is high for indices k that are close to integer multiples of L/r. If such a k is obtained by a measurement, then together with the knowledge of the L value, the period r can be found with constant probability of success using the method of *continued fraction expansion* (CFE) which is explained in section 3.5 and further analyzed in Appendices A.2 and A.3.

For the special case where r divides L the probabilities are

$$Prob\,[k] = \frac{1}{r}\sum_{n=0}^{r-1}\delta(k - n\frac{1}{r}) = \frac{1}{r}Comb_{\frac{1}{r},L}(k) \tag{3.22}$$

which are the squared DFT values of Eq. (3.8), ignoring the normalization factor $1/r$.

Figure 3.6 depicts the measurement probabilities when applying Shor's period finding quantum algorithm with parameters $a = 2$ and $N = 21$. In this case $2^6 \bmod 21 = 1$ and thus the period is $r = 6$. The top register which is measured after the inverse QFT consists of $2n = 10$ qubits as $n = \lceil 21 \rceil = 5$, consequently the range of the measurements results is between 0 and 1023. Clearly, the peak probabilities occur close to integer multiples of $L/r = 1024/6 = 170.66\ldots$, namely 0, 171, 341, 512, 683 and 853. A zoom of the probabilities is shown at bottom of Figure 3.6 near the last probability peak that occurs at $k = 853$, which is close to $5 \cdot 1024/6 = 853.33\ldots$ The period $r = 6$ can be extracted by the knowledge of the measurement $k = 853$ together with the knowledge of $L = 1024$, using the CFE method.

3.5 Post-Processing: Retrieval of the exact period

The continued fraction expansion representation of a given rational number ξ is given by

$$c_0 + \cfrac{1}{c_1 + \cfrac{1}{c_2 + \cfrac{1}{\cdots + \cfrac{1}{c_R}}}} \tag{3.23}$$

which is symbolized in compact form as $[c_0, c_1, \ldots, c_R]$. The $j-$th convergent of the CFE is the rational $\xi_j = [c_0, c_1, \ldots, c_j], j = 0 \ldots R$ and it can be always written in fraction form $\xi_j = \frac{e_j}{d_j}$. The period r is found by searching through trial an error over all the convergents' denominators d_j of CFE representation for k/L. In the example above, where $k/L = 853/1024$ the convergents are

$$\frac{0}{1}, \frac{1}{1}, \frac{4}{5}, \frac{5}{6}, \frac{424}{509}, \frac{853}{1024}.$$

The denominator of the fourth convergent is the desired period $r = 6$.

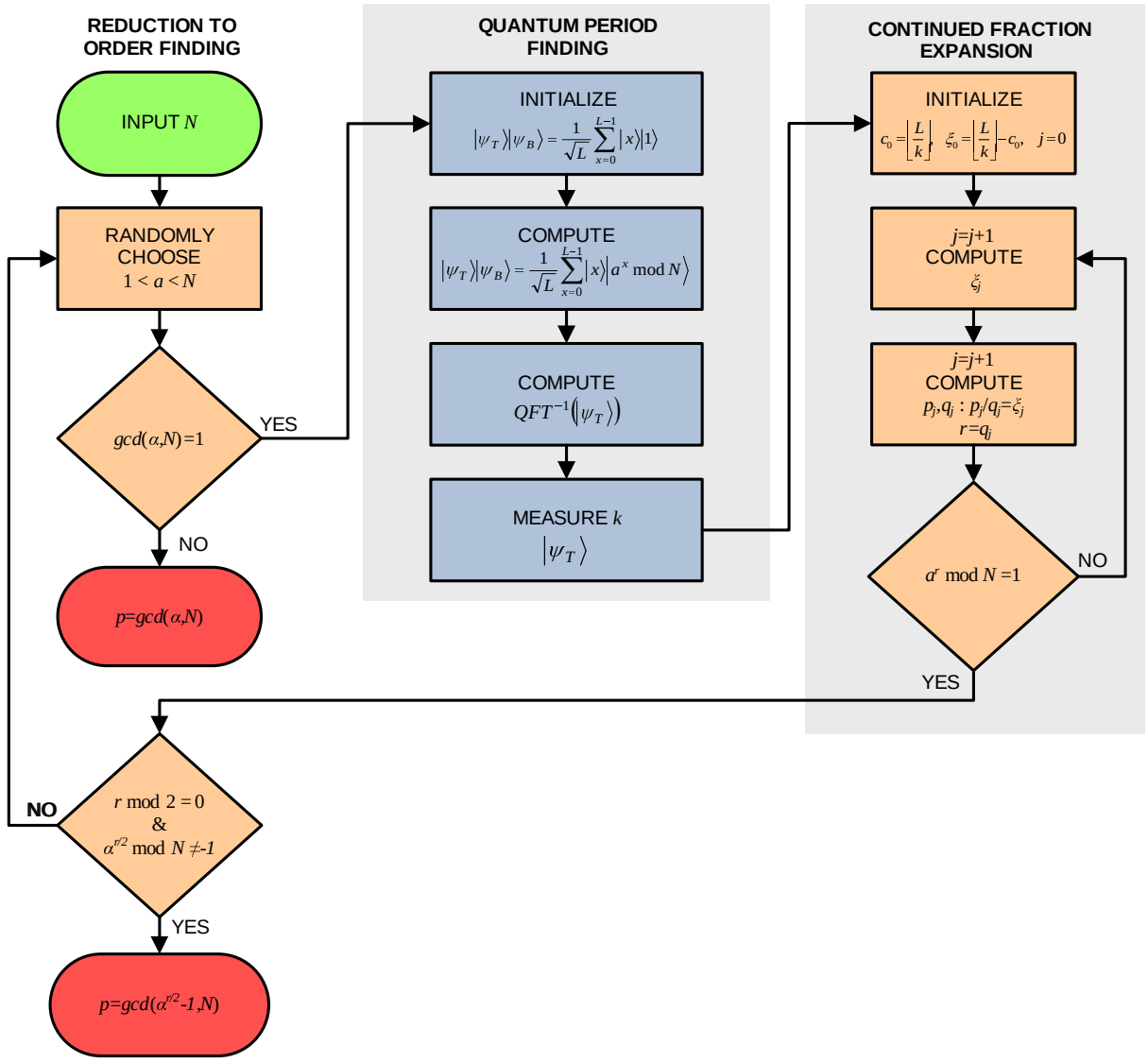


Figure 3.7: Shor's integer factoring algorithm flowchart. The algorithm is divided into the three grayed shades submodules; the probabilistic reduction of the factoring problem to period finding, the quantum computation for the period finding and the exact extraction of the period using the continued fraction expansion method (Appendix A.2).

The CFE computation can be accomplished using a classical computer in polynomial time with respect to n, where n is the number of bits which are adequate to represent the modulus N of the function $f(x) = a^x \bmod N$.

Not every peak leads to a succesful retrieval of the period r. In Appendices A.2 and A.3 the sufficient conditions and the probability of successful retrieval is given. Moreover, the random nature of the measurement does not guarantee a k that is close to an integer multiple of L/r. Thus, the success probability of the quantum part of the period finding algorithm depends on these factors, and it is shown that it has a lower bound of $\Omega(\frac{1}{\log\log N})$. Consequently, $O(\log\log N) = O(\log n)$ iterations of the quantum part are adequate to lead to a constant probability of success to find the unknown period.

The quantum part essentially consists of the QFT which has polynomial space and time complexity with respect to n and the modular exponentiation part which also has polynomial complexity. Thus, the quantum algorithm combined with classical calculations can solve the period finding problem in polynomial time and consequently it can factor an integer N in polynomial time with respect to the number of its digits.

The summary of Shor's factoring algorithm combining both classical and quantum computations is shown in the flowchart of Figure 3.7.

3.6 Decomposition of Quantum Modular Exponentiation

The efficient implementation of the quantum period finding circuit shown in Figure 3.5 requires that the modular exponentiation circuit U_f described by Eq. (3.15) can be implemented polynomially both in space and time with respect to n. A step towards the implementation of U_f is to consider that the modular exponentiation function $a^x \bmod N$ can be decomposed as

$$a^x \bmod N = (a^{2^0} \bmod N)^{x_0} \cdot (a^{2^1} \bmod N)^{x_1} \cdots (a^{2^{2n-1}} \bmod N)^{x_{2n-1}} \bmod N \tag{3.24}$$

where $x_j, (j = 0 \ldots 2n-1)$ are the bits of the x's binary expansion, that is $x = (x_{2n-1} \ldots x_1 x_0)$. Each factor of Eq. (3.24) has the exponent x_j, and thus takes the form of $(a^{2^j} \bmod N)$ if $x_j = 1$, or the constant 1 if $x_j = 0$. Consequently, the operation of U_f given by Eq. (3.15) can be reformulated in the computational basis as

$$U_f(|x\rangle|z\rangle) = |x_{2n-1} \ldots x_1 x_0\rangle|(a^{2^{2n-1}} \bmod N)^{x_{2n-1}} \cdots (a^{2^0} \bmod N)^{x_0} z \bmod N\rangle \tag{3.25}$$

The definition of $n+1$ qubits controlled modular multipliers $c\text{-}U_{a^{2^j}}$ with control qubit $|c\rangle$ and n qubits target register $|y\rangle$, which operate in the computational basis as

$$CU_{a^{2^j}}(|c\rangle|y\rangle) = |c\rangle|(a^{2^j})^c y \bmod N\rangle \tag{3.26}$$

leads to the equivalent design for quantum period finding algorithm shown in Figure 3.8.

The above modular exponentiation decomposition not only helps to construct the U_f operator by reducing the problem to the construction of controlled modular multipliers, but also offers an alternative interpretation of Shor's algorithm as a quantum phase estimation procedure [118],[63], which is explained briefly in Appendix A.5.

The efficient implementation of the modular multipliers of Eq. (3.26) is the subject of Chapter 4 where an overview of the literature is given and various efficient quantum arithmetic circuits are proposed, which when combined together they can offer fast quantum modular exponentiation.

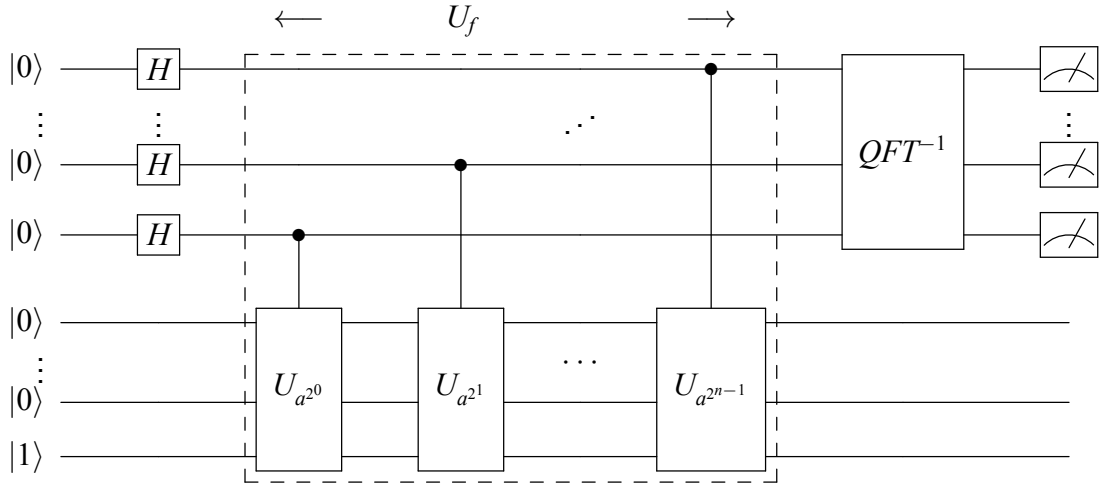


Figure 3.8: Decomposition of the quantum modular exponentiation into quantum modular multiplication in Shor's algorithm.

3.7 Generalizations and the Hidden Subgroup problem

In [10], Shor also described a quantum algorithm which solves efficiently the discrete logarithm problem. This problem can be stated as follows: Given two elements a and $b = a^k$ of a the multiplicative group G_N, where k is an integer, find k. In general, no efficient classical algorithm is known for this problem. This problem can be reduced to period finding problem, where the periodic function has now two arguments. Namely, the function $f(m,n) \doteq b^m a^n = a^{km+n} \bmod N$ is periodic in its arguments m and n, with period $(1,-k)$ because $f(m+1,n-k) = a^{mk+k+n-k} = f(m,n)$. Thus, the extraction of the period $(1,-k)$ directly leads to the discrete logarithm of b in the base a.

A high level quantum circuit description of the discrete logarithm algorithm is given in Figure 3.9. It consists of two control registers (top) and a target register (bottom). The size n of the registers depends on the order r of the base b. The blocks U_a and U_b that are controlled by the two top registers are quantum modular exponentiators of modulus N with parameters a and b respectively, and they can be decomposed as a sequence of controlled modular multipliers. Inverse QFT is applied on the control registers and then measurement of both can lead to the answer k.

Quantum algorithms such as Deutsch's, Deutsch-Josza, Simon's, Shor's period finding and discrete logarithm can be described in a common framework under the *Hidden Subgroup Problem*.

Definition 3.1. *(Hidden Subgroup Problem) Let G is a finite generated group, and f a function with domain the group G and with range another set X. If f is constant and distinct in each coset gH, the hidden subgroup problem is to find the subgroup H through its generators.*

The cosets gH of the subgroup H with respect to the element g are defined as $gH = \{gh : h \in H\}$.

In Deutsch's problem the group is $G = \{0,1\}$ and it has two subgroups $H_1 = \{0\}$ and $H_2 = G$. If the function f is balanced then f is constant and distinct in each coset of H_1, while if f is constant, then f is constant in each coset of H_2. The problem to decide if f is balanced or constant is to find the subgroup H_1 or H_2, respectively.

In Shor's quantum period finding, the group G is $\mathbb{Z}$ and the subgroup H is $r\mathbb{Z} = \{0, r, 2r, \ldots\}$, where r is the period of f. The function $f(x) = a^x \bmod N$ is constant with distinct values in

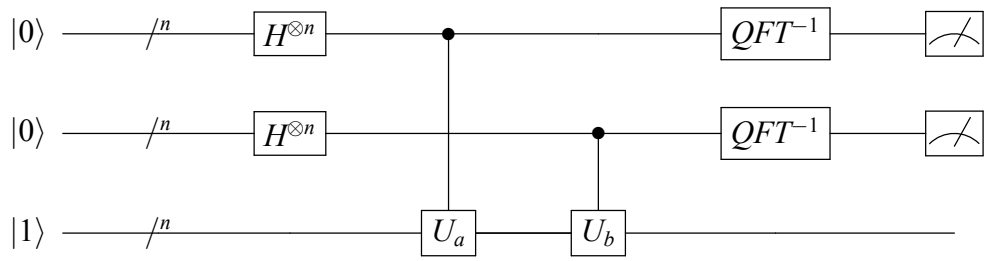


Figure 3.9: Quantum circuit of the discrete logarithm algorithm

each coset rH and the generator of $r\mathbb{Z}$ is r itself, which is the period.

In the discrete logarithm problem, the group G is $\mathbb{Z}_r \times \mathbb{Z}_r$ and the hidden subgroup is $H = \{(m,n) \in G : km + n = 0 \bmod r\}$. A generator of H is $(1-,k)$ which reveals the logarithm k.

Some other problems that can be also casted under the Hidden Subgroup formulation are the Hidden Linear functions [119] and the Abelian Stabilizer problem [118]. The common characteristic of all these problems is that in their Hidden Subgroup formulation the group is Abelian (commutative). In this particular case of Abelian groups, general procedures that lead to efficient quantum algorithms exist [120] and the structure of their circuits is similar to that of Figure 3.9. The Hidden Subgroup problem for non-Abelian groups can be solved only for some special cases of groups. A review of this non-Abelian Hidden Subgroup problem can be found in [121].

4. FAST QUANTUM MODULAR EXPONENTIATION

This Chapter, which is our first contribution in this thesis, is devoted to quantum arithmetic circuits, especially those that are useful in modular exponentiation, which is the most computational intensive part of the very important Shor's factoring and discrete logarithm algorithms. An overview of elementary quantum arithmetic and modular exponentiation designs is provided and novel elementary quantum arithmetic circuits are proposed. These novel circuits are based on the representation of an integer in the Fourier domain using QFT and they can be integrated together to offer fast quantum modular exponentiation with quadratic depth and linear space. Expressly, the proposed modular exponentiation circuit has a depth of about $700n^2$ and requires $9n + 2$ qubits (or $8n + 2$ in an improved version discussed in section 4.7), where n is the number of bits of the integer. The total quantum cost of the proposed design is $400n^3$ for single or two-qubit gates counting. These characteristics refer to the case where the modular exponentiation is applied to Shor's algorithm. A general case modular exponentation circuit exposed in section 4.4 has inferior characteristics in cost, space and time. The main contributions of this Chapter are :

- A QDT based quantum controlled multiplier/accumulator by constant, based on previous work done by Draper [23] and Beauregard [24], offering the advantage of linear depth instead of the original quadratic. This reduction is achieved by exploiting the fact that one factor of the product is constant, decomposing doubly controlled rotation gates to simply controlled rotation gates [53] and then suitably rearranging them so as many of them can be executed concurrently.

- A quantum divider by constant based on the division by constant classical algorithm proposed by Granlund and Montgomery [122]. The quantum divider incorporates QFT based arithmetic blocks like adders and the uncontrolled version of the previous multiplier/accumulator. This quantum divider has also linear depth. A controlled version of the quantum divider is also presented.

- By combining the two previous building blocks we achieve to build a quantum controlled modular multiplier with linear depth which is then used as the basic building block for the quantum modular exponentiation circuit of quadratic depth.

We provide detailed complexity analysis both in terms of space and time along with comparisons with other circuits presented in the literature. Implementation difficulties related to fault tolerance and local interactions between the qubits are also discussed and procedures to address them are given in Chapter 5.

4.1 Background and related work

We have already shown in section 3.6 how to decompose the quantum modular exponentiation part into a sequence of controlled modular mutlipliers defined with Eq. (3.26). Similar decompositions are used to break the controlled modular mutlipliers to simpler circuit stages down to quantum adders. These intermediate circuits may use ancilla qubits for the computation of some temporary results. It is of crucial importance that all these ancilla qubits are reset back to a constant state (e.g. $|0\rangle$) so as the end result of modular exponentiation does not contain garbage information. This is because the derivation of Eq. (3.18) relies on the fact that an equal superposition of $|x\rangle$ states is entangled with states given by a periodic function $|f(x)\rangle$. If garbage states $|g(x)\rangle$ are generated additionally to the modular exponentiation computation $|f(x)\rangle$, then the joint state $|f(x)\rangle|g(x)\rangle$ is no longer periodic with respect to x and the superposition

$$|\psi\rangle = \frac{1}{\sqrt{2}} \sum_{x=0}^{L-1} |x\rangle |f(x)\rangle |g(x)\rangle \tag{4.1}$$

cannot lead to any period extraction by applying an inverse QFT on the left register.

4.1.1 Modulo adder,constant adder and controlled constant adder

Any abstract quantum adder design from the literature, e.g [61, 62, 123], can be used to describe the usual design hierarchy of these controlled modular multipliers. This abstract adder has two operating registers of n qubits, initially in the states $|a\rangle = |a_{n-1} \dots a_0\rangle$ and $|b\rangle = |b_{n-1} \dots b_0\rangle$. Usually, another ancilla n qubits register is needed for the carries calculations, although designs with no ancilla have appeared [123, 23]. This register is initialized in the zero state and through uncomputation goes back again to the zero state. The adder operation is to leave the first register intact in the state $|a\rangle$, and write the sum (modulo 2^n) to the second register, that is

$$ADD(|a\rangle|b\rangle) = |a\rangle|(b+a) \bmod 2^n\rangle \tag{4.2}$$

This operation is unitary, and the inverse operation ADD^{-1} which is subtraction (modulo 2^n),

$$ADD^{-1}(|a\rangle|b\rangle) = |a\rangle|(b-a) \bmod 2^n\rangle \tag{4.3}$$

can be easily implemented by reversing the order of gates used in the adder circuit and using in their place, the respective inverse ones.

When the two n qubits integers are considered unsigned, the most significant qubit $|b_{n-1}\rangle$ of the second register could be used as an overflow indicator for the range $\left[0 \dots 2^{n-1}-1\right]$, in a subtraction operation. Consequently, the subtractor could be used as a comparator too, by checking this most significant qubit.

The adder of two integers a and b can be transformed to a controlled constant adder; it adds a constant integer N to a register initially in state $|b\rangle$ conditioned on a control qubit $|c\rangle$ as

$$c\text{-}ADD_N(|c\rangle|b\rangle) = |c\rangle|(b+cN) \bmod 2^n\rangle \tag{4.4}$$

The modulo 2^n operation will be implied from now on in an adder or subtractor operating with n bits numbers. Figure 4.1 shows the transformation of a general adder to a controlled constant adder. The controlled blocks c-N act on an n qubits target register and perform the operation $|x\rangle \rightarrow |x \oplus cN\rangle$ where $|c\rangle$ is the control qubit state. When $|x\rangle$ is initially $|0\rangle$, on the condition that $|c\rangle = |1\rangle$, the register is loaded with the value $|N\rangle$, otherwise it remains $|0\rangle$. It can be realized by using up to n CNOT gates depending on the binary representation of constant N. Thus, N is hardwired in the internal structure of c-N. On the other side, the action of a c-N circuit on a register with initial state $|N\rangle$ is to reset it to the state $|0\rangle$, when the control qubit is $|1\rangle$. When the control qubit of Figure 4.1 is $|1\rangle$, the top register of the adder ADD which is initially in state $|0\rangle$ is fed with the constant value N and afterwards is reset again to $|0\rangle$. The lower register contains the sum $b+cN$, that is b if $c=0$, or $b+N$ if $c=1$.

A similar construction can be employed to build a constant adder without any control. E.g., a normal adder fed with the constant value N on one register and afterwards reseting this

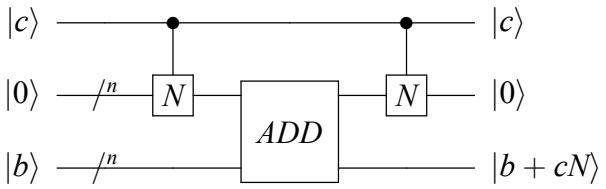


Figure 4.1: Quantum circuit for controlled constant addition.

register with suitable arranged X gates. Another option would be to discard the register carrying the constant value and reorganize the internal structure of the adder depending on this constant value.

Normal adders, constant adder and controlled constant adder can inter-operate as shown in Figure 4.2 to build a modulo N adder whose operation is defined by

$$c\text{-}ADDM_N(|a\rangle|b\rangle) = |a\rangle|(b+a) \bmod N\rangle \tag{4.5}$$

Two n qubits registers and an ancilla qubit are shown in this construction. It is assumed that n is chosen so that $0 \leq a, b < N < 2^{n-1}$. The first adder alters the state of the two register to $|a\rangle|b+a\rangle$. The constant inverse adder ADD_N^{-1}, which is effectively a constant subtractor, evolves the state of the second register to $|a+b-N\rangle$. The most significant qubit of the second register gives an indication whether an overflow has occurred, that is $a+b<N$. In such a case its state would be $|1\rangle$, otherwise (when $a+b \geq N$) it state would be $|0\rangle$. The CNOT gate targeting the single ancilla qubit at the bottom of the circuit is controlled by this most significant qubit. Thus, a state $|1\rangle$ on the ancilla qubit indicates overflow, while a state $|0\rangle$ indicates no overflow. The ancilla qubit containing the overflow information, controls the constant adder ADD_N, altering the second register state to $|b+a-N+N\rangle = |b+a\rangle$ if $a+b<N$, or to $|b+a-N\rangle$, if $a+b \geq N$. Both states are equal to $|b+a \bmod N\rangle$, due to the assumption $0 \leq a, b < N$. At this point the joint state of the circuit is $|a\rangle|b+a \bmod N\rangle|\overline{\lceil (b+a)/N \rceil}\rangle$, where notation $\overline{\lceil (b+a)/N \rceil}$ indicates the undeflow information.

The purpose of the last two adders and the CNOT gate is to reset the ancilla qubit holding this overflow information back to $|0\rangle$ without disturbing the two registers. The effect of the inverse adder and the last adder on both registers is the identity operation (no effect) and consequently both registers wiil remain at the state $|a\rangle|b+a \bmod N\rangle$ at the end. Yet, in between these last adders, the state of the second register is $|((b+a) \bmod N) - a\rangle$. Thus, its most significant qubit is in state $|0\rangle$ if an overflow had occurred previously, as long as in such a case $(b+a) \bmod N = b+a$, $((b+a) \bmod N) - a = b$ and $b < N$ by assumption. The modified CNOT gate with the white circle on the control qubit (which is again the most significant qubit of the register) resets the ancilla state back to $|0\rangle$ and thus the ancilla qubit

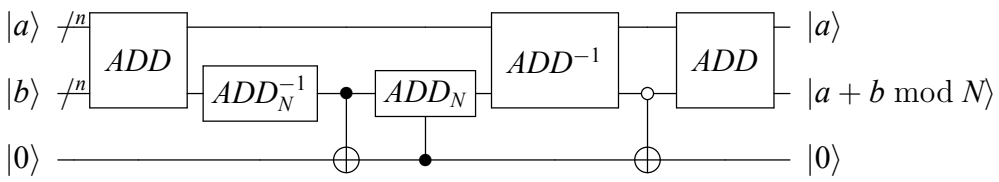


Figure 4.2: Quantum circuit for modular addition. The white circle of the second CNOT gate denotes inversion of its target qubit iff the control qubit is $|0\rangle$. Control qubits of both CNOT gates emerge from the most significant qubit of the register on which they are attached. Ancilla qubits of the various adders used in this figure are hidden inside their symbols.

can be reused. If no overflow occured, again the ancilla qubit remains in state $|0\rangle$ as the control qubit of the white circled CNOT gate is $|1\rangle$.

4.1.2 Controlled modulo multiplier

The controlled modular multiplier of Eq. (3.26) can be broken down to a sequence of modular adders. For each modular multiplier $CU_{a^{2^j}}$, defined in Eq. (3.26), the argument y to be multiplied by the constant a^{2^j} can be expanded as $y = 2^{n-1}y_{n-1} + 2^{n-2}y_{n-2} + \cdots 2y_1 + y_0$, where $(y_{n-1}y_{n-2}\ldots y_1y_0)$ is the binary representation of y. After this observation, we define a controlled modular multiplier/accumulator $CV_{a^{2^j}}$ with

$$\begin{aligned} CV_{a^{2^j}}\left(|c\rangle|y\rangle|z\rangle\right) &= |c\rangle|y\rangle|z + (a^{2^j})cy \bmod N\rangle \\ &= |c\rangle|2^{n-1}(z + a^{2^j})cy_{n-1} + 2^{n-2}(a^{2^j})cy_{n-2} + \cdots (a^{2^j})cy_0 \bmod N\rangle \end{aligned} \tag{4.6}$$

The above equation suggests a conditionally iterated modulo N addition of the constants $A_k \doteq 2^k a^{2^j} \bmod N$, for $k = 0 \ldots n-1$ to an n qubits register initialized to zero. The condition to execute the addition of A_k is that both c and y_k must be 1. A step towards this construction is the circuit of Figure 4.3. A modulo N adder is sandwiched between two double-controlled A_k blocks which are controlled by qubits $|c\rangle$ and $|y_k\rangle$. They affect the first register of the modulo N adders, which is initialized to $|A_k\rangle$ before each addition and reset back to $|0\rangle$ after each addition. The implementation of the doubly controlled A_k circuit is similar to the implementation of the simply controlled blocks c-N used in the constant adder of Figure 4.1, but it requires Toffoli gates instead of CNOT gates.

The circuit of Figure 4.3 implements a slightly different operation from the desired one described in Eq. (3.26). It can achieve the controlled modular multiplication if the register that contains the $|y\rangle$ argument is reset to zero and the last register is initialized to $|z\rangle = |0\rangle$. If constant a is co-prime with N, which is the case in Shor's algorithm, then there exist its inverse a^{-1} so as $a \cdot a^{-1} \bmod N = 1$. An inverse of a^{2^j} also exists, namely a^{-2^j}. The circuit $CV_{a^{2^j}}$ combined with its inverse $CV^{-1}_{a^{-2^j}}$ can be used to implement the desired controlled modular multiplier of Eq. Eq. (3.26). This inverse block subtracts instead of adding and has parameter a^{-2^j}, thus it computes

$$CV^{-1}_{a^{-2^j}}\left(|c\rangle|y\rangle|z\rangle\right) = |c\rangle|y\rangle|z - (a^{-2^j})cy \bmod N\rangle \tag{4.7}$$

Figure 4.4 shows a combination of $CV_{a^{2^j}}$ with its inverse $CV^{-1}_{a^{-2^j}}$ and a Fredkin gate that derives the controlled modular multiplication which is essential for the modular exponentiation circuit as decomposed in Figure 3.8. When the control qubit is $|0\rangle$ the circuit outputs

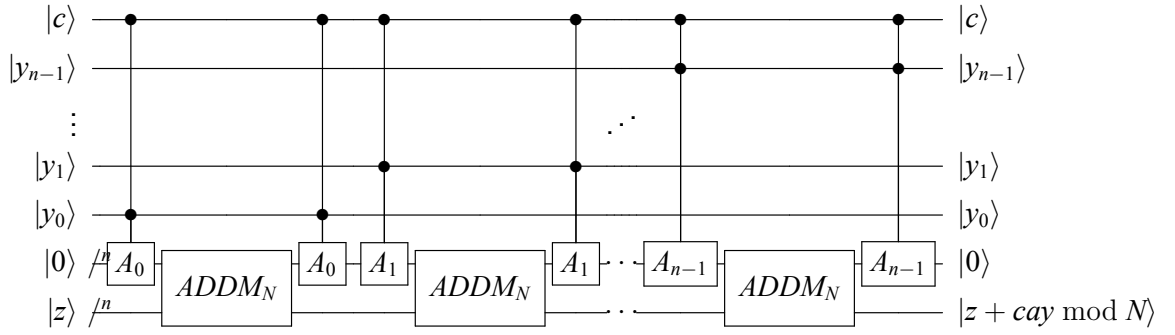


Figure 4.3: Quantum circuit for controlled accumulation of modular multiplication. Ancilla qubits of the modular adders used in this figure are hidden inside their symbols.

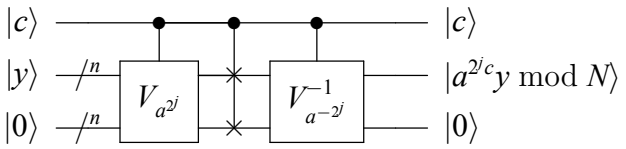


Figure 4.4: Quantum circuit for controlled modular multiplication. Ancilla qubits are not shown in the symbols of the two blocks.

the state $|y\rangle$ on the top register and $|0\rangle$ on the bottom register. When the control qubit is $|1\rangle$, the two registers after the $CV_{a^{2^j}}$ block evolve in the joint state $|y\rangle|a^{2^j} y \bmod N\rangle$. Fredkin gate swaps the contents of the two registers and the $CV^{-1}_{a^{-2^j}}$ block evolves its swapped input joint state $|a^{2^j} y \bmod N\rangle|y\rangle$ to $|a^{2^j} y \bmod N\rangle|y - a^{-2^j} a^{2^j} y \bmod N\rangle = |a^{2^j} y \bmod N\rangle|0\rangle$.

The standard modular exponentiation decomposition, for modulus N which has a length of n bits, needs $O(n)$ controlled modular multipliers. Each controlled modular multiplier needs in turn $O(n)$ adders, thus $O(n^2)$ adders are required for the modular exponentiation. The total depth of the quantum modular exponentiation for the standard decomposition described above depends on the particular adder implementation and in general it ranges between $O(n^2 \log n)$ and $O(n^3)$.

The space cost (number of qubits required) of the standard modular exponentiation is $7n+1$ qubits, of which $4n+1$ qubits are ancilla. These ancilla are broken down as follows: n qubits for the internal carries of each adder, 1 qubit for the sign extraction for modular addition, n qubits for the constant N, n qubits for the constants $2^k a^{2^j} \bmod N$ and last, n qubits for the conversion of the controlled modular MAC in Figure 4.3 to the controlled modular multiplier in Figure 4.4. The working qubits consist of a $2n$ qubits register holding the superposition of x's while an n qubits register holds the computation of the modular exponentiation $a^x \bmod N$. The number of the ancilla registers could be reduced if we exploit the fact that some of them are fed with constant values depending on N and a. This reduction is possible if these constants are "hardwired" inside each adder by suitably reorganizing its structure. In this case the space requirements can be reduced to $5n + 2$ qubits. Usage of adders which have no ancilla qubits for the internal carries (like the ones in [123, 23]) leads to circuits of $4n + 2$ width for the modular exponentiation.

Further reduction of the space can be achieved by exploiting the semi-classical QFT implementation and the fact that the controlled modular multiplier blocks $CU_{a^{2^j}}$ mutually commute. Therefore, the circuit of Figure 3.8 can be re-designed into that of Figure 4.5 which uses only one qubit for controlling $2n$ $CU_{a^{2^i}}$ gates instead of using $2n$ different control qubits [24, 120, 124].

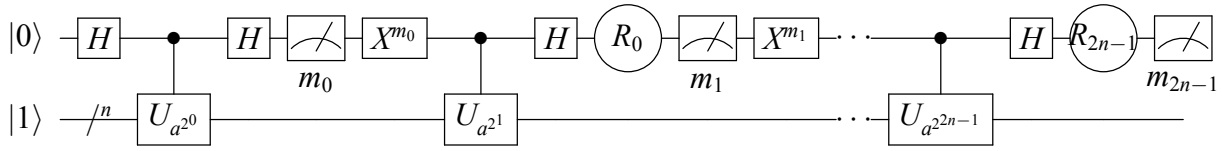


Figure 4.5: Design of modular exponentiation circuit using only one qubit to control the modular multipliers. The phase shift gates R depend on all previous measurement results and implement the inverse QFT, while the X gates are negations conditioned on the result of the previous measurement.

4.1.3 Prior Work

Several designs for quantum exponentiation have been proposed in the literature. In general, they adopt the top-down approach of the previous subsection which reduces the construction down to adders. For this reason, most of the effort in the literature has been

devoted to the design of the quantum equivalent of a digital adder and its improvement in terms of complexity: reduction of the total number of required quantum gates and qubits (ancilla and working) and reduction of the circuit depth (total number of steps required to complete the computation).

Many of the quantum addition circuits introduced in the literature are inspired from their known classical counterparts through the design of reversible versions of them. The most important of these approaches are the Vedral-Barenco-Ekert (VBE) ripple-carry adder [61], the Beckman-Chari-Devabhaktuni-Preskill (BCDP) ripple-carry adder [62], the Cuccaro-Draper-Kutin-Moulton (CDKM) ripple-carry adder [125], the Draper-Kutin-Rains-Svore (DKRS) carry-lookahead adder [126], the Takahashi-Kunihiro (TK) ripple-carry adder [123], the Gossett carry-save adder [127], the Zalka carry-select adder [128] and the Van-Meter-Itoh (VI) carry-select adder and conditional-sum adder [129]. Some of the above proposals for quantum addition circuits emphasize on minimizing the number of required qubits, while other methods try to minimize the circuit depth. Other efforts concentrate on building architectures restricted on the condition of local communications between the qubits either in 1D-NTC (1-Dimension, linear Nearest-neighbour, Two qubit gates, Concurrent execution) such as those of Fowler-Devitt-Hollenberg (FDH) [130] and Kutin's [131], or in 2D-NTC such as those of Choi-VanMeter (CV) [132] and Pham-Svore (PS) [133].

The method to build a complete modular exponentiation circuit based on a particular addition circuit is not unique, and various studies concerning the trade-off between space (number of required qubits) and time (depth of the circuit) have been reported as in [129]. Not all previous publications provide a complete modular exponentiation circuit, but assuming we can build one using the previously discussed hierarchy (adder)-(modular adder)-(modular multiplier)-(modular exponentiator), we can make rough approximations about the design complexity in each case and comparisons with our proposed exponentiation design (see Section 4.6 for the comparisons).

Notable exceptions of the above top-down trend that builds a complete modular exponentiation circuit from the quantum equivalent of classical binary adder are circuits that use the Draper's QFT adder [23] like Beauregard's circuit [24], Fowler-Devitt-Hollenberg circuit (FDH) [130], and Kutin's first circuit of [131]. These three circuits implement the addition of two integers by converting one of them in the Fourier domain using QFT and then converting the sum back to the binary representation. Another method which surpasses the common hierarchy of computation is Zalka's FFT multiplier [128] that implements a multiplier using the FFT method of computing a convolution sum.

4.1.4 QFT adders

We first describe the four QFT adders [23] that will be extensively used in our modular exponentiation design. Since in every iteration of Shor's algorithm the randomly picked number a in Eq. (3.15) remains constant, we need an adder receiving a quantum integer (that is a potential superposition of integer x as required by the algorithm) as its first operand and a constant classical integer as its second. Three variations of this adder are required for the multiplier/accumulation unit: the constant QFT adder (ΦADD), the controlled constant QFT adder (CΦADD) and the doubly controlled constant QFT adder (CCΦADD). A generic QFT adder for adding two quantum integers will be also used subsequently in the quantum divider circuit. These four QFT adders will be denoted in all figures with ΦADD as they can be easily differentiated by the quantum wires connected as inputs and outputs on their symbols. Figure 4.6 shows the simple (uncontrolled) constant QFT adder ΦADD. It adds the n-bit constant integer a to an n-qubit quantum integer $|b\rangle = |b_{n-1}\rangle \cdots |b_1\rangle|b_0\rangle = |b_{n-1}\rangle \otimes \cdots \otimes |b_1\rangle|b_0\rangle$. Integer b must be already transformed in

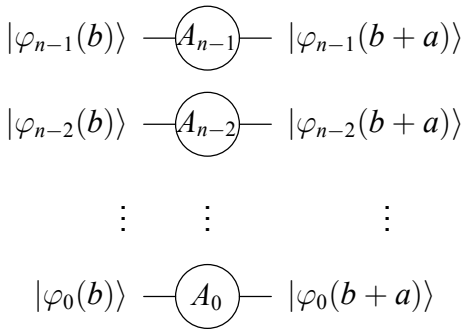


Figure 4.6: ΦADD adder circuit of depth 1. This circuit adds a constant integer a to the quantum integer b, when b is already in the Fourier domain. The value of integer a is hardwired in the angles of the phase shift gates $A_j, j = 0 \ldots n-1$ as defined in Eq. (4.10)

.

the Fourier domain by a QFT block before entering the ΦADD block through the relation:

$$|b\rangle \xrightarrow{QFT} |\varphi(b)\rangle = |\varphi_{n-1}(b)\rangle|\varphi_{n-2}(b)\rangle \ldots |\varphi_0(b)\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{i\frac{2\pi}{2^n}bk}|k\rangle \tag{4.8}$$

The individual j^{th} qubit $|\varphi_j(b)\rangle$ of the quantum Fourier transformed integer is given (see also Eq. (3.5))by the relation:

$$|\varphi_j(b)\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\frac{2\pi}{2^j}b}|1\rangle) \tag{4.9}$$

The single qubit gates shown in Figure 4.6 are rotation quantum gates described by the equation (with R_k is denoted the phase shift gate $R_z(2\pi/2^k)$):

$$A_j = \prod_{k=1}^{j+1} R_k^{a_{j+1-k}}, \quad R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{2\pi}{2^k}} \end{bmatrix} \tag{4.10}$$

Therefore, it can be shown that the output of the ΦADD circuit is $|\varphi(b+a)\rangle$, the quantum Fourier transformed sum $b+a$. The sum can be recovered in the computational basis by applying an inverse QFT afterwards. Excluding the forward and inverse QFT, the circuit

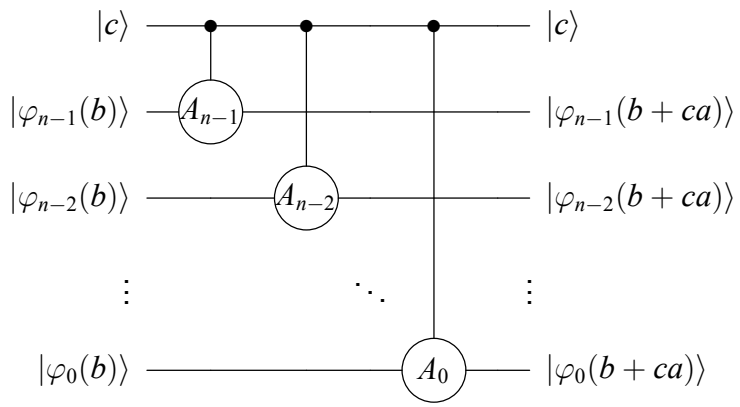


Figure 4.7: CΦADD controlled adder circuit of depth n. This circuit adds the constant a to the quantum integer b when the control qubit $|c\rangle$ is $|1\rangle$. Again, the constant value a is hardwired in the controlled rotation gates as defined in Eq. (4.10)**.**

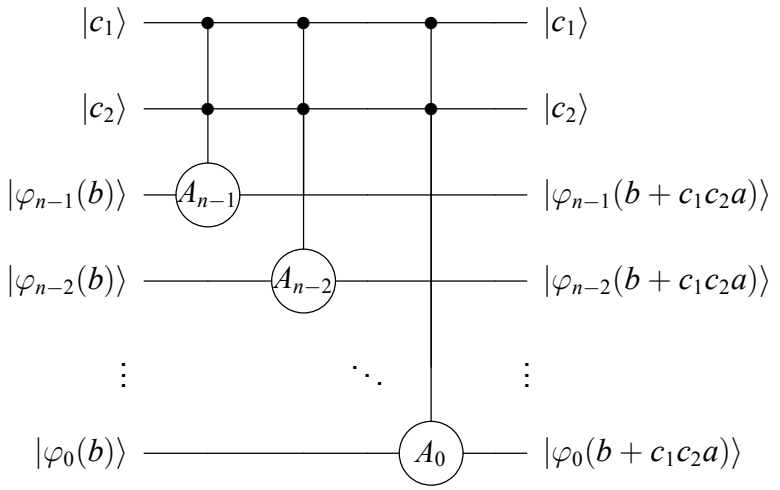


Figure 4.8: The doubly-controlled adder circuit CCΦADD of depth n. This is an extension of the CΦADD circuit where the addition is performed when both the control qubits $|c_1\rangle$ and $|c_2\rangle$ are $|1\rangle$.

has a complexity of n qubits and a depth of 1 because the rotation gates can all operate in parallel.

An extension of the constant adder ΦADD circuit can be done if we use controlled rotation gates with same rotation angles as those defined in Eq. (4.10). This circuit is depicted in Figure 4.7. It has a common controlling qubit $|c\rangle$ for each rotation gate and performs the following transform

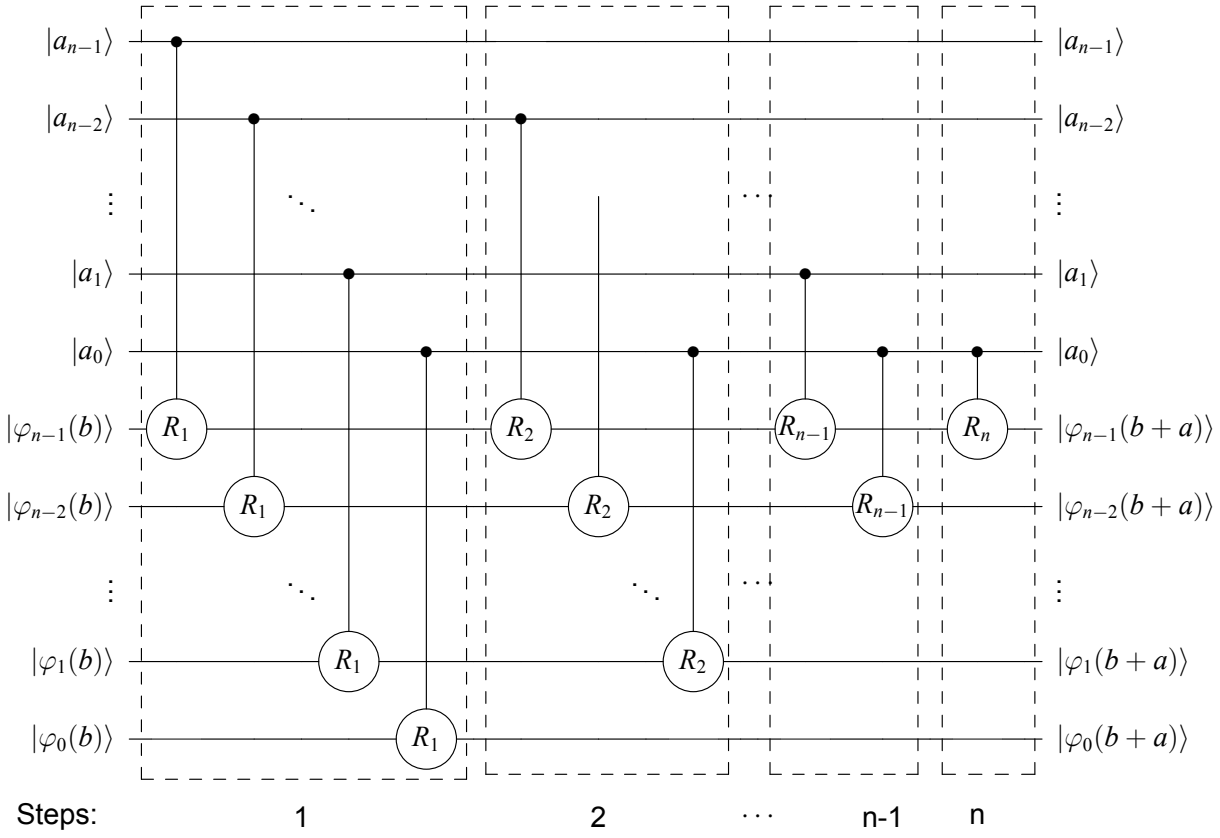


Figure 4.9: Generic adder ΦADD circuit and its symbol. The top bus consists of the qubits $|a_0\rangle, \ldots, |a_{n-1}\rangle$ that control the rotation gates.

$$C\Phi ADD_a(|c\rangle|\varphi(b)\rangle) = |c\rangle|\varphi(b+ca)\rangle \tag{4.11}$$

The CΦADD circuit performs the addition only when the controlling qubit $|c\rangle$ is $|1\rangle$ giving the result $|\varphi(b+a)\rangle$, otherwise the result is the input $|\varphi(b)\rangle$. The CΦADD adder uses $n+1$ qubits and its depth is n (excluding the required QFT and inverse QFT) because the controlled rotation gates must operate sequentially as they have a common controlling qubit.

A further extension is shown in Figure 4.8, which is the doubly-controlled ΦADD circuit (CCΦADD). The circuit is similar to the CΦADD, but it uses doubly controlled rotation gates. The two controlling qubits of each rotation gate are $|c_1\rangle$ and $|c_2\rangle$. The CCΦADD circuit applies the transform:

$$CC\Phi ADD_a(|c_1\rangle|c_2\rangle|\varphi(b)\rangle) = |c_1\rangle|c_2\rangle|\varphi(b+c_1c_2a)\rangle \tag{4.12}$$

That is, it computes the addition only when both the controlling qubits $|c_1\rangle$ and $|c_2\rangle$ are $|1\rangle$ and gives $|\varphi(b+a)\rangle$, otherwise the result is the input $|\varphi(b)\rangle$. The CCΦADD adder uses $n+2$ qubits and like the CΦADD has a depth of n because the doubly-controlled rotation gates must operate sequentially as they have common controlling qubits.

Finally, we give in Figure 4.9 the circuit diagram of the generic QFT adder (ΦADD) which adds two quantum integers. The circuit diagram of Figure 4.9 is the parallel version of the adder and it has depth n. The operation of the circuit is to add two quantum integers, each one of n qubits, and is described by Eq. (4.13):

$$\Phi ADD(|a\rangle|\varphi(b)\rangle) = |a\rangle|\varphi(b+a)\rangle \tag{4.13}$$

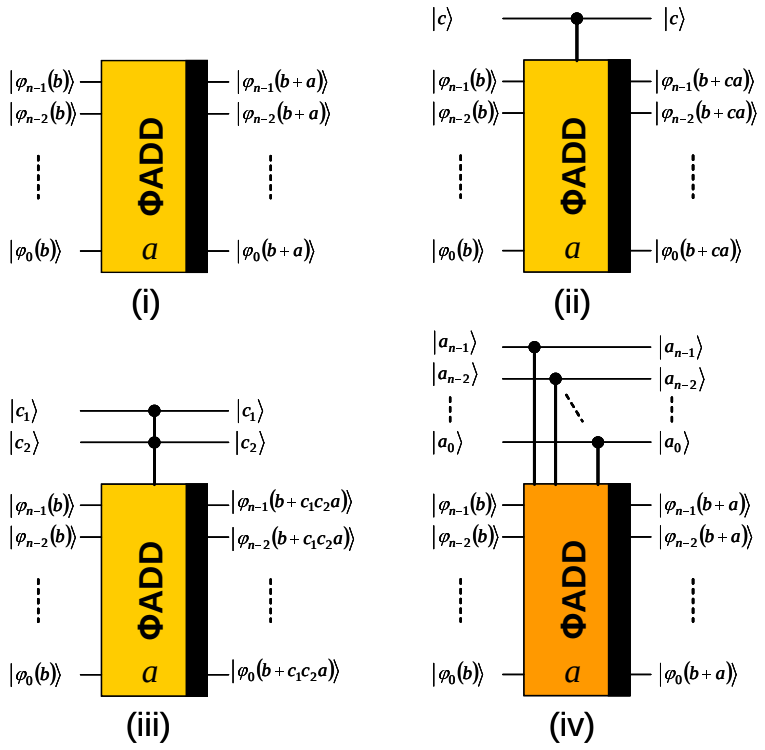


Figure 4.10: Symbols for the four introduced QFT adders. (i) Adder with constant a, (ii) controlled adder with constant a, (iii) doubly controlled adder with constant a and (iv) adder of two quantum integers a and b.

The top input bus of the circuit consists of the qubits $|a_0\rangle, \ldots, |a_{n-1}\rangle$ that control the rotation gates R_k. These qubits remain unaltered by the ΦADD unit. As shown in Figure 4.9 and Eq. (4.13) one of the integers must be already transformed in the QFT domain prior the application of the circuit and then inverse QFT transformed to deliver the sum in the computational basis. The depth of the generic adder is also n.

Figure 4.10 shows the symbols of the four QFT adders that will be used throughout this Chapter for the construction of more complex arithmetic units. The thick bars on the right of each symbol denote addition. Inverse version of the four adders (that is subtractors) can be built by using opposite sign in the angles of each rotation gates used in each one of the circuits. These inverse blocks will be denoted with a thick bar on the left side of each symbol.

4.1.5 Fourier Multiplier/Accumulator - ΦMAC

In this section we describe a quantum circuit given in [24], which from this point onwards we name ΦMAC; it utilizes the CCΦADD adders described in the previous subsection and accumulates the product of a constant n-bit integer a with a quantum n-qubit integer, $|x\rangle$ to a quantum $2n$-qubit integer $|b\rangle$, giving the accumulation result $|b+ax\rangle$. Furthermore, the circuit has a controlling qubit $|c\rangle$, that enables (when $|c\rangle = |1\rangle$) or disables (when $|c\rangle = |0\rangle$) the accumulation operation (in the latter case the result is $|b\rangle$). Hence, the circuit uses a total of $3n+1$ qubits and its operation can be described as:

$$\Phi MAC_a(|c\rangle|x\rangle|b\rangle) = (|c\rangle|x\rangle|b+cax\rangle) \tag{4.14}$$

Taking into account the binary expansion of integer $x = (x_{n-1}, \ldots, x_1, x_0)$, we can write the product ax as:

$$ax = x_0 a + x_1 2a + \cdots + x_{n-1} 2^{n-1} a \tag{4.15}$$

Therefore, the accumulation of the product ax with b can be achieved by the successive addition of n constant integers $a, 2a, \ldots, 2^{n-1}a$, each one being added conditionally on the qubit value $x_j, j = 0, 1, \ldots, n-1$, respectively. Hence, the ΦMAC circuit can be built as

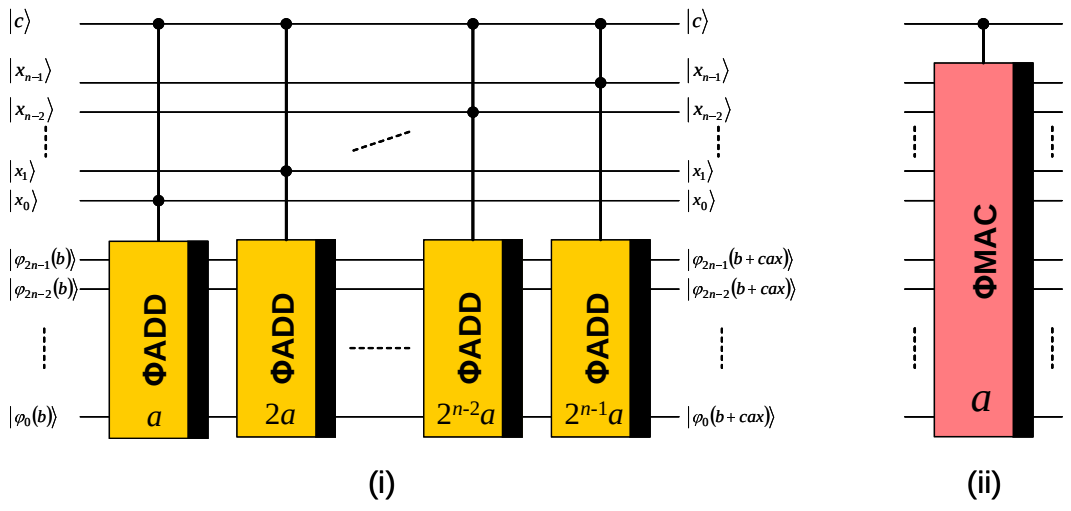


Figure 4.11: Block level design of the multiplier/accumulator unit ΦMAC and its symbol. The basic blocks depicted here are the CCΦADD units of Figure 4.8. A detailed diagram of the above circuit is provided in Figure 4.12.

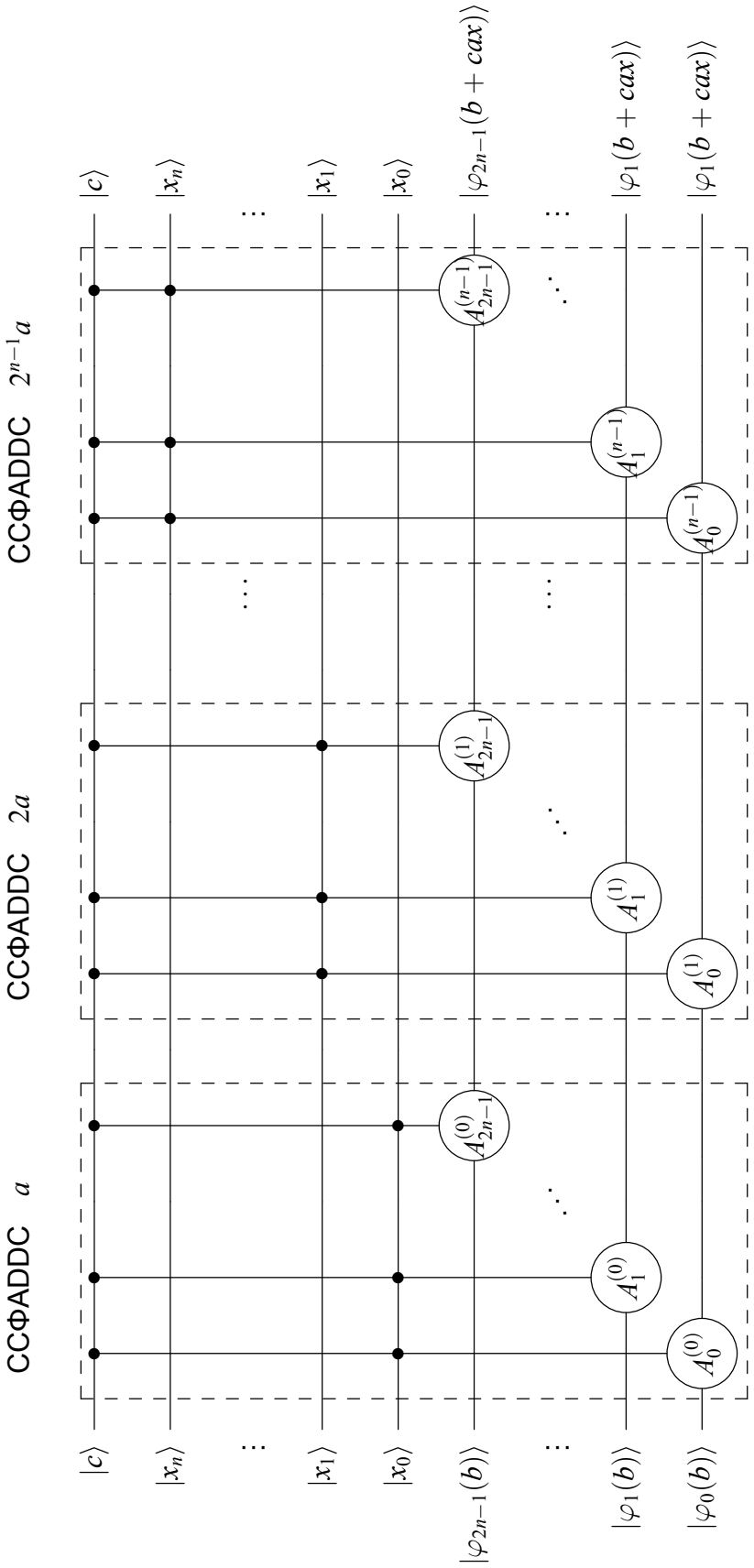


Figure 4.12: Detailed design of the initial multiplier/accumulator ΦMAC unit which has a depth of $2n^2$. The depth improvement of this circuit is described in Section 4.2 and the improved ΦMAC is depicted in Figure 4.15.

shown in Figure 4.11, assuming that the lowest $2n$ qubits (those that participate in the accumulation) are already transformed in the QFT representation by a previous QFT block. Actually, the circuit comprises a series of CCΦADD blocks described in the previous subsection, each one adding in succession the integers $a, 2a, \ldots, 2^{n-1}a$, and controlled by their two controlling qubits. The first controlling qubit $|c_1\rangle$ is common to all the CCΦADD blocks and becomes the controlling bit for the ΦMAC block. The second controlling qubit $|c_2\rangle$ of the j^{th} adder CCΦADD corresponds to input qubit $|x_j\rangle, j = 0, \ldots, n-1$ of the ΦMAC.

The detailed design of the ΦMAC block that consists of doubly controlled rotation gates, is depicted in Figure 4.12. The j^{th} CCΦADD block adds the constant integer $c^{(j)} = 2^j a$ to the $2n$ qubits that hold the accumulation result (in the QFT field). The bits c_l^j corresponding to the binary expansion of $c^{(j)} = 2^j a$ are given by:

$$c_l^{(j)} = a_{l-j}, \quad l = 0, \ldots, 2n-1, j = 0, \ldots, n-1 \tag{4.16}$$

assuming that $a_{-1}, a_{-2}, \ldots, a_{-n} = 0$. Taking into account Eq. (4.10) we can deduce that the doubly controlled rotation gates $A_l^{(j)}$ in Figure 4.12 affect the l^{th} qubit of the accumulation register by using the following rotation matrix (if both the controlling qubits are in state $|1\rangle$.

$$A_l^{(j)} = \prod_{k=1}^{l+1} R_k^{c_{l+1-k}^{(j)}} = \prod_{k=1}^{l+1} R_k^{a_{l+1-k-j}} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i2\pi \sum_{k=1}^{l+1} \frac{a_{l+1-k-j}}{2^k}} \end{bmatrix}, l = 0, \ldots, 2n-1, j = 0, \ldots, n-1 \tag{4.17}$$

Thus, constant number a parametrizes each adder of the ΦMAC unit through Eq. (4.17), by determining the rotation angle of its doubly controlled rotation gates.

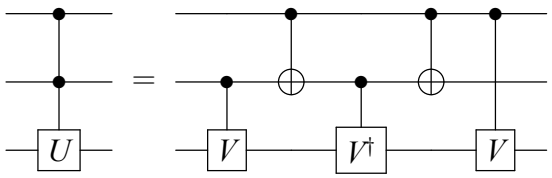
4.2 Depth-Optimized Fourier Multiplier/Accumulator - ΦMAC

In this Section we improve the ΦMAC unit which is one of the basic bulding blocks of the proposed modular exponentiation design. We modify the multiply/accumulate circuit reducing its depth from $O(n^2)$ to $O(n)$.

The circuit of Figure 4.12 is composed of n CCΦADD adders, each consisting of $2n$ doubly-controlled rotation gates with rotation matrix $A_l^{(j)}$ as described by Eq. (4.17), requiring a total of $2n^2$ such gates. All these gates have one common control qubit $|c\rangle$, which is the control qubit of the ΦMAC unit. Apparently, this fact restricts the execution of all these gates to be sequential and thus leads to a $O(n^2)$ depth. However, if we decompose each doubly controlled rotation gate into a network of single controlled gates [53] as depicted in Figure 4.13, we can re-arrange the rotation gates of the whole circuit so as to have a revised circuit with smaller depth.

Figure 4.14(i) depicts this decomposition applied on the j-th CCΦADDC adder of the ΦMAC unit, while 4.14(ii) depicts the re-arrangement that leads to the smaller depth. Matrices $V_l^{(j)}$ and $V_l^{(j)\dagger}$ of the controlled rotation gates in Figure 4.14 correspond to the decomposition of matrices $A_l^{(j)}$, and they are given by the following equations:

$$V_l^{(j)} = \sqrt{A_l^{(j)}} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi \sum_{k=1}^{l+1} \frac{a_{l+1-k-j}}{2^k}} \end{bmatrix}, l = 0, \ldots, 2n-1, j = 0, \ldots, n-1 \tag{4.18}$$



$$V = \sqrt{U}$$

Figure 4.13: Doubly controlled three-qubit gate decomposition to a network of two-qubit gates.

$$V_l^{(j)\dagger} = \sqrt{A_l^{(j)}}^{\dagger} = \begin{bmatrix} 1 & 0 \\ 0 & e^{-i\pi \sum_{k=1}^{l+1} \frac{a_{l+1-k-j}}{2^k}} \end{bmatrix}, l = 0, \ldots, 2n-1, j = 0, \ldots, n-1 \quad (4.19)$$

A closer look of the subcircuit of Figure 4.14(i) that corresponds to the adder of constant $a2^j$ (the subcircuit corresponding to qubits $|c\rangle, |x_j\rangle, |\varphi_{2n-1}\rangle, \ldots, |\varphi_1\rangle, |\varphi_0\rangle$) reveals that all the "first" $2n$ $V_l^{(j)}$ gates controlled by qubit $|x_j\rangle$ can be moved to the left of the subcircuit of Figure 4.14(ii), because they are all controlled by the same qubit $|x_j\rangle$ upon which a CNOT gate controlled by qubit $|c\rangle$ has acted an even number of times. This is equivalent to no CNOT gate acting. Similarly, all the "odd numbered" $2n$ CNOT gates that correspond to the decomposition of each $A_l^{(j)}$ gate can be replaced by exactly one CNOT gate affecting qubit $|x_j\rangle$ and controlled by qubit $|c\rangle$.

Next, all the $V_l^{(j)\dagger}$ gates controlled by qubit $|x_j\rangle$ can be moved exactly after the CNOT gate as shown in Figure 4.14(ii), because their control is done by the qubit $|x_j\rangle$, upon which a CNOT gate controlled by qubit $|c\rangle$ has acted an odd number of times. This is equivalent to only one CNOT gate. Also, the "even numbered" group of $2n$ CNOT gates corresponding to the decomposition of each $A_l^{(j)}$ gate are merged to a simple CNOT gate exactly after the grouping of the controlled $V_l^{(j)\dagger}$ gates. Finally, using the same arguments as before we can merge the "last" $2n$ $V_l^{(j)}$ gates controlled by qubit $|c\rangle$ at the right of the subcircuit of Figure 4.14(ii).

After these transformations, we can combine the n quantum adders CCΦADD in a highly parallel circuit as depicted in Figure 12 for the case $n = 3$. Figure 12 refers to the case of multiplying a three bit integer constant a with a three qubits quantum integer $|x\rangle$ and accumulating the resulting product into a six qubits quantum register. This circuit is built by successively connecting n CCΦADD blocks and exploiting the fact that all the controllable rotation gates commute. Furthermore, the last $V_l^{(j)}$ gates acting upon qubit $|\varphi_l\rangle$ can be merged (as long as they are all controlled by qubit $|c\rangle$) to a single controlled gate W_l, with rotation matrix:

$$W_l = \prod_{j=1}^{n-1} V_l^{(j)}, l = 0, \ldots, 2n-1 \quad (4.20)$$

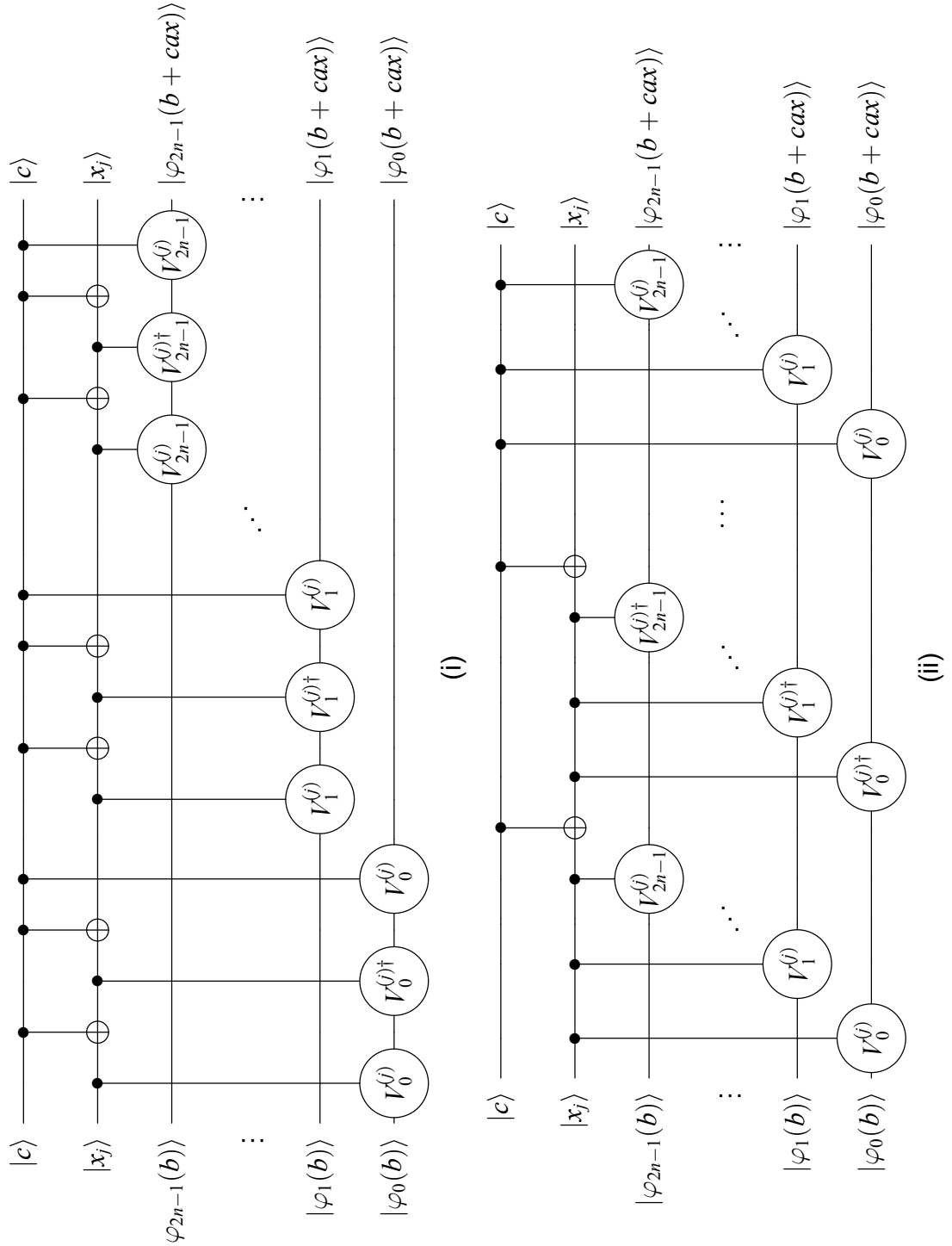


Figure 4.14: (i) The j^{th} ΦADD subcircuit of the ΦMAC, (ii) the rearrangement of the j^{th} ΦADD subcircuit after exploiting the decomposition of Figure 4.13.

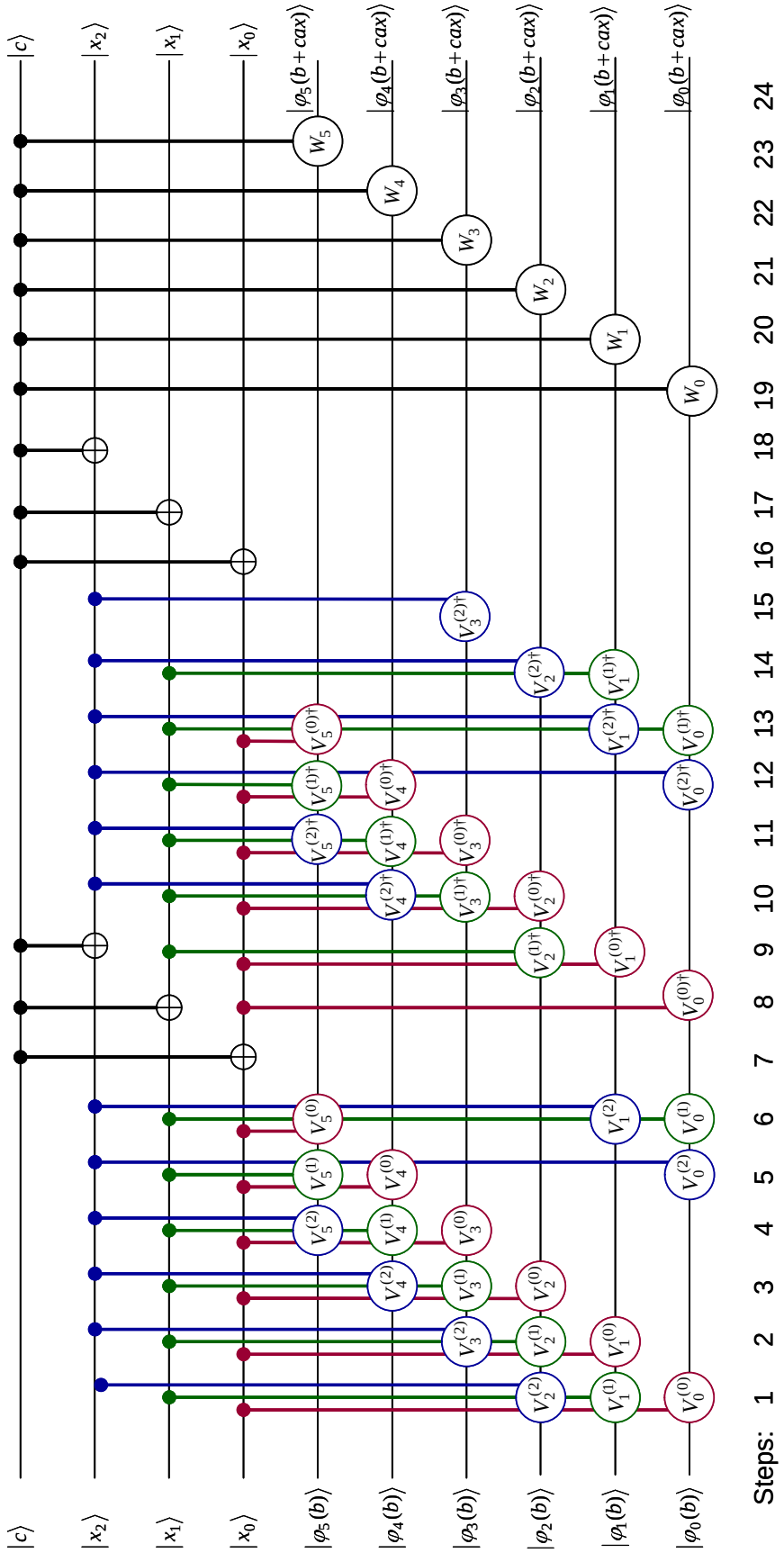


Figure 4.15: Fully decomposed and rearranged ΦMAC unit with linear depth of $8n$ for the case $n = 3$. In this case it requires $8 \cdot 3 = 24$ timesteps as shown in the figure. The rotation gates angles are determined by the constant a (see Eq. (4.18), (4.19) and (4.20)).

A circuit depth analysis for the ΦMAC unit of Figure 4.15 shows that if the two-qubit gates acted upon and controlled by different qubits operate in parallel, then for the "first" $V_l^{(j)}$ gates a total of $2n$ computation steps are required, for the "odd group' CNOT gates and the $V_l^{(j)\dagger}$, $3n$ more computations steps are required, for the "even group" CNOT gates n more computation steps are required, and for the W_l gates, $2n$ more computation steps are required. Consequently, the total computational steps required for the proposed implementation of the multiplier/accumulator unit is linear in size and has a depth of $8n$. The total quantum cost is $4n(n+1)$ gates. The proposed circuit uses exclusively two-qubit rotation gates, instead of three-qubit gates, making it more suitable for physical realizations [52, 134, 135, 136, 137]. Fault tolerance aspects of the circuit depth are not taken into account. This point is separately addressed in Section 4.6 and section 5.1.

A ΦMAC unit with no control is easily obtained by discarding the control qubit of Figure 4.12 and using simply controlled rotation gates, instead of two-qubit controlled gates. In this case, the described decomposition is not needed to achieve the depth reduction. A re-ordering of the gates shown in 4.12, so as n gates are applied at each instance, gives a depth of $2n$. The quantum cost for the uncontrolled ΦMAC is $2n^2$ gates.

4.3 QFT Divider by constant - GMΦDIV

The proposed modular exponentiation design does not use a modular adder to construct the multiplier/accumulator unit but it is rather based on simple adders. For this reason, we are forced to implement the modular operation after the multiplication by incorporating a divider module. Dividers are the most complex elementary arithmetic operation circuits in terms of computation time, but for our Shor's algorithm circuit design we can again take advantage of the fact that only divisions by the integer to be factored, N, are required. Integer N is constant throughout each quantum iteration of Shor's algorithm and thus a simpler division module suffices.

There are a few quantum dividers known, among them there are some [138, 139] which are suitable for multiplicative inversion in the Galois field GF(2^m) with depth of $O(n^3)$ and $O(nlog_2 n)$, respectively. Another divider suitable for integer division appears in [25], it is based in QFT and offers depth of $O(n^3)$. This divider receives two quantum integers and gives the quotient and the remainder. If one tries to convert it to a divider by constant then its depth can be reduced to $O(n^2)$.

We searched the literature for constant division classical algorithms that could offer improved time complexity over the general division algorithms. Various algorithms for classical division of an integer by constant have been appeared in the literature, such as those in [122] and [140]. In this section we describe a quantum version of an algorithm proposed by Granlund and Montgomery in [122]. This algorithm divides a $2n$ bits integer by an n bits constant integer and generates an n bits quotient and an n bits remainder, subject to the constraint that the quotient is less than 2^n. The algorithm can be easily modified to an unconstrained algorithm that divides an n bits integer by an n bits constant integer by simply resetting the upper n bits of the dividend. The algorithm in [122] utilizes multiplications, additions, logical operations such as shifting and bit selections. It has a constant time complexity. Table 4.1 shows the algorithm in pseudocode format as presented in [122].

This division algorithm takes as input the unsigned integer z (dividend) and divides it by the constant unsigned integer d (divisor) giving as results the quotient q and the remainder r. The three computation steps of the initialization (lines 1-3 of Table 4.1) are to be done once, as they depend only on the constant divisor d. These initialization steps are "hard-wired" in the quantum version of the algorithm and reflect a particular divisor which is the

Table 4.1: Granlund-Montgomery division-by-constant algorithm [122]. Comments, after //, in some of the lines show equivalent arithmetic operations.

	/* Declaration of input and output variables */		
	udword z;	//	Dividend (input)
	uword q;	//	Quotient (output)
	udword r;	//	Remainder (output)
	const uword d;	//	Divisor (constant)
	/* Initialization (given uword d, where $0 < d < 2^n$) */		
1:	uword $l = 1 + \lfloor(\log_2 d\rfloor$;	//	$2^{l-1} \leq d < 2^l$
2:	uword $m' = \lfloor(2^n(2^l - d) - 1)/d\rfloor$;	//	$m' = \lfloor(2^{n+1} - 1)/d\rfloor - 2^n$
3:	uword $d_{norm} =$ SLL$(d, n - l)$;	//	Normalized divisor $d \cdot 2^{n-l}$
	/* Start of main procedure */		
4:	uword $n_2 =$ SLL(HIGH$(z), n - l)$+SRL(LOW$(z), l)$;		
5:	uword $n_{10} =$ SLL(LOW$(z), n - l)$;		
6:	sword $n_1 = -$XSIGN(n_{10});		
7:	uword $n_{adj} = n_{10}+$AND$(-n_1, d_{norm} - 2^n)$;	//	$n_{adj} = n_{10} + n_1 \cdot (d_{norm} - 2^n)$
8:	uword $q_1 = n_2+$HIGH$(m' \cdot (n_2 + n_1) + n_{adj})$;		
9:	sdword $dr = z - 2^n \cdot d + (2^n - 1 - q_1) \cdot d$;	//	$dr = z - q_1 \cdot d - d, -d \leq dr < d$
10:	$q =$HIGH$(dr) - (2^n - 1 - q_1) + 2^n$;	//	Add 1 to quotient if $dr \geq 0$
11:	$r =$LOW$(dr)+$ AND $(d - 2^n,$ HIGH $(dr))$;	//	Add d to remainder if $dr < 0$

number to be factored. The computation steps of the main division procedure (lines 4-11) are executed whenever a new dividend must be divided by the constant divisor, that is at each iteration of the quantum part of Shor's algorithm for each new random number a. An explanation of the meaning of the variables and data types of the algorithm along with the various logical operations follows in Table 4.2. Also, the required number of ancillae is shown for each operation, so that it can be done reversibly.

The three shift operations (SLL, SRA, SRA) can be easily implemented reversibly (without discarding any bit) with the help of an ancilla register initialized with value 0. As shown in lines 4 and 5 of the algorithm, the result of the shifting must be added to a value 0 or added

Table 4.2: Explanation of the various logical operations and data types used in the classical version of the division by constant algorithm.

Operation Data Type	Meaning	Input (# of bits)	Output (# of bits)	Ancilla (# of bits)
SLL (x, i)	Logical left shift of x by i bits	n	n	n
SRA (x, i)	Arithmetic right shift of x by i bits	n	n	n
SRL (x, i)	Logical right shift of x by i bits	n	n	n
XSIGN (x)	-1 if $x < 0$, 0 if $x \geq 0$	n	1	1
AND (x, y)	Bitwise logical AND of x and y	n	n	0
HIGH (x)	Upper half of integer x	$2n$	n	0 or n
LOW (x)	Lower half of integer x	$2n$	n	0
uword	n bits unsigned integer			
sword	n bits signed integer			
udword	$2n$ bits unsigned integer			
sdword	$2n$ bits signed integer			

one after the other to a value 0, thus we can just select the desired bits to drive the control input of the generic ΦADD unit of Figure 4.9, while the QFT transformed input of the same ΦADD unit is the ancilla register. The original input to be shifted will remain intact in the initial register. Of course the ancilla register has to be set back to the zero value somehow at a later time when the shifted value has been used at a following computation step.

The `XSIGN` operation is simply a copying of the most significant bit (the sign bit) of a register holding a signed integer, to an ancilla qubit. Similarly, this ancilla has to be reset later.

The `AND` operations are implemented as the arithmetic operations shown in the comments of lines 7 and 11 of the algorithm of Table 4.1, that is additions on condition of the value of n_1 and dr, respectively.

The `LOW` operations at lines 4 and 5 of the algorithm are just the selection of the relevant bits to be "shifted" and then added by the QFT adders as described above. The third `LOW` in line 11 of algorithm is done implicitly as an addition shown in the comment of line 11.

The `HIGH` operation at line 4 is implemented with the help of an ancilla register, initially in value 0. The upper half bits are copied via CNOT gates to the ancilla register and later this ancilla must be reset to the zero value. The `HIGH` operation at line 11 is accomplished similarly to the `LOW` as the addition shown in the comment.

A quantum division by constant circuit implementing the algorithm of [122] is depicted in the two diagrams of Figures 4.16 and 4.17 (subsequently referred as Figures 4.16-4.17). A quantum circuit for this algorithm needs a few ancilla qubits because of the intermediate variables used in its classical counterpart, like m', d_{norm}, n_2, n_{10}, n_1, n_{adj}, q_1, dr and these ancilla qubits must be reset again (assuming initial zero state) at the end of the computation, since we want to reuse them for subsequent computations. Figures 4.16-4.17 show a quantum circuit that implements the algorithm of Table 4.1 for the case $n = 4$ and for a constant divisor $d = 5$. The extension for other sizes of n and different constant divisors d is straightforward; we refer to such division circuits as GMΦDIV.

4.3.1 Building blocks and registers of the quantum divider.

A generic GMΦDIV circuit will have a total of $7n + 1$ qubits, $5n + 1$ of which are the ancilla qubits. The GMΦDIV unit uses the following blocks and their inverses (inverses are noted with the same symbol with the thick bar on its left side instead of its right side):

- QFT for computing the quantum Fourier transform.
- Three types of adders ΦADD, that is adder with quantum integer, adder with constant and controlled adder with constant. Their inverses are simply the reverse circuit (signal flow from output to input) with the angles of rotation gates having the opposite sign.
- Multiplier/accumulator ΦMAC with no control and its inverse which is simply the reverse circuit with opposite sign angles in its rotation gates.
- CNOT, X (NOT) gates and SWAP gates (implied by the rerouting of the registers around the third ΦMAC unit).

The input qubits of the GMΦDIV circuit are grouped in a $2n$ qubits register (Reg0:Reg1), five n qubits registers (Reg2,. . .,Reg6), most of which are ancillae, and a single ancilla qubit (Aqbit) as shown in Figures 4.16-4.17. We give below a description of the purpose of each register:

- Reg0:Reg1: We distinguish two cases:
  1. In the generic divider case, where we divide an n qubits integer by an n bits constant, this register initially contains the dividend z in the qubits of its lower half, while the upper half is initially set to zero. This way we conform to the constraint that the quotient must be less than 2^n. At the end of the computation, this register will contain the remainder r in its lower half and zero in its upper half. The upper half acts essentially as an ancilla register.
  2. In the special case where we divide a 2^n qubits integer by an n bits constant, under the restriction that the quotient is known to be less than $2n$, both Reg0 and Reg1 will contain the upper and lower part of dividend, respectively. Since both cases of division differ only in the permitted type of initialization in registers Reg0 and Reg1, otherwise they have the same circuit network, we distinguish these cases by different symbols as shown later.
- Reg2: This register contains the quotient q at the end of the computation. It is initialized in the zero state.
- Reg3: Ancilla register holding the intermediate variable q_1. Initialized and end up in zero state.
- Reg4: Ancilla register used to successively hold the values n_2, $n_2 + n_1$, n_2, 0, n_1, 0. Initialized and end up in zero state.
- Reg5: Ancilla register used to hold the value $\texttt{HIGH}(m'(n_2 + n_1) + n_{adj})$. Initialized and end up in zero state.
- Reg6: Ancilla register used to successively hold the values n_{10}, n_{adj}, $\texttt{LOW}(m'(n_2+n_1)+ n_{adj})$.
- Aqbit: Ancilla qubit used to hold the sign n_1. Initialized and end up in zero state.

Next a brief description of the whole circuit is given. Part of the circuit is dedicated to forward computations, while another part is dedicated to “restoring” the ancilla qubits back to the zero state, so as the whole circuit is reversible without generating any garbage. The latter part stands out as the gray shaded area. We begin describing the forward computations. For simplicity we refer to the values of the registers as integers regardless of being integers in the computational basis or being their respective values in the quantum Fourier transform domain, in other words we ignore in the description the various QFT blocks present in the register buses. In essence QFT and inverse QFT blocks are needed at the buses before and after, respectively, each arithmetic block that process integers in the Fourier domain. These are: one of the bus of each adder (ΦADD, CΦADDC) and the accumulator bus of each ΦMAC block. Whenever two adders are connected one after the other in order to successively add different values to a quantum integer, there is no need to transform from the QFT domain back to the computational basis and then again perform the forward transform. The schematic diagram is adequate to distinguish when a bus has an integer in the computational basis or in the Fourier domain.

4.3.2 Forward computations of the quantum divider.

Initialization steps (Lines 1, 2, 3, of the Algorithm)

As noted before, the circuit of Figures 4.16-4.17 refers to the case of $d = 5$. The initialization steps in lines 1, 2 and 3 of the algorithm of Table 4.1 result in the following values of

$l = 3$, $d_{norm} = 10$ (and $d_{norm} - 2^n = -6$) and $m' = 9$. The initialization is computed "offline" and these values are "hardwired" in the circuit, i.e. $m' = 9$ is hardwired as the constant parameter of the first ΦMAC unit, $d_c = d_{norm} - 2^n = -6$ is hardwired as the constant parameter in some of the CΦADD units and $l = 3$ is hardwired in the logical shift section. The logical shift sections are indicated by dashed arrows pointing from the logical operation to the adders performing the shift.

Computation of n_2, n_{10}, n_1. (Lines 4, 5, 6 of Algorithm)

The first operation in the diagram is to compute the value of $\mathtt{SLL}(\mathtt{HIGH}(z), n-l) = (z_6 z_5 z_4 0)_2$ and then added to Reg4. Indeed, we select these three qubits of from Reg0 along with a zero qubit from ancilla Reg5 and add them to Reg4 through the use of a ΦADD unit. (This operation is to be done only in the special case -2- of the divider mentioned above where both halves of the dividend may contain non-zero values. In the general case -1- where the upper half contains zero values this operation becomes gratuitous). The next adder affecting Reg4 is the one that selects three zero qubits from Reg5 as most significant qubits and the qubit z_3 from Reg4 to add them, that is to add $\mathtt{SRL}(\mathtt{LOW}$(z),l)$=$ $\mathtt{SRL}(\mathtt{LOW}(z), 3) = (000 z_3)$. This way we have computed in Reg4 the quantity n_2. In the same manner we compute in Reg6 the value of n_{10}. Having computed n_{10}, it is straightforward to compute the sign n_1 in the Aqbit by using a CNOT gate controlled by the most significant qubit of Reg6 and targeting the Aqbit.

Computation of $_{nadj}$, q_1 (Lines 7,8 of Algorithm)

Now, we add the constant d_c (that is $d_{norm} - 2^n$) to Reg6 (which already has the value n_{10}) conditioned on the value of n_1, thus forming the quantity n_{adj}. Also, we add n_1 to Reg4, forming the value $n_2 + n_1$. Now the first ΦMAC(m') unit has at its accumulator input (high qubits at Reg5, low qubits at Reg6) the value n_{adj} and has at its multiplicand input (Reg4) the value $n_2 + n_1$. Thus, the ΦMAC outputs have Reg4=$n_2 + n_1$ and (Reg5:Reg6)=$m'(n_2 + n_1) + n_{adj}$. By copying with CNOT gates the content of Reg5 to Reg2 we have at Reg2 the value $\mathtt{HIGH}(m'(n_2 + n_1) + n_{adj})$. Then we can add to Reg2 the value of Reg4, which is still $n_2 + n_1$ and then we subtract n_1 leaving as end result the desired $q_1 = n_2 + \mathtt{HIGH}(m'(n_2 + n_1) + n_{adj})$.

Computation of d_r (Line 9 of Algorithm)

The second ΦMAC(-d) unit has at the accumulator input (Reg0:Reg1) the dividend z and at the multiplicand input (Reg2) the value q_1, forcing the output (Reg0:Reg1) to be $z - d \cdot q_1$ and after subtracting the constant d becomes $d_r = z - d \cdot q_1 - d$.

Computation of the quotient q (Line 10 of Algorithm)

Now that we have computed the d_r value we are ready to proceed to the last steps of the algorithm of Table 4.1 doing a sign check to the quantity dr as these steps suggest and this is equivalent to checking the most significant qubit of Reg0:Reg1. For this reason we add the integer 1 to the Reg2 conditionally on the inverted most significant qubit of Reg0:Reg1. This way we have formed at the Reg2 the quotient q, because if $d_r \geq 0$ then its inverted most significant qubit will be 1 thus adding the value 1 to q_1, otherwise it adds nothing.

Computation of the remainder r (Line 11 of Algorithm)

Meanwhile q_1 has been copied to Reg3 by CNOT gates and becomes the multiplicand input of the third ΦMAC(d) unit. Its accumulator register Reg0:Reg1 has become again $z - d \cdot q_1$ after the addition of d, and the end result for the accumulator register after the third ΦMAC(d) is to restore its initial value of the dividend z. The last ΦMAC($-d$) unit

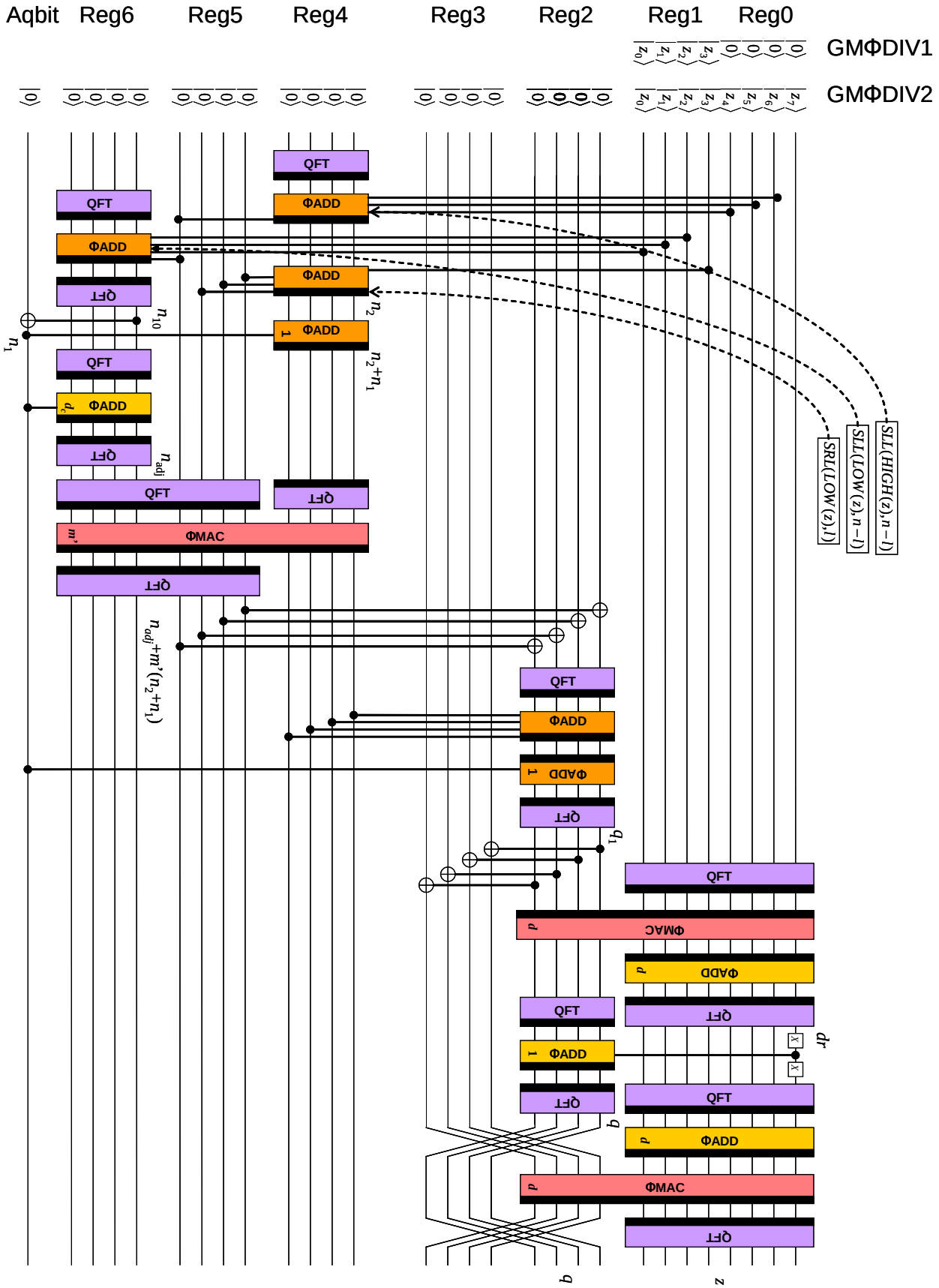


Figure 4.16: The GMΦΦDIV circuit (first part) for 8 or 4 qubits dividend and constant divisor $d = 5$. Intermediate variables are shown at places where they have been computed (in computational basis or QFT transformed).

Continued from previous Figure

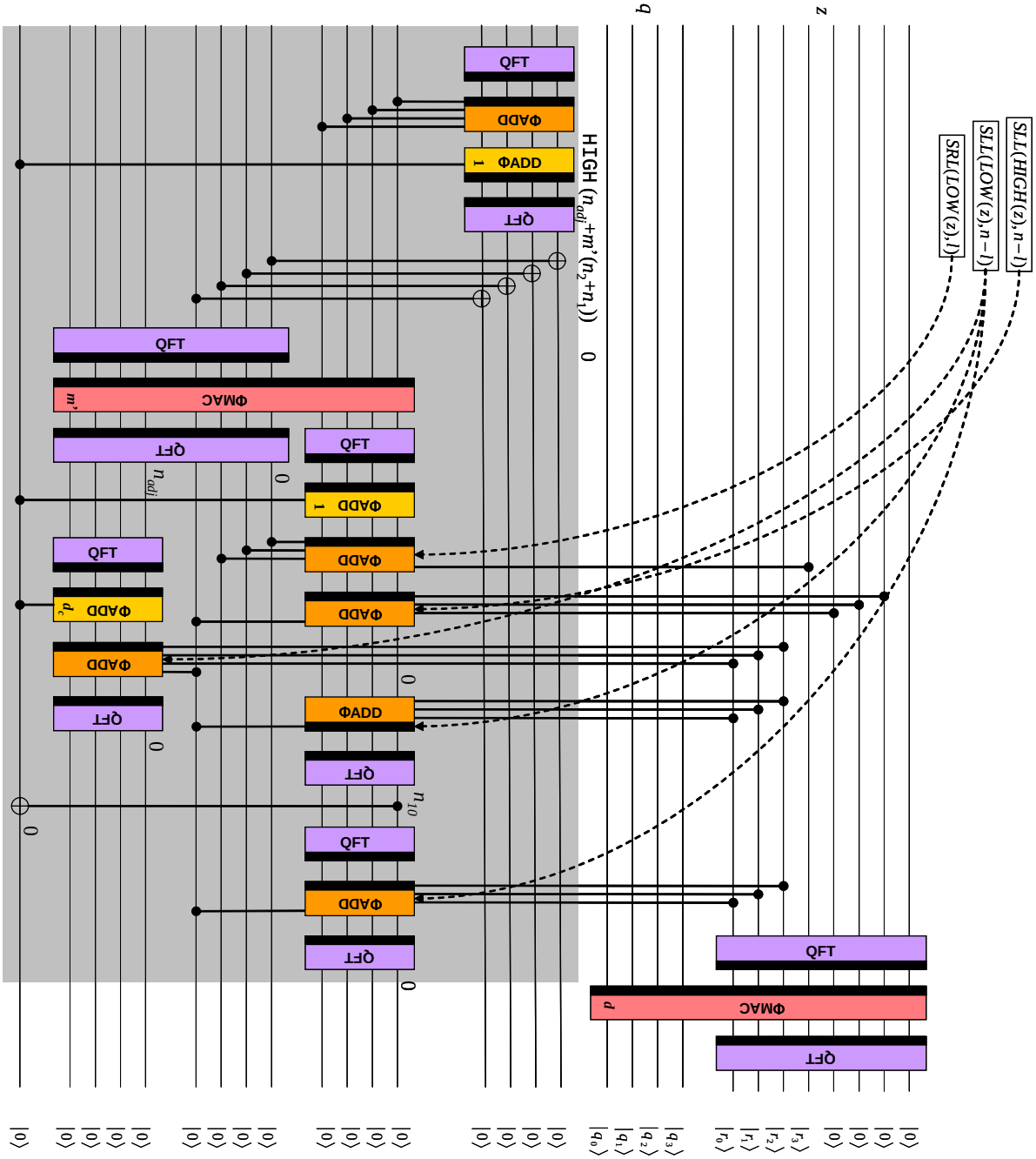


Figure 4.17: The GMΦDIV circuit (second part) for 8 or 4 qubits dividend and constant divisor $d = 5$. Shaded areas indicate computations for resetting the ancilla qubits.

acts on this register again, subtracting the product $q \cdot d$, giving thus the remainder r at its lower half (Reg1) while its upper half (Reg0) becomes zero, thus completing the forward computations. Note that in both the generic case (dividend of n qubits) and in the special case (dividend of $2n$ qubits subject to the constraint that the quotient is known to be less than 2^n) the upper half (Reg0) becomes zero.

4.3.3 Ancilla Resetting.

It remains to show the computations that reset the ancilla qubits. The ancilla qubits to be reset are those in registers Reg3, Reg4, Reg5, Reg6 and the single qubit Aqbit.

- Reg3: The first reset occurs at Reg3 which contains q_1. This is accomplished if we subtract from it the quantity $n_2 + n_1$ (stored in Reg4) and adding n_1 (stored in Aqbit) leaving a value of $\texttt{HIGH}(m'(n_2 + n_1) + n_{adj})$. But this value is already stored in Reg5, consequently a "qubitwise" CNOT operation controlled by Reg5 and targeted to Reg3 effectively resets Reg3.
- Reg5: Then we reset Reg5 through the usage of an ΦMAC($-m'$), that is we add to the accumulator registers (Reg5:Reg6) containing $m'(n_2 + n_1) + n_{adj}$ the quantity $-m'(n_2 + n_1)$ leaving the result n_{adj}. But n_{adj} is an $\texttt{uword}$ and consequently the upper register (Reg5) becomes zero. The lower accumulator register (Reg6) contains n_{adj}.
- Reg4: Next we reset Reg4 by subtracting from it the quantities n_2 and n_1. Quantity n_{10} is formed again easily from Reg0:Reg1 which now contains again the dividend z, by using the same method of shifting and additions we used in the forward computations. The value of n_{10} is needed to reset the Aqbit as described below. Then by substracting n_{10} we end up in the zero value in Reg4.
- Reg6: This register, which contains n_{adj}, is reset by subtracting the constant d_c conditioned on n_1 and subtracting n_{10}.
- Aqbit: To reset qubit Aqbit we use the value n_{10} formed in Reg4 at the step just before we reset it for the second time and use its most significant qubit as the control of the CNOT gate targeting the Aqbit.

Note that the proposed QFT divider does not include any controlling qubit, but this is not required for the construction of the modular multiplier as it will be shown. A symbol for the GMΦDIV for the generic case, that is dividend and divisor of n bits wide, is shown in Figure 4.18 with the name GMΦDIV1. This symbol shows only the qubits used for input (dividend with the upper qubits having initial value of zero) and outputs (quotient and remainder),

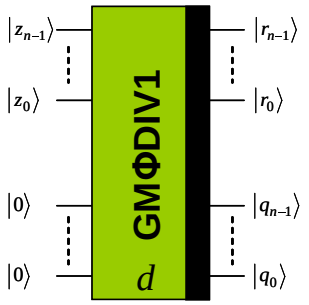


Figure 4.18: The symbol of the GMΦDIV1. It receives an n qubits dividend to divide it by the constant divisor d of n bits. The underlying circuit uses $7n+1$ qubits, including $5n+1$ ancilla qubits which are not shown.

leaving hidden the other ancilla qubits. In section 4.7 a modification of the circuit is given that converts it to a controlled divider.

Regarding the GMΦDIV circuit, a thorough depth analysis leads to a depth of $74n-6$ (the initial depth of $244n-8$ reported in [141] was an overestimation mainly due to miscalculations in QFT depths and ignoring the fact that uncontrolled ΦMAC are used). In this study we have take in account the following facts. Each QFT unit has a depth of $2n-1$ (through parallelization of its rotation gates), the constant adder ΦADD has a depth of 1, the controlled CΦADD and the adder ΦADD have a depth of n and the uncontrolled ΦMAC as analyzed in the previous section has a depth of $2n$. In this depth analysis we have also taken into account that many of the blocks can be executed in parallel. Total number of gates used in the GMΦDIV is $44n^2+76$. Fault tolerance and implementation aspects are discussed in a separate paragraph in section 4.6 and in section 5.1.

4.4 Generic Modular Multipler/Accumulator and Modular Multiplier

In this Section we employ the blocks described in the previous subsections to build two quantum circuits; the generic QFT based controlled modular multipler/accumulator and then we proceed to a generic quantum controlled modular multipler which can be used as the basic building block for the modular exponentiation circuit as shown in Figure 4.5.

4.4.1 Generic QFT Modular Multiplier/Accumulator - ΦMAC_MOD1

Controlled modular multipliers/accumulators can be built using the ΦMAC and GMΦDIV1 units as their basic blocks. Such blocks can realize Eq. (4.21), and then can be used as described in subsection 4.4.2 to realize a controlled modular multiplier as that of Eq. (3.26). When the multiplicand input of the ΦMAC unit is n qubits wide its accumulator is $2n$ qubits wide. These $2n$ qubits must be fed as input to the dividend input of the GMΦDIV1 to compute the modulo N result. Therefore, the size of the GMΦDIV1 unit must such that it can receive a dividend of $2n$ qubits, which means that the dividend input must be $4n$ qubits wide and its upper half $2n$ qubits should be zero. Therefore the required ancillae number for the GMΦDIV1 grows from $5n+1$ to $10n+1$. Note that a simple interconnection of the two units,ΦMAC and GMFDIV1, in succession is not adequate to give a result that follows Eq. (4.21) because the GMΦDIV1 unit gives at its outputs both the remainder which is the useful part of the computation and the quotient which contains garbage qubits that must be coherently reset to keep the garbage-less reversibility of the circuit and then reused at a later time.

$$\Phi MAC\_MOD1_{a,N}(|c\rangle|y\rangle|0\rangle) = |c\rangle|y\rangle|cay \bmod N\rangle \tag{4.21}$$

The proposed architecture of the generic controlled modular multiplier/accumulator by constant, named ΦMAC_MOD1, is shown in Figure 4.19. This circuit diagram shows $6n+1$ qubits, but there are $10n+1$ more “hidden” qubits in the GMΦDIV1 symbol (the ancillae of the first divider can be reused in the second one, as the second one cannot be operated in parallel to the first) which we don’t show for the clarity of Figure 4.19. The input lines with a slash symbol in the figure correspond to “buses” of n qubits. It is straightforward for the reader to understand the equivalence between the symbols with individual qubits presented in previous sections with the ones in Figure 4.19 having buses as inputs and outputs. Apart from the two basic blocks and the QFT units there is a group of CNOT gates in this diagram. These units operate on a “qubitwise” basis, i.e. the first qubit of the controlling bus controls the CNOT of the first qubit of the target bus, etc. An analysis of this circuit follows for the two cases of the controlling qubit $|c\rangle=|1\rangle$ and $|c\rangle=|0\rangle$. As in the previous section, we will not care of the various QFT blocks present in the register buses

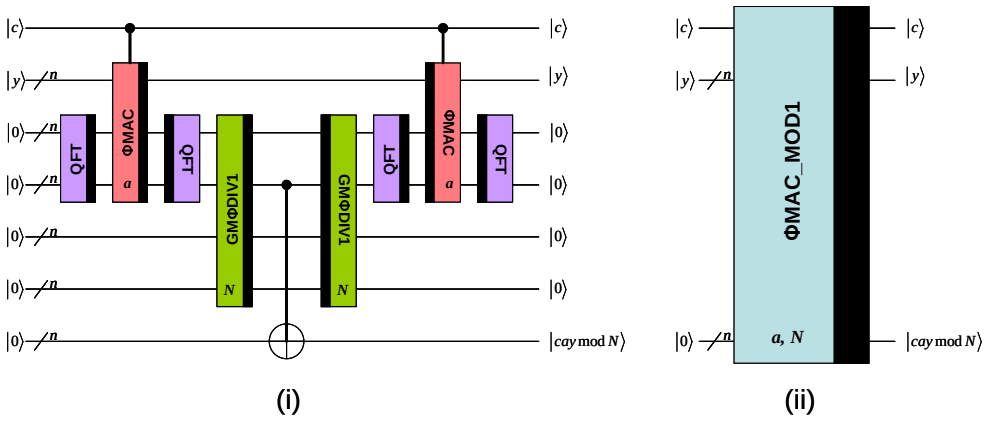


Figure 4.19: (i) The full diagram of the generic controlled modular multiplier/accumulator ΦMAC_MOD1 and (ii) its symbol. A total of $6n+1$ qubits are shown, but there are $10n+1$ more ancilla qubits not shown in the GMΦDIV1 symbol.

and we give the integers values as if they were not QFT transformed.

For the case of $|c\rangle = |1\rangle$ both ΦMAC units are enabled. Initially all the n ancilla qubit buses are in state $|0\rangle$ while the value y, the number to be multiplied by the constant a, is fed to the multiplicand input of the first ΦMAC(a) unit. After the operation of this ΦMAC(a) unit the multiplicand qubits are still $|y\rangle$ while the $2n$ qubits of its accumulator go to state $|0+ay\rangle = |(ay)\rangle_U|(ay)\rangle_L$, where subscripts U and L denote upper and lower register qubits, respectively. These $2n$ qubits feed the dividend input of the GMΦDIV1(N) giving as outputs the remainder $|(ay \bmod N)\rangle_U|(ay \bmod N)\rangle_L = |0\rangle|(ay \bmod N)\rangle_L$ and the quotient $|q\rangle_U|q\rangle_L$, where N is again the number to be factored. It is assured that the upper $2n-n=n$ qubits of the remainder are zero because N is n bits wide. The remainder is copied with the aid of the CNOT gates group to the bottom n qubits bus and then becomes the output of the ΦMAC_MOD1 block. The original remainder bus is fed to an inverse GMΦDIV1(N) unit whose other input is the quotient $|\lfloor (ay)/N \rfloor\rangle$ computed by the first GMΦDIV1(N) unit. Such an inverse divider can be easily designed by reversing the signal flow of the normal divider and setting the angle of every rotation gate of the inverted GMΦDIV1 to the opposite of the original angle. By feeding the quotient and the remainder in an inverse GMΦDIV1 we have available at its output the input that would give this remainder and quotient, that is $|ay\rangle$ at the top $2n$ qubits and $|0\rangle$ at the bottom $2n$ qubits (an inverse divider effectively becomes a multiplier). The second ΦMAC($-a$) unit, which is the inverted ΦMAC(a), takes as multiplicand input the state $|y\rangle$ and as accumulator input the output of the previous inverted GMΦDIV1(N), $|ay\rangle$. Similarly, the outputs of this inverted ΦMAC(a) are the inputs that would lead a normal ΦMAC unit to give as outputs the inputs being fed to the inverted, that is $|y\rangle$ at the top n qubits and $|0\rangle$ at the lower $2n$ qubits. This way we achieved to clear the useless, in our application, quotient. At the same time, we have the desired remainder $|(ay \bmod N)\rangle$ available along with the initial input $|y\rangle$.

The case of setting the control qubit $|c\rangle = |0\rangle$ is simpler than the previous one. Both the ΦMAC units are disabled and they simply pass their inputs unmodified to their outputs. Also, the dividend input of the first divisor is zero. Therefore, input $|y\rangle$ traverses the circuit through the ΦMAC units while the ancilla buses remain in the zero state.

4.4.2 Generic QFT Modular Multiplier - ΦMUL_MOD1

The last step in the construction of the modular multiplier required by Shor's algorithm is to reset the state $|y\rangle$ at the output of the modular multiplier/accumulator so that we can successively connect several modular multiplier units as shown in Figure 4.5. Figure 4.20

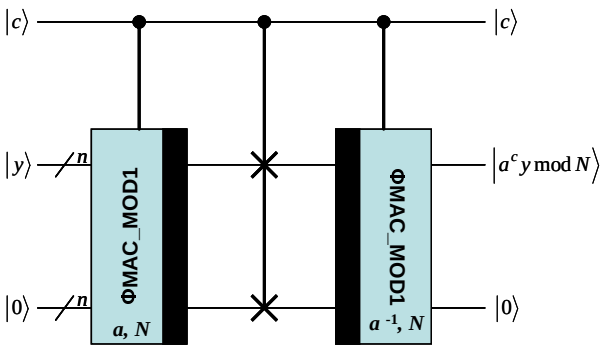


Figure 4.20: Generic modular multiplier ΦMUL_MOD1. The circuit requires $16n+1$ qubits, where $14n$ of them are ancillae hidden in the blocks of the lower levels of hierarchy.

shows a method for clearing the undesired $|y\rangle$ [24, 62, 61]. Two ΦMAC_MOD1 units are used in this diagram, with a block of controlled SWAP gates (Fredkin gates). The second ΦMAC_MOD1 unit is a reverse unit with multiplication parameter a^{-1}, where the inverse a^{-1} is defined with respect to the operation of multiplication modulo N, that is it must hold $a \cdot a^{-1}(\mathrm{mod}N) = 1$. Such an inverse always exists, because the randomly picked number a is selected based on the restriction that it must be co-prime with N.

The analysis of this circuit for the case of $|c\rangle = |0\rangle$ is very simple as both the ΦMAC_MOD1 units and the CSWAP gates are disabled. Thus, input state $|y\rangle$ remains unmodified and passes to the output, while all the ancilla qubits remain in the zero state. In the case of $|c\rangle = |1\rangle$, the first ΦMAC_MOD1(a, N) unit gives as result the input $|y\rangle$ and the remainder $|r\rangle = |ay \bmod N\rangle$. This remainder is then fed, through the CSWAP gates, to the multiplicand input of the second ΦMAC_MOD1(a^{-1}, N) unit, while the accumulator input of this second unit is $|y\rangle$. This way the multiplicand output of ΦMAC_MOD1(a^{-1}, N) becomes the remainder $|r\rangle$ while the accumulator becomes $|y - a^{-1}(ay \bmod N) \bmod N\rangle = |y - y\rangle = |0\rangle$.

Combining the two case of $|c\rangle = |0\rangle$ and $|c\rangle = |1\rangle$ we have the transformation :

$$\Phi MUL\_MOD1_{a,N}(|c\rangle|y\rangle|0\rangle) = |c\rangle|a^c y(\mathrm{mod}N)\rangle|0\rangle \tag{4.22}$$

which has exactly the same form as Eq. 3.26, if the ancilla qubits are not taken into account. The final result is that we can combine many ΦMUL_MOD1 units of Figure 4.20 as shown in Figure 4.5 to build a quantum modular exponentiation circuit.

4.5 Optimized Modular Multipler/Accumulator and Modular Multiplier

Exploitation of the specific application where the modular multiplier is to be used can be advantageous in terms of both depth and space requirements. The following paragraphs present optimized versions of the modular multiplier/accumulator and modular multiplier.

The second version of the modular multiplier / accumulator, which we denote as ΦMAC_-MOD2, applies to Shor's factorization algorithm. As shown in Figure 4.5, each modulo N multiplier unit takes as input the output of its previous unit which is again a modulo N multiplier block (or the integer 1 for the first unit) and thus this input is always less than N. This input is to be multiplied by an integer which is again always less than N. Therefore, the product of these two integers, being less than N^2, has to be divided by N to calculate the remainder. The quotient of this division is of course again less than N. Taking as $n = \lceil \log_2 N \rceil$ the number of qubits for the division circuit we can see that for this specific case the quotient is less than 2^n. In other words, the restriction imposed for the operation of the Granlund-Montgomery division algorithm holds. For this reason we can use the

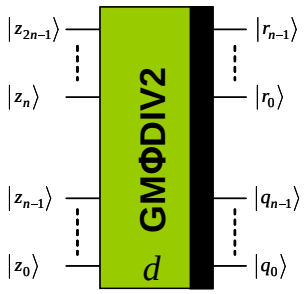


Figure 4.21: Symbol of GMΦDIV2 that receives a dividend of $2n$ qubits, subject to the constraint that the quotient is less than 2^n. The underlying circuit uses $7n+1$ qubits.

division circuit of Figures 4.16-4.17 with a size of only n bits, that is a dividend of $2n$ qubits (the upper half is not necessary to be zero) and quotient and remainder sizes of n qubits, instead of using the double sized generic divider of the previous section.

We introduce a new symbol for the same divider in Figure 4.21, merely reflecting the fact that all of the $2n$ qubits are reserved for the dividend, in contrast to the symbol of Figure 4.18 where half of them where set to zero in order to comply with the restriction of the quotient being less than 2^n. All the other internal aspects of the GMΦDIV2 are the same as of GMΦDIV1. This second version of the divider is to be used exclusively in a Shor's quantum algorithm architecture where the quotient is expected to be always less than 2^n, while the first version of the divider can be used whenever a general quantum divider by constant is needed.

The proposed architecture of the second version of controlled modular multiplier / accumulator by constant, named ΦMAC_MOD2, is shown in Figure 4.22. The circuit diagram shows a total of $4n+1$ qubits but there are $5n+1$ more "hidden" qubits in the GMΦDIV2 symbols which are not shown for the shake of clarity of Figure 4.22. Again, the input lines with a slash symbol in the figure correspond to buses, each bus consisting of n qubits. Using similar arguments as in the previous section we give a brief analysis of this circuit.

For the case of $|c\rangle = |1\rangle$ both ΦMAC units are enabled. The accumulator register output of the first ΦMAC(a) unit is in state $|ay\rangle_U|ay\rangle_L$, while its multiplicand register still holds the multiplicand $|y\rangle$. The product $|ay\rangle_U|ay\rangle_L$ is then fed to the first GMΦDIV2(N) to produce the remainder (upper bus of GMΦDIV2(N)) and the quotient (lower bus of GMΦDIV2(N)). The remainder is copied to the bottom bus which is the output ΦMAC_MOD2 circuit. The quotient and the original remainder buses are fed to an inverted GMΦDIV2(N) unit, giving again $|ay\rangle_U|ay\rangle_L$ at its output. The second (inverted) ΦMAC(a) unit has then at its multiplicand input the $|y\rangle$ and at its accumulator input $|ay\rangle_U|ay\rangle_L$, setting its accumulator out-

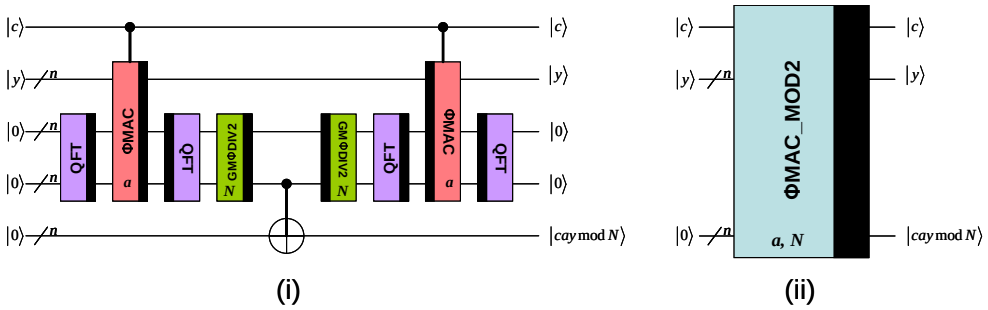


Figure 4.22: The optimized controlled modular multiplier/accumulator ΦMAC_MOD2 and its symbol. A total of $4n+1$ qubits are shown in this figure, but there are $5n+1$ more ancilla qubits not shown in the GMΦDIV1 symbol.

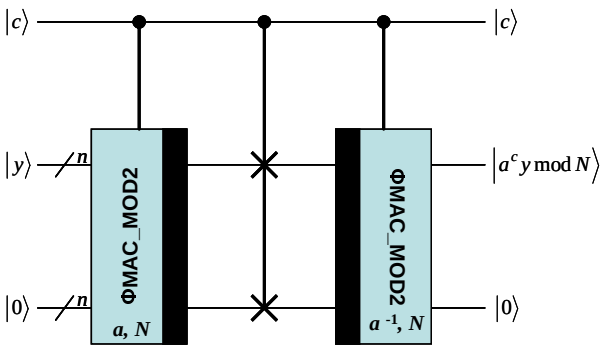


Figure 4.23: Optimized modular multiplier ΦMUL_MOD2. The circuit requires the $4n+1$ qubits shown in the diagram, plus $5n+1$ ancilla qubits hidden in the divider units.

put to state $|0\rangle$. Now, as in the case of ΦMAC_MOD1, we have the desired remainder $|(ay \bmod N)\rangle$ available along with the initial input $|y\rangle$. Similarly, the case of $|c\rangle = |0\rangle$ leads the top qubits bus to have the initial input $|y\rangle$, while all the other buses are set to the zero state.

A design of a modular multiplier based on the optimized multiplier/accumulator ΦMAC_-MOD2 is shown in Figure 4.23. This design is very similar to the generic case of subsection 4.4.2 using the generic ΦMAC_MOD1 unit and for this reason we don't analyze the circuit.

4.6 Complexity Analysis

In this section we analyze the depth (speed), width or space (the required number of qubits) and the quantum cost (total number of single or two-qubit gates) of a quantum modular exponentiation circuit, when implemented using either the ΦMUL_MOD1 unit, or the optimized ΦMUL_MOD2 unit. We also compare the two proposed designs with other designs proposed in the literature, in terms of depth and width. The derived complexities are with respect to $n = \lceil \log_2 N \rceil$, the bits size of the factorizable integer N.

We begin the depth analysis for the case of the ΦMUL_MOD1 controlled modular multiplier as top level unit for the modular exponentiator. The ΦMUL_MOD1 block consists of the two ΦMAC_MOD1 units and controlled SWAP gates (Fredkin), while each ΦMAC_MOD1 unit consists of the units QFT, ΦMAC, GMΦDIV1, whose depths are already analyzed, and CNOT gates. Figure 4.19 indicates that none of these units can operate in parallel, because each one gets its inputs from the previous one, so the depth of this modular multiplier/accumulator is merely the sum of each unit's depth. We note here that the depth of the CNOT unit used in ΦMAC_MOD1 is 1 since the gates of this unit can operate in parallel and the depth of the CSWAP unit used in ΦMUL_MOD1 is about $5n$ (a CSWAP gate consists of two CNOT gates and a Toffoli gate). The size of the four QFT units used is $2n$ qubits and the size of the two GMΦDIV1 units is for $2n$ bits dividend even if the divisor N is n bits wide, therefore the depths of the QFT and GMΦDIV1 units used in the ΦMUL_MOD1 unit are $2 \cdot 2n - 1 = 4n - 1$ and $74 \cdot 2n - 6 = 148n - 6$, respectively.

In Table 4.3 we summarize the depth of each unit used in the modular multiplier unit (we have take into account that each modular multiplier unit uses two multiplier/accumulator units), the number of units of each type that are used, the depth contribution of each type of unit and finally we calculate the total depth for the controlled modular multiplier ΦMUL_MOD1 which is $661n - 30$. Since the complete modular exponentiator circuit consists of $2n$ ΦMUL_MOD1 blocks successively connected in series, its depth is about $2n \cdot 661n \approx 1300n^2$. Discrepancies between Table 4.3 and the respective Table of [141] are due to miscalculations in the GMΦDIV unit (both in depth and cost) and miscalculation of

Table 4.3: Units used in the ΦMUL_MOD1 design, depth of each unit, number of gates in each unit, number of units used for each type, gates contribution and depth contribution of each unit type to the total quantum cost and depth of the modular multiplier.

Unit	Depth/unit	Cost/unit	# of units	Cost	Depth
QFT($2n$)	$4n-1$	$2n^2+3n$	8	$16n^2+24n$	$32n-8$
ΦMAC(n)	$8n$	$4n^2+4n$	4	$16n^2+16n$	$32n$
GMΦDIV1($2n$)	$148n-6$	$176n^2+152n$	4	$704n^2+608n$	$592n-24$
CNOT(n)	1	n	2	$2n$	2
CSWAP(n)	$5n$	$5n$	1	$5n$	$5n$
Total				$736n^2+655n$	$661n-30$

the QFT cost which was overestimated.

Concerning the number of qubits required for the complete modular exponentiator circuit (ΦEXP1), we see that in the case of the ΦMUL_MOD1 unit, it requires the same number of qubits as the controlled modular multiplier, that is $6n+1$ qubits plus the hidden ancilla bits of the GMΦDIV1 not shown in Figure 4.19. The ancilla qubits for a GMΦDIV1 unit of size $2n$ is $10n+1$ qubits, so the total qubits required for the circuit is $16n+2$.

The quantum cost for the ΦMUL_MOD1 modular multiplier calculated from Table 4.3 is about $750n^2$, and thus the cost of the first modular exponentiation circuit ΦEXP1 is about $1500n^3$.

A similar depth analysis for the second, optimized design of the complete modular exponentiator ΦEXP2, which utilizes the ΦMUL_MOD2 is presented in Table 4.4. When the optimized modular multiplier is used, we get an improved depth for ΦEXP2 of about $2n \cdot 365n \approx 700n^2$, that is about half the depth when using the generic ΦMUL_MOD1 multiplier.

However, the most important gain in using the ΦMUL_MOD2 is that the required qubits are only the $4n+1$ qubits shown in Figure 4.22 plus the $5n+1$ qubits hidden inside the symbol of GMΦDIV2, that is a total of $9n+2$, a significant improvement over the $16n+2$ qubits if using the ΦMUL_MOD1 modular multiplier.

Similarly, the quantum cost for the ΦMUL_MOD2 modular multiplier is about $200n^2$, hence the quantum cost for the modular exponentiator circuit ΦEXP2 is about $400n^3$.

We compare the two proposed designs to various quantum modular exponentiation circuit designs found in the literature. Table 4.5 shows the abbreviated names of the basic

Table 4.4: Units used in the ΦMUL_MOD2 design, depth of each unit, number of gates in each unit, number of units used for each type, gates contribution and depth contribution of each type of unit to the total quantum cost depth.

Unit	Depth/unit	Cost/unit	# of units	Cost	Depth
QFT($2n$)	$4n-1$	$2n^2+3n$	8	$16n^2+24n$	$32n-8$
ΦMAC(n)	$8n$	$4n^2+4n$	4	$16n^2+16n$	$32n$
GMΦDIV2(n)	$74n-6$	$44n^2+76n$	4	$176n^2+304n$	$296n-24$
CNOT(n)	1	n	2	$2n$	2
CSWAP(n)	$5n$	$5n$	1	$5n$	$5n$
Total				$208n^2+351n$	$365n-30$

building blocks for each exponentiation circuit as given in subsection 2.2, a description of its main iterated building block, the number of qubits required for the full modular exponentiation circuit and an estimation of its depth. The two circuits proposed in this paper are referred in Table 4.5 as ΦEXP1 and ΦEXP2. The design ΦEXP1 is the modular exponentiation circuit built from the generic components GMΦDIV1 and ΦMAC_MOD1 and has inferior performance to the ΦEXP2 because it doesn't exploit the property mentioned in Section 4.5. It appears to the comparison table just for reference.

Regarding the circuit depth, it is referred to the depth of single qubit gates or two qubits gates. Whenever three-qubit gates are encountered (e.g. Toffoli gates) their depth is rough approximated to an equivalent depth of one-qubit or two-qubits gates by assuming each three-qubit gate can be replaced by five one-qubit or two-qubits gates. Not all designs of the literature provide a full circuit; thus some estimations are rough and they are based on [129, 132] and our assumptions. For example to calculate the depth of a full exponentiation circuit based on an particular adder if a full circuit for the exponentiation operation is not given, we made the assumption that a modular adder for the exponentiation circuit needs five normal adders to be built as described in Section 4.1. At this point we have to warn that some depths referred by some authors are not to be taken as is, as these authors tend to make their calculation not by counting single qubit, two qubit gates and converting the depth of the Toffoli gates to an equivalent depth, but they rather group together various gates and count computation steps for each group. Also, in some of the previous works the depth is calculated as two qubit interactions, e.g. an arbitrary sequence of gates on two adjacent qubits is measured as having depth 1. This assumption, depending on the specific physical implementations, may not hold. For the above reasons (not full or detailed circuit given, adjacent qubit gates depth assumption), some entries of Table 4.5 have depth measures in $O(\cdot)$ notation instead of expressions with leading order constants.

Similar remarks apply for the estimation of the width (number of qubits) and especially the quantum cost (total number of gates) of the circuits.

We can see in Table 4.5 that the ΦEXP2 circuit outperforms in terms of speed in most cases all the circuits that are based on ripple carry adders, carry look-ahead adders and outperforms Beauregard's, Fowler-Devitt-Hollenberg and Takahasi-Kunihiro QFT based circuits, while at the same time requires qubits of the same size order.

Algorithms D, E and F of Van Meter and Itoh (VI) try to improve the depth by applying various techniques such as better depth modulo calculation, indirection [129] but the main improvement is done by operating in parallel many modular multipliers at the cost of a respective increase of the qubits. As many tuning parameter are used in the VI algorithms, the expressions giving the qubits number and the depth are complicated. In Table 4.5 we show the asymptotic depth when the largest number of qubits ($2n^2$) can effectively be used by these algorithms [142] that is when we take advantage of the highest offered concurrency. In this case algorithms D and E can achieve a depth of $O(nlog_2n)$ which is asymptotically better than the proposed ΦEXP2 design but this improvement will require $O(n^2)$ space. Algorithm F has for the same number of qubits ($2n^2$) a depth of $100n^2log_2n$ which is asymptotically worst than the proposed design. If we relax the space requirements of the three VI algorithms D, E and F to be linear $O(n)$, then algorithms D and E offer an asymptotical depth $O((nlog_2n)(n/s+log_2s))$ where s is the number of concurrent multipliers used and algorithm F offers a depth of $O((nlog_2n)(n/s+log_2s))$. These depths are worst in order than the asymptotical depth of the ΦEXP2 architecture if s is set constant.

Table 4.5: Comparison of various modular exponentiation quantum circuits in terms of qubits requirement (width), speed (depth), number of gates used (quantum cost) and depth-width product. Second column succinctly describes the architecture and the basic block used (usually the kind of adder), third column shows the interactions requirement and the fourth column distinguishes between exact or approximate calculations performed. For the depth and gates estimations we have assumed that whenever Toffoli gates are used, they contribute five times the quantum cost and depth of two or single qubit gates.

Name	Arch./Basic Block	Interactions	Calculations	Width	Depth	Gates	Depth x Width
VBE [61]	Ripple Carry	Distant	Exact	$\sim 4n \ldots 7n$	$\sim 500n^3$	$O(n^3)$	$O(n^4)$
BCDP [62]	Ripple Carry	Distant	Exact	$\sim 4n \ldots 7n$	$\sim 280n^3$	$O(n^3)$	$O(n^4)$
CDKM [125]	Ripple Carry	Distant	Exact	$\sim 3n \ldots 6n$	$\sim 200n^3$	$O(n^3)$	$O(n^4)$
DKRS [126]	Carry Look-Ahead	Distant	Exact	$\sim 5n \ldots 8n$	$\sim 400n^2 \log n$	$O(n^3)$	$O(n^3 \log n)$
TK 1[123]	Ripple Carry	Distant	Exact	$3n+2$	$\sim 500n3$	$O(n^3 \log n)$	$O(n^4)$
TK 2[143]	QFT	Distant	Exact	$2n+2$	$O(n^3)$	$O(n^4)$	$O(n^4)$
VI-algoD [129]	Conditional Sum	Distant	Exact	$2n^2$	$\sim 45n(\log n)^2$	$O(n^3)$	$O(n^3 \log^2 n)$
VI-algoE [129]	DKRS	Distant	Exact	$2n^2$	$\sim 55n(\log n)^2$	$O(n^3)$	$O(n^3 \log^2 n)$
VI-algoF [129]	CDKM	1D-LNN	Exact	$2n^2$	$\sim 100n^2 \log n$	$O(n^3)$	$O(n^4 \log n)$
Beauregard [24]	QFT	Distant	Exact	$2n+3$	$\sim 100n^3$	$O(n^4)$	$O(n^4)$
Gosset [127]	Carry Save	Distant	Exact	$8n^2$	$O(n \log n)$	$O(n^3)$	$O(n^3 \log n)$
Zalka 1 [128]	Carry Select	Distant	Approx.	$5n$	$\sim 3000n^2$	$O(n^3)$	$O(n^3)$
Zalka 2 [128]	FFT Multiplier	Distant	Exact	$24n \ldots 96n$	$\sim 2^{19}n^{1.2}$	$2^{19}n^{1.2}$	$2^{21}n^{1.2}$
FDH [130]	QFT	1D-LNN	Exact	$\sim 2n$	$O(n^3)$	$O(n^4)$	$O(n^4)$
Kutin 1 [131]	QFT	1D-LNN	Approx.	$\sim 3n$	$O(n^2)$	$O(n^3)$	$O(n^3)$
Kutin 2 [131]	CDKM	1D-LNN	Approx.	$\sim 3n$	$O(n^2 \log n)$	$O(n^3)$	$O(n^3 \log n)$
CV [132]	Carry Look-Ahead	2D-LNN	Exact	$\sim 5n \ldots 8n$	$\sim 750n^2\sqrt{n}$	$O(n^3)$	$O(n^3\sqrt{n})$
PS [133]	Carry Save	2D-LNN	Exact	$O(n^4)$	$O((\log n)^2)$	$O(n^4)$	$O(n^4 \log n)$
ΦEXP1 [141]	QFT Add/MAC/Div	1D-LNN	Exact	$16n+2$	$\sim 1300n^2$	$O(n^3)$	$O(n^3)$
ΦEXP2 [141]	QFT Add/MAC/Div	1D-LNN	Exact	$9n+2$	$\sim 700n^2$	$O(n^3)$	$O(n^3)$

Gosset's carry save adder circuit has smaller depth than ΦEXP2 circuit but with a large penalty in space because the number of qubits bits it requires depends quadraticaly on the size of the number to be factored.

The same applies for the Pham and Svore (PS) two-dimensional architecture which has a $O(log_2 n)^2$ depth but requires a tremendous $O(n^4)$ space. The second two-dimensional architecture (CV) which requires space of about $4n$ has also asymptotically worst depth of $O(n^2\sqrt{n})$.

Zalka's FFT multiplier circuit performs better than ΦEXP2 but only for numbers to be factored with more than 10kbits in size due to its big depth constant. Also it uses much more qubits than ΦEXP2. Zalka's first circuit (using carry select adder) is comparable to ΦEXP2 in terms of both depth and qubits required, the ΦEXP2 being faster but using twice the qubits number of Zalka's circuit. At this point we have to mention that Zalka's first circuit makes only approximate calculations of the modular exponentiation function.

The other architecture which has comparable asymptotical depth to ΦEXP2 is Kutin's one-dimensional architecture based on QFT addition, being an approximate calculation circuit like Zalka's first circuit. The second one of Kutin's circuit is slightly worse in terms of depth and makes the same approximation as its first.

To conclude the complexity analysis, the ΦEXP2 circuit has the lowest asymptotical depth among the circuits that require a linear number of qubits smaller than $10n$ and are based on exact (as opposed to approximate) calculations.

The ΦEXP2 circuit natively uses almost exclusively two-qubit gates (the only exception are the CSWAP gates which use Toffoli gates). This is an advantage over most of the architectures of Table 4.5 (apart from Beauregard's circuit which can be also transformed to use almost exclusively two qubit gates) because physical implementations of quantum gates of three qubits is difficult [52]. Even recent proposals of Toffoli gate implementation in various technologies [134, 135, 136, 137] essentially resolve this problem by decomposition into two qubit gates.

Of particular interest is the depth-width product metric, which is related to the fault tolerance demands of the circuit, the smaller this product is the smaller the error correction demanding [144]. The proposed circuit has among the smallest depth-width products, of the order $O(n^3)$, among the circuits which perform exact calculations. The only circuits that have the same depth-width product are the ones of Zalka's and Kutin's which perform approximate calculations.

On the other hand, concerning the implementation, the proposed architecture seems to have two major weaknesses. The first comes from the fact that it uses extensively controlled rotation gates and there are known problems concerning the fault tolerance capability of such gates [129] and their implementation if small rotation angles are desired. The rotation gates used in the described arithmetic circuits may have very small angles like $2\pi/2^n$ and it is difficult to realize physical gates of such accuracy. Also, the rotation gates are not included in the family of gates that can be realized in a fault tolerant manner using known quantum error correcting codes, e.g. Steane codes. But even if they are not inherently fault tolerant capable they can become such, if a decomposition into a set of fault tolerant gates is applied to them exploiting the Solovay-Kitaev theorem [56].

The problem of decomposing an arbitrary quantum single qubit gate into a pre-determined set of fault tolerant gates is important because for such a decomposition a cost have to be paid related to the sequence length of gates to realize the decomposition and consequently it is related to a depth increment of the total circuit. The Solovay-Kitaev theorem states that

given a small constant ε, an arbitrary gate U can be approximated by a finite sequence of gates equivalent to a gate S up to an approximation error ε (that is $d(U,S) < \varepsilon$, where d is a distance function), the length of this sequence being $O(log^c(1/\varepsilon))$. The constant c is somewhere between 1 and 2. As the smallest rotation angle in the rotation gates of the proposed architecture is of the order $\pi/2^n$, the required approximation error is of the same order (otherwise we would use the identity gate instead [145]) and the depth cost is of the order n^c.

Much research has been done in the area of this problem. Some of this work is oriented to improve the constant factor c, that is to improve the cost and depth of the decomposed circuit. In [145, 146] a decomposition of a controlled rotation gate into two single qubit rotation gates and a CNOT gate is applied, and then the single qubit rotation gate is approximated, up to an error constant ε, by a sequence of H and T ($\pi/8$) gates. It is shown that this approximation sequence has a length linear in $O(log(1/\varepsilon))$, that is the constant c is equal to 1, but the major drawback of this method is the synthesis time required, which is exponential in $1/\varepsilon$.

Recently, a very intense research activity has been observed in the area of efficiently synthesizing arbitrary gates using fault tolerant primitive gates. Some work is oriented in trading-off circuit complexity with synthesizing time complexity, other work is diverted between building arbitrary gates with and without ancillae, and some other work is directed to using not standard primitive gate set. A non exhaustive list of such work includes [60, 59, 147, 148, 60, 149]. In summary, these results show that an arbitrary gate can be approximated with linear length with respect to $\log(1/\varepsilon)$ and with linear synthesis time with respect to $\log(1/\varepsilon)$. The constant factor in the complexity of the approximation circuit is about 10 for the most promising proposals. As the desired error ε is of the order of the smallest angle $2\pi/2^n$, the depth overhead factor is of the order $O(n)$. Thus, the depth of the overall modular exponentiation circuit would become $O(n^3)$.

Nevertheless we could achieve a better depth performance if we relax the approximation error requirement. In section 5.1 we show that by permitting ε in the order of $O(1/2^{t\log n}) = O(1/n^t)$ for a suitable constant t, we could still achieve adequate probability of success in the quantum period finding. This choice of ε leads to an overall depth of $O(n^2 \log n)$ for the quantum modular exponentiation.

The second weakness of the proposed architecture seems to be at first sight that it does not account for the possible constraints on the communication distance between the qubits that may imposed by the underlying physical implementation. For example in [26] it is shown that the physical mapping of any quantum adder of $\Omega(log_2 n)$ depth to a k-dimensional nearest neighbors architecture limits its theoretical depth to $\Omega(\sqrt[k]{n})$. Such depth limitations are due to the additional SWAP gates needed to convert from the abstract concurrent architecture to the nearest neighbor architecture.

This is not the case for the proposed architectures ΦEXP1 and ΦEXP2. Section 5.2 gives an adaptation procedure to an one dimensional linear array physical machine with parallel gates execution capability (1D-NTC) which leaves the overall depth to $O(n^2)$ before the fault tolerance transformations discussed previously and detailed in section 5.2.

Other models of quantum computation more suitable for long distance communication between gates are the measurement based quantum computation (MBQC) [150] where the interaction between distant qubits can be accomplished in constant time. In [151] the DKRS carry look-ahead adder is redesigned in the MBQC context and it is shown that its logarithmic depth can be maintained but with a substantial overhead in space requirements. Surface code quantum computing is another promising scalable architecture which

also permits long distance interactions [152, 153, 154].

In general, the estimation of area and speed of a quantum algorithm is a cumbersome task when considering the real physical implementation of the algorithm, including the error-correcting codes which achieve the required fault-tolerance. In this view, the comparison between the various designs exposed in Table 4.5 is only indicative for their performance.

It is important to note that the ΦEXP2 circuit can be made about three times faster if the *approximate* QFT method of performing the additions [23, 155, 156] is utilized. This happens because significant part of the constant divider GMΦDIV consists of QFT adders and the constant divider has the dominant depth contribution. This gate pruning may be incorporated as part of the approximation procedures described in section 5.1. Unfortunately, the approximate QFT method can not be applied to the the ΦMAC blocks because the angles in every rotation gate in this block is a sum of angles in a range from the bigger to smaller angle, depending on the numbers to be multiplied (see Eq. (4.17)).

In conclusion, even if the modular exponentiation architectures ΦEXP1 and ΦEXP2 seem to have the two disadvantages mentioned above in relation to the fault tolerance and their usage in a local interaction physical implementation, they can address these disadvantages as proposed in Chapter 5 and still achieve a low depth of $O(n^2 \log n)$. In contrast, some seemingly $O(n^2 \log n)$ depth architectures that are based on fast adders of depth $O(\log n)$ loose their depth advantage when applied on 1D-NTC or 2D-NTC architectures. We think, as these results suggest, that the proposed fast quantum modular exponentiation could be a candidate architecture in the future if someone gives priority to depth, i.e. the performance of the circuit.

4.7 Divider Improvement and Extension

After the publication of [141] it was realized that a more efficient design of the proposed quantum divider by constant can be achieved. This improved divider is shown in Figure 4.24 and requires $6n+1$ qubits, instead of the $7n+1$ of the original design exposed in Figures 4.16-4.17. Depth and quantum cost are similar to the original divider. The incorporation of this improved divider into the proposed architecture for the modular exponentiation leads to space requirements $8n+1$ qubits for Shor's algorithm.

A slight modification of this circuit, with the addition of one more qubit, leads to a controlled version of a divider. Figure 4.25 is the modified circuit of the controlled divider GMΦDIV, where the control qubit is added on the top of the figure and denoted with $|c\rangle$. The operation of this circuit is to give the quotient and the remainder if $|c\rangle = |1\rangle$, otherwise it gives the dividend $|z\rangle$ in their place. The control qubit affects the two top buses (the one which normally would deliver the quotient and the bus of the dividend high qubits) as follows: The CNOT gates copying the $|q_1\rangle$ state on the top bus are promoted to Toffoli gates controlled additionally by the added control qubit of the divider. Also, the two X gates operating on the most significant qubit of the dividend are promoted to CNOT gates and they are additionally controlled by the same control qubit of the divider. It is easy to ascertain that the case $|c\rangle = |1\rangle$ leads to identical operation as in the case of an uncontrolled divider. On the other side, when $|c\rangle = |0\rangle$, the top bus normally carrying the quotient at the end of the computation, remains in its zero initial state. This is because the Toffoli gates are disabled and the same applies for the controlled adder of constant 1, as the most significant qubit of the dividend is always zero, both in the case of GMFDIV1 and GMΦDIV2 modes of operation. The ancilla buses remain in their initial states of zero due to the symmetry of the computation - uncomputation effect. The dividend bus remains in the iniitial state due to this symmetry and the fact that the last ΦMAC unit has no effect on it, as the top bus remains in zero state.

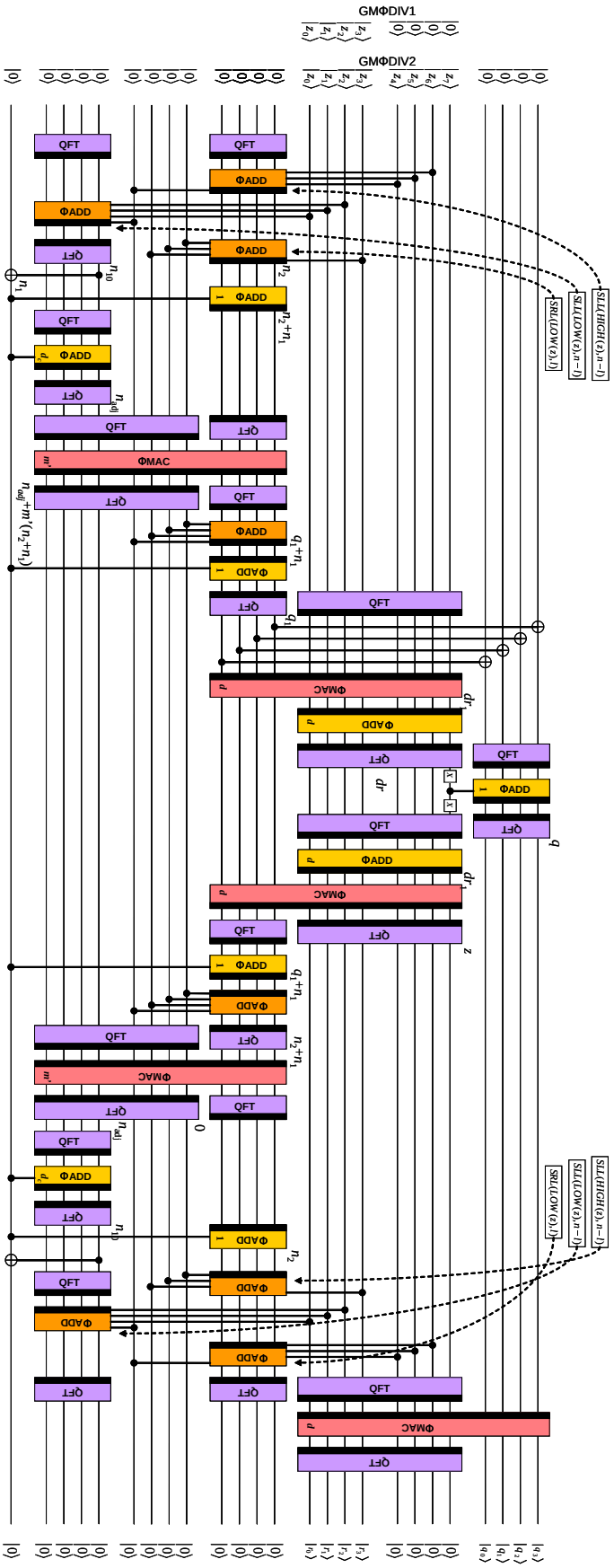


Figure 4.24: Improved GMΦDIV circuit of space complexity $6n+1$ qubits for the case of 8 or 4 qubits dividend and constant divisor $d=5$.

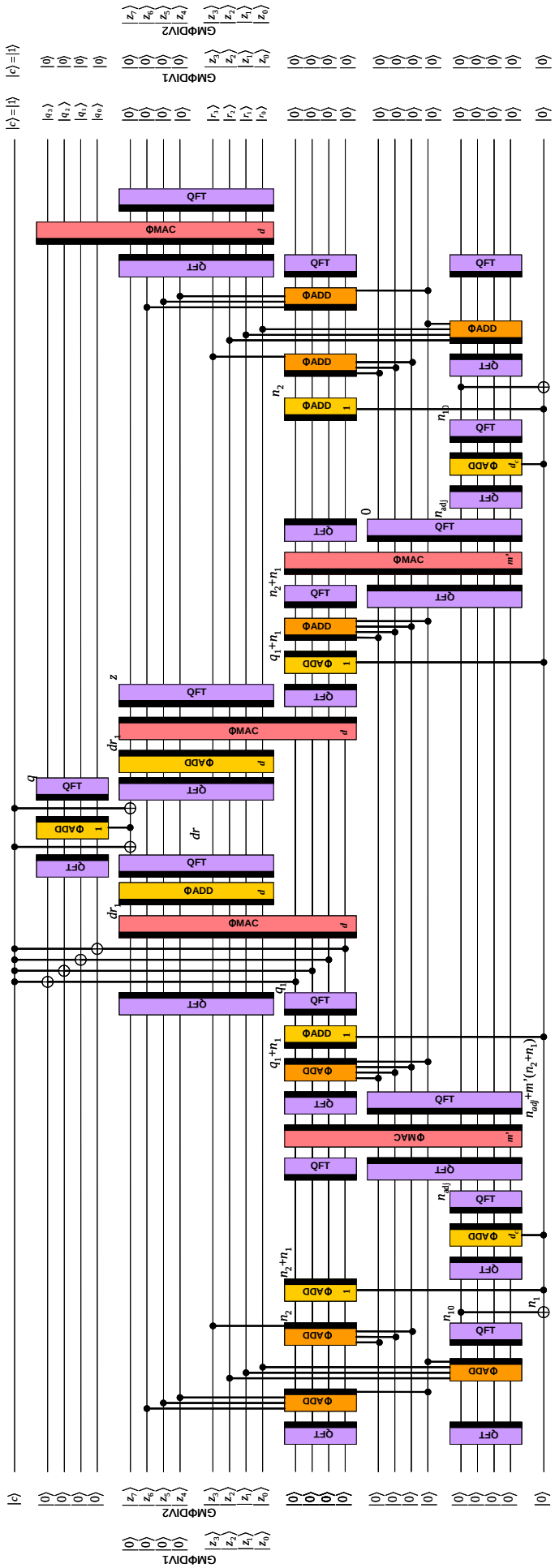


Figure 4.25: Controlled GMΦDIV circuit of space complexity $6n+2$ qubits for the case of 8 or 4 qubits dividend and constant divisor $d = 5$. The controlling qubit is $|c\rangle$.

5. IMPLEMENTATION ISSUES OF QFT BASED ARITHMETIC CIRCUITS

In this Chapter we address the two major weaknesses of the proposed arithmetic quantum circuits which are based on the QFT representation of an integer. First, the fact that the requirement to implement fault tolerant rotation gates of high accuracy angle of order $(1/2^n)$ [1], leads to a depth increase from $O(n)$ to $O(n^2)$ for components like ΦMAC and GMΦDIV, and consequently to an overall depth increase from $O(n^2)$ to $O(n^3)$ for the whole quantum modular exponentiation. In section 5.1 we show that we can relax the angle accuracy requirement, using angles of accuracy $O(1/n)$ instead of $(1/2^n)$ and accomplish depths of $O(n \log n)$ for the ΦMAC and GMΦDIV, leading to a depth of $O(n^2 \log n)$ for the whole modular exponentiation. This accuracy relaxing does not degrade significantly the operation of the quantum period finding algorithm in terms of success probability, as we show.

The second implementation issue is the interaction topology between the qubits. The basic arithmetic blocks proposed in the previous Chapter imply distant interactions between qubits (e.g. up to a distance of $3n$ for the ΦMAC). Yet, the physical implementations in various technologies are usually one-dimensional or two-dimensional arrays of qubits restricted to nearest neighbors interactions (1D-LNN or 2D-LNN). In section 5.2 we show that the various blocks like ΦMAC and GMΦDIV can be mapped to such local interactions topologies through the usage of SWAP gates and still achieve depths of the same order, that is overhead of constant factor.

5.1 Angle Quantization of Rotation Gates

The problem with the small angles rotation gates usage has been investigated in the context of QFT application [156, 155, 145, 146], because QFT is an integral part of Shor's algorithm. Those works study the effect on the performance of the period finding algorithm, in terms of success probability to find the period, that has the usage of the approximated QFT. In an approximated QFT with parameter b, all the rotation gates with angle smaller than $2\pi/2^{b+1}$ are discarded. This elimination of gates is also called *QFT banding* due to the shape of the remaining gates in the resulting approximated QFT circuit. An approximated QFT with banding $b = n - 1$ of the n qubits QFT corresponds to the regular QFT (all the gates are present), while at the other edge a banding $b = 0$ corresponds to a QFT circuit in which only the Hadamard gates are left. The above works concluded more or less that using a $b = O(\log n)$ is adequate to extract the period with acceptable probability of success. These results were also refined and verified with simulations in more recent works [157, 158, 159].

Recently, an interest appeared to extend the method of pruning rotation gates to other circuits as well, and studied the effect on their performance in terms of probability of success. Beauregard's design [24] for period factoring algorithm is such an example. It implements the modular exponentiation part by using Draper's QFT adder of Figure 4.9 as a basic building block in order to successively build the modular adder and the modular multiplier of the usual hierarchy described in Section 4.1. The works in [158, 160] study the effect of simultaneously eliminating small angles rotation gates of both QFT and Draper's adder used in the Beauregard's design.

In [158] a full quantum gate level simulation of Bauregard's design was performed to factor integers up to $N = 57$. The simulations covered various random a parameters of the modular exponentiation $a^x \bmod N$. The results showed that if all the rotation gates with angles

[1] See for example Eq. (4.18), (4.19) and (4.20).

smaller than $\pi/2^b$ are discarded, using $b = 4$ and $b = 5$ for the QFT part and the modular exponentiation part, respectively, the algorithm still succeeds to factorize with degradation of about 70% relative to the algorithm operation if no gate pruning would happen. These results were extended in [160] where both banding in the QFT and Draper's adder is applied and at the same time the remaining gates are allowed to have defects modeled as fluctuations about their real angles values.

All the above results suggest a significant robustness of circuits that have a QFT like structure to both banding and gate defects. The proposed architecture of Chapter 4 is based almost exclusively on building blocks that have such a QFT like structure. In the following sections it is shown that this architecture is robust to another kind of approximation, which we call *angle quantization of rotation gates*. In this procedure, the angle value of each rotation gate is changed to the nearest value of the set $\{\frac{2\pi}{2^b}j : j = 0 \dots 2^b - 1\}$. This is equivalent to permit an angle error of at most $\pi/2^b$ in each gate. It is shown that if parameter b of this quantization is chosen logarithmic in n, where n is the length of the factorisable integer, then adequate probability of success is maintained.

The consequence of this result is that the gate and depth overhead of the circuit supervened due to the required conversion of the rotation gates to sequence of gates that admit fault tolerant implementation, is logarithmic, instead of linear as it would seem at first sight. Thus, including the conversion to a fault tolerant set of gates, the modular exponentiation circuit analyzed in Chapter 4 has a depth $O(n^2 \log n)$ instead of $O(n^3)$ that it would have if not using the proposed angle quantization method.

5.1.1 Definitions and basic properties

The induced 2-norm of an operator U acting on a d-dimensional Hilbert space $\mathcal{H}^d$ is defined by

$$\begin{aligned} \|U\|_2 &\doteq \sup_{|\psi\rangle} \frac{\|U|\psi\rangle\|_2}{\||\psi\rangle\|_2} \\ &= \max_{|\psi\rangle} \|U|\psi\rangle\|_2 \\ &= \sqrt{\lambda_{\max}(U^\dagger U)} \end{aligned} \tag{5.1}$$

where $\|\cdot\|_2$ is the 2-norm of a state vector in $\mathcal{H}^d$ defined by

$$\||\psi\rangle\|_2 = \sqrt{\langle\psi|\psi\rangle} \tag{5.2}$$

and $\lambda_{\max}(\cdot)$ denotes the largest eigenvalue of an operator.

The above operator norm will be used to quantify the distance between a target quantum gate or circuit U when it is approximated by another gate or circuit V. The approximation error between the two circuits U and V is defined by

$$E(U, V) \doteq \|U - V\|_2 \tag{5.3}$$

This definition is justified by the fact that the distance between the probability distributions $\mathrm{Prob}_U[k]$ and $\mathrm{Prob}_V[k]$ ($k = 0 \dots 2^d - 1$) generated at the outputs of the two circuits for the same projective measurement is at most the approximation error $E(U, V)$, where the distance between probability distributions is the *Total Variation Distance* [161]

$$\mathrm{TVD}(\mathrm{Prob}_U, \mathrm{Prob}_V) \doteq \frac{1}{2}\sum_{k=0}^{2^d-1} \left|\mathrm{Prob}_U[k] - \mathrm{Prob}_V[k]\right| \tag{5.4}$$

Thus, we have

$$\mathrm{TVD}(\mathrm{Prob}_U, \mathrm{Prob}_V) \leq E(U, V) \tag{5.5}$$

It can be proven [161, 41] that if a sequence of d-qubits gates $U_1, \ldots, U_k$ is approximated by another sequence of gates $V_1, \ldots, V_k$ with approximation errors between each pair $E(U_j, V_j)$ for $j = 1 \ldots k$, then the total error between each product of the sequence is at most the sum of errors of each pair

$$E(U_k U_{k-1} \cdots U_1, V_k V_{k-1} \cdots V_1) \leq \sum_{j=1}^{k} E(U_j, V_j) \tag{5.6}$$

A large quantum circuit of d qubits can be thought as a sequence of d qubits gates $U_1 \ldots U_k$ as described above, but usually the elementary gates used in the circuits act on a single or two qubits alone. A first step to handle this case is to consider that an elementary gate U of n qubits is embedded to a larger circuit of $d = m + n$ qubits, in which it acts on the lower n qubits and the identity gate acts on upper m qubits as shown in left of Figure 5.1. Let $A = I^{\otimes m} \otimes U$ be the embedding of U and V is the approximation of U. Then the embedding $B = I^{\otimes m} \otimes V$ of V has the same distance to A as has V to U,

$$\left\|I^{\otimes m} \otimes U - I^{\otimes m} \otimes V\right\|_2 = \|U - V\|_2 \tag{5.7}$$

Proof. By Eq. (5.1), the value of $\|I^{\otimes m} \otimes U - I^{\otimes m} \otimes V\|_2$ is the largest eigenvalue of $(A - B)^\dagger(A - B)$. The calculation of $(A - B)^\dagger(A - B)$ follows (for simplicity $I^{\otimes m}$ will be denoted by I)

$$\begin{aligned}
(A-B)^\dagger(A-B) &= (I\otimes U - I\otimes V)^\dagger(I\otimes U - I\otimes V) \\
&= (I\otimes U^\dagger)(I\otimes U) - (I\otimes V^\dagger)(I\otimes U) - (I\otimes U^\dagger)(I\otimes V) + (I\otimes V^\dagger)(I\otimes V) \\
&= I\otimes U^\dagger U - I\otimes V^\dagger U - I\otimes U^\dagger V + I\otimes V^\dagger V \\
&= I\otimes \left(U^\dagger U - V^\dagger U - U^\dagger V + V^\dagger V\right) \\
&= I\otimes (U-V)^\dagger(U-V)
\end{aligned} \tag{5.8}$$

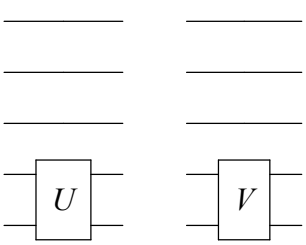


Figure 5.1: Left circuit $A = I \otimes U$ approximated by the right circuit $B = I \otimes V$. The distance $\|A - B\|_2$ between the two five-qubits circuits is the same as the distance between the two gates alone $\|U - V\|_2$.

The eigenvalues of $I \otimes (U-V)^\dagger(U-V)$ are the same with the eigenvalues of $(U-V)^\dagger(U-V)$ as the eigenvalues of I are all 1. Consequently, $\lambda_{\max}((I \otimes (U-V)^\dagger(U-V)) = \lambda_{\max}((U-V)^\dagger(U-V))$ and thus $\|I^{\otimes m} \otimes U - I^{\otimes m} \otimes V\|_2 = \|U-V\|_2$ □

It is easy to generalize the above result by allowing the n qubits of U and V gates to correspond to any qubits of the largest $n+m$ qubits circuit, instead to be the least significant qubits as Figure 5.1 implies. This is accomplished by inserting suitable SWAP gates at the beginning and at the end of the original circuit A and the approximating circuit B to derive the corresponding $A' = P^{-1} \cdot A \cdot P$ and $B' = P^{-1} \cdot B \cdot P$ circuits. The P operator and its inverse P^{-1} are permutations that correspond to the effect of SWAP gates inserted at the beginning and at the end of circuit. Figure 5.2 is an example of this equivalence where c-R is used as an example of a two-qubits gate U.

It is easy to see that

$$\|A' - B'\|_2 = \left\|P^{-1} \cdot A \cdot P - P^{-1} \cdot B \cdot P\right\|_2 = \|A - B\|_2 \tag{5.9}$$

because P and P^{-1} are unitary.

In the same line of reasoning, if $C_1 \ldots C_p, C_q \ldots C_r$ are unitary operators and U is approximated by V, then

$$\|C_1 \cdots C_p U C_q \cdots C_r - C_1 \cdots C_p V C_q \cdots C_r\|_2 = \|U - V\|_2 \tag{5.10}$$

The U and V in the above equation may be of the form of A' and B' of Eq. (5.9), respectively.

Combining the previous results we can state the following

Proposition 5.1. *Let Q a quantum circuit and $\hat{Q}$ an approximating quantum circuit which is derived from Q by replacing some of its gates $U_1, U_2, \ldots U_k$ with the respective gates $V_1, V_2, \ldots V_k$. The 2-norm distance between Q and $\hat{Q}$ is upper bounded as*

$$\left\|Q - \hat{Q}\right\|_2 \leq \sum_{j=1}^{k} \|U_j, V_j\|_2 \tag{5.11}$$

Proof. The proof is straightforward taking into account Eq. (5.6),(5.9), (5.10) and the fact that every quantum circuit is represented by a product of tensor products of unitary operators. □

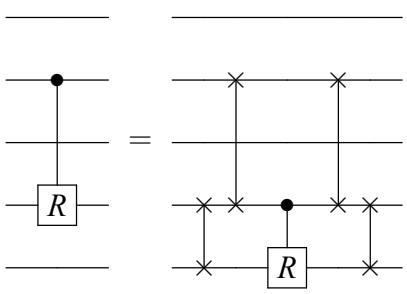


Figure 5.2: Equivalence of a five qubits circuit A' containing a two-qubit gate c-R acting in some of the middle qubits (left) to another five qubits circuit A in which the same gate acts on the two lowest qubits. Swap gates are used to interchange the order of the qubits.

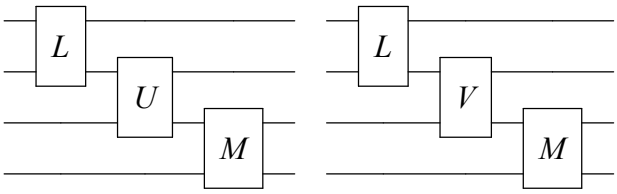


Figure 5.3: The left circuit is approximated by the right circuit by replacing gate U with V.

An example of the previous results is shown in Figure 5.3. The left circuit is $A = (I^{\otimes 2} \otimes M)(P^{-1}UP)(L \otimes I^{\otimes 2})$ and it is approximated by the right circuit $A = (I^{\otimes 2} \otimes M)(P^{-1}VP)(L \otimes I^{\otimes 2})$ where P is a suitable permutation matrix which models the fact that U and V gates act on the two middle qubits. The distance of these two circuits is $\|A - B\|_2 = \|U - V\|_2$.

5.1.2 Approximation of the multiplier/accumulator ΦMAC.

The ΦMAC presented in Section 4.2 is a basic bulding block for both the modular multiplier ΦMUL_MOD and the divider by constant GMΦDIV used in the modular exponentiation. It consists of $4n^2$ controlled rotation gates given by Eq. (4.18) and (4.19), $2n$ controlled rotation gates given by Eq. (4.20) and $2n$ CNOT gates. These equations show that each one of these rotation gates has an angle φ which is a sum of angles between $2\pi/2^m$, $m = 1 \ldots 2n$ and the sum depends on the multiplication constant a, so that each angle φ is an integer multiple of $2\pi/2^{2n}$. Thus, there isn't any set of these gates guaranteed to have small angles so as to apply the banding procedure as it can be applied to the approximated QFT case or the QFT adder of two quantum integers.

Instead, we can uniformly "quantize" the φ angles with quantization step of $2\pi/2^b$ so as they are all multiples of $2\pi/2^b$ instead of $2\pi/2^{2n}$, for a suitable constant $b \ll n$. Namely, a controlled rotation gate $c\text{-}R_z(\varphi)$ is approximated by the gate $c\text{-}R_z(\hat{\varphi})$ using the relation

$$\hat{\varphi} = \text{round}\left(\varphi \frac{2^b}{2\pi}\right) \frac{2\pi}{2^b} \tag{5.12}$$

It is clear from the above relation that the distance between the desired angle φ and its quantized value $\hat{\varphi}$ is half the quantization step,

$$|\varphi - \hat{\varphi}| \leq = \frac{\pi}{2^b} \tag{5.13}$$

The distance between a desired controlled rotation gate $c\text{-}R_z(\varphi)$ and its approximation $c\text{-}R_z(\hat{\varphi})$ is

$$\begin{aligned} \|c\text{-}R_z(\varphi) - c\text{-}R_z(\hat{\varphi})\|_2 &= \max_{|\psi\rangle} \left\| \text{diag}\left(0,0,0,e^{i\varphi} - e^{i\hat{\varphi}}\right) |\psi\rangle \right\|_2 \\ &= \left| e^{i\varphi} - e^{i\hat{\varphi}} \right| \\ &\leq |\varphi - \hat{\varphi}| \\ &\leq \frac{\pi}{2^b} \end{aligned} \tag{5.14}$$

We choose the quantization parameter b to scale logarithmically with respect to n, e.g. $b = c \log n$. By applying Proposition 5.1 we can estimate the distance of the original ΦMAC and the approximated ΦMAC with the following upper bound

$$\begin{aligned}\left\|\Phi MAC - \widehat{\Phi MAC}\right\|_2 &\leq \sum_{k=1}^{4n^2+2n} \|U_k - V_k\|_2 \\ &\leq \sum_{k=1}^{4n(n+1)} \frac{\pi}{2^{c\log n}} \\ &\leq \frac{5\pi}{n^{c-2}}\end{aligned} \tag{5.15}$$

In the above formula, U_k are all rotation gates of the original ΦMAC embedded in the whole Hilbert space $\mathcal{H}^{3n}$ of ΦMAC, that is they are modeled with

$$U_k = P_k^{-1}\left(I \otimes c\text{-}R_z(\varphi_k)\right) P_k \tag{5.16}$$

The V_k gates are the respective ones for the approximation ΦMAC circuit and are given similarly as

$$U_k = P_k^{-1}\left(I \otimes c\text{-}R_z(\hat{\varphi}_k)\right) P_k \tag{5.17}$$

5.1.3 Approximation of the Fourier adders and QFT.

Similar analysis can be applied to the two kind of adders used in the quantum divider and the QFT. In all the cases, the angles of all the rotation gates used in each block are approximated with quantization parameter $b = c\log n$. The distance between each original block and its approximation is related to the quantum cost of each block as stated in Proposition 5.1.

The controlled constant adder CΦADD of n qubits uses n rotation gates and consequently its distance to the approximating circuit is

$$\left\|C\Phi ADDC - C\widehat{\Phi ADDC}\right\|_2 \leq \frac{\pi}{n^{c-1}} \tag{5.18}$$

The distance for the general adder ΦADD of two quantum integers of n qubits is

$$\left\|\Phi ADD - \widehat{\Phi ADD}\right\|_2 \leq \frac{\pi}{n^{c-2}} \tag{5.19}$$

because it conists of $\frac{n(n+1)}{2}$ rotation gates.

Similarly, the distance for any QFT (and its inverse) of n qubits is

$$\left\|QFT - \widehat{QFT}\right\|_2 \leq \frac{\pi}{n^{c-2}} \tag{5.20}$$

Note that in the last two cases (adder of two quantum integers and QFT) the angle quantization with parameter b corresponds to banding, because the angles of all the rotation gates are of the form $2\pi/2^m$ for $m = 1\ldots n$, instead of sum of such angles as it was the case of ΦMAC.

5.1.4 Approximation of the whole modular exponentiation.

The GMΦDIV divider consists mainly of a constant number of blocks like QFT, ΦADD, CΦADD, ΦMAC that can approximated as described in the previous sections, and some constant number of other gates that are not approximated.

Taking into account Eq. 5.15,5.18,5.19 and 5.20 which give the approximation error if the quantization of angles with $b = c \log n$ is adopted, we derive the approximation error of the whole divider as

$$\left\| GM\Phi DIV - \widehat{GM\Phi DIV} \right\|_2 \leq C_d n^{2-c} \tag{5.21}$$

Constant C_d depends on the number of the various blocks used in the divider, but it is estimated to be somewhere above 89, because the dominant contribution to the error comes from the ΦMAC, ΦADD and QFT blocks (order of n^{2-c}) and a GMΦDIV block uses 5 ΦMAC units, 8 ΦADD units, 16 QFT units of n qubits and 10 QFT units of $2n$ qubits.

Error of the same order is found for the controlled modular multiplier CΦMUL_MOD as this block consists of two dividers and two ΦMAC

$$\left\| C\Phi MUL\_MOD - \widehat{C\Phi MUL\_MOD} \right\|_2 \leq C_m n^{2-c} \tag{5.22}$$

Similar reasoning, leads to an estimation of the C_m near to $2 \times C_d + 2 \cdot 5 \approx 198$. Thus, the total approximation distance of the whole modular exponentiation circuit ΦEXP is

$$\left\| \Phi EXP - \widehat{\Phi EXP} \right\|_2 \leq C_e n^{3-c} \tag{5.23}$$

because it uses $2n$ modular multipliers, where C_e is above $2 \times C_m = 396$.

The whole Shor's algorithm circuit includes a QFT at the end which can be approximated too, with error $O(n^{2-c})$. Thus, taking into account Eq. (5.23) which gives the dominant distance, the Total Variation Distance for the approximated Shor's circuit proposed is given by

$$\mathrm{TVD}(\mathrm{Prob}_{\mathsf{Shor}}, \mathrm{Prob}_{\widehat{\mathsf{Shor}}}) \leq C_s n^{3-c} \tag{5.24}$$

for C_s above 396. The above upper bound for this distance can be arbitrary low for large n. It is a measure of quality between the probability of success for the quantum period finding using the original circuit and the approximated one. (The probability of success is about $4/\pi^2$ as shown in Eq. (A.14)). Let $\mathcal{K}_g$ be the set of the "good" measurements (the peaks appeared after the final QFT), $\mathrm{Prob}_{\mathrm{succ}}$ the probability of success when using the original circuit and $\mathrm{Prob}_{\widehat{\mathrm{succ}}}$ the probability of success when using the approximating circuit. Then,

$$\begin{aligned}
\mathrm{Prob}_{\mathrm{succ}} - \mathrm{Prob}_{\widehat{\mathrm{succ}}} &= \sum_{k \in \mathcal{K}_g} \mathrm{Prob}_{\mathsf{Shor}}[k] - \mathrm{Prob}_{\widehat{\mathsf{Shor}}}[k] \\
&\leq \sum_{k \in \mathcal{K}_g} \left| \mathrm{Prob}_{\mathsf{Shor}}[k] - \mathrm{Prob}_{\widehat{\mathsf{Shor}}}[k] \right| \\
&\leq \sum_{k} \left| \mathrm{Prob}_{\mathsf{Shor}}[k] - \mathrm{Prob}_{\widehat{\mathsf{Shor}}}[k] \right| \\
&= 2 \cdot \mathrm{TVD}(\mathrm{Prob}_{\mathsf{Shor}}, \mathrm{Prob}_{\widehat{\mathsf{Shor}}}) \\
&\leq 2 C_s n^{3-c}
\end{aligned} \tag{5.25}$$

A choice of $c > 3$ for any n leads to the conclusion that if the angle quantization step $2\pi/2^b$ is smaller than $2\pi/n^3$, then the probability of success of Shor's algorithm can be as close we want (depending on the choice of constant c) to the original one if no angle quantization is applied. In more detail, let the desired difference between the probabilities of success be smaller than δP. This can be accomplished if $2C_s n^{3-c} < \delta P$. A little algebra leads to the requirement that

$$n^{3-c} < \frac{\delta P}{2C_s} \implies c > 3 - \frac{\log \delta P - \log 2C_s}{\log n} \tag{5.26}$$

Then, the quantization parameter $b = c \log n$ must be at least $3 \log n + \log 2C_s - \log \delta P$. E.g. for $\delta P = 0.1$ (no more than 10% degradation in probability of success) a choice of $b > 3 \log n + 9.7 + 3.3 = 3 \log n + 13.0$ is adequate, while for $\delta P = 0.01$ we derive $b > 3 \log n + 16.3$.

The proposed angle quantization does not need to be performed explicitly. Instead, it can be integrated to the Solovay-Kitaev like synthesis algorithms incorporated to the physical realization design flow which transforms the rotation gates to a sequence of gates admitting fault tolerant realization. In this case, the error parameter ε of the approximation synthesis algorithm is set to about the half the quantization step, $\varepsilon \approx \pi/n^c$. An efficient synthesis algorithm, like these of [59, 60, 149], approximates the rotation gates with a sequence of length $O(\log 1/\varepsilon) = O(\log n)$. In other words, a logarithmic overhead of depth is feasible (instead of linear) related to the length of integer to be factored n, if fault tolerance aspects are taken into account.

5.2 Communications Localization

The various modules used in the construction of the modular exponentiation based on Fourier arithmetic can be converted to modules that use local communications only; they can use local interactions between neighboring qubits. In particular, the proposed transformation is adapted for an one dimensional linear array of qubits, in which each qubit interacts with its two neighboring qubits only, while concurrent execution of gates applied on different qubits is allowed (1D-NTC architecture). SWAP gates are used extensively to circulate the different qubits around the array. The depth overhead in the basic modules with local communications compared to the depth of the original modules with global communications is linear in the number of the qubits, that is a constant multiplicative factor. For example, the QFT, ΦADD, ΦMAC and GMΦDIV modules retain their linear depth, although with a larger multiplicative constant. Thus, the overall depth of the modular exponentiation ΦEXP1 or ΦEXP2 circuits remains quadratic.

Some basic blocks are introduced to describe the localization of the ΦMAC. The "interleave" block shuffles $2n$ qubits initially in the order $a_{n-1}, \ldots, a_0, b_{n-1}, \ldots, b_0$ so that their

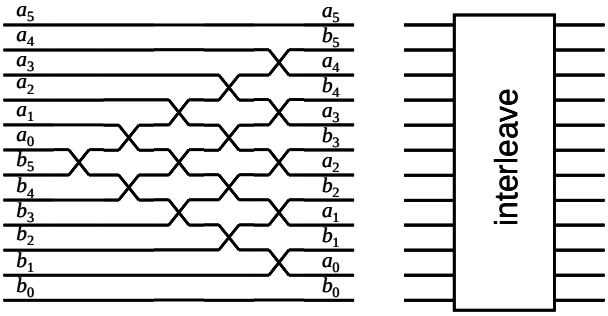


Figure 5.4: Qubits interleaving using local interactions.

order becomes $a_{n-1}, b_{n-1}, an-2, b_{n-2}, \ldots, a_0, b_0$. The example shown in Figure 5.4 is for the case $n = 6$. The generalization to other values of n is straightforward. The SWAP gates from now on will be denoted by crossing qubit wires (X symbol) to better keep track the exchanging of qubits in the array. The inverse block is the same circuit reflected vertically. The depth of the "interleave" is $n - 1$ steps.

Another useful block is the "exchange" block which exchanges the first n qubits of a group of $2n$ qubits with the last n qubits. Figure 5.5 shows the "exchange" block for the case $n = 6$. Observe that the "exchange" block consists of an "interleave" and an inverse "interleave" block which have n SWAP gates between them. The depth of the "exchange" block is $2n - 1$.

The shifted control block "sc" can be used when n different two-qubit gates must be successively applied to pairs of a common qubit c and n different qubits $a_{n-1} \ldots, a_0$ in order. The qubits $a_{n-1} \ldots, a_0$ must successively appoach c before the respective two qubit interaction. Figure 5.6 shows how this can be accomplished in $3n - 2$ steps. The two-qubit gates are denoted with a vertical line with two dots between two neighboring qubits. The initial order of the qubits $c, a_{n-1}, \ldots, a_0$ is settled down at the end. At each step is shown the qubit that has become neighbor of c. The symbol of the shifted control shown contains a parameter $\vec{A}$ which is a vector the matrices determining the n two-qubit gates, e.g. $\vec{A} = (A_1, \ldots A_n)$.

When the two-qubits gates $A_1, \ldots A_n$ mutually commute, their execution order is irrelevant so that a smaller $2n$ depth circuit like the "scc" of Figure 5.7 can be constructed. Such a case occurs when the gates are controlled rotation gates or when the gates are CNOT gates.

Equipped with the above localized modules we can proceed to transform the Fourier based multiplier/accumulator ΦMAC of Section 4.2 so as it operates using local communications only. Figure 4.15 shows that in general, the ΦMAC operates on $3n+1$ qubits in five stages.

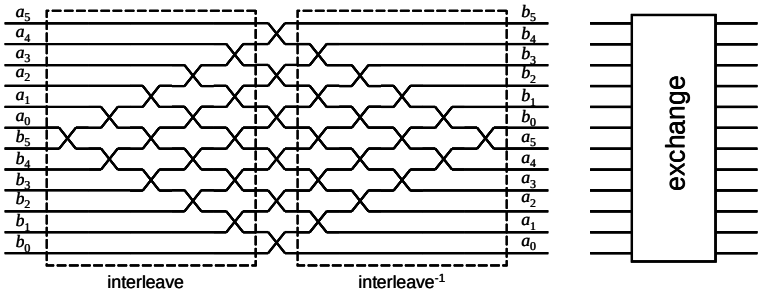


Figure 5.5: Exchange of two quantum registers using local interactions.

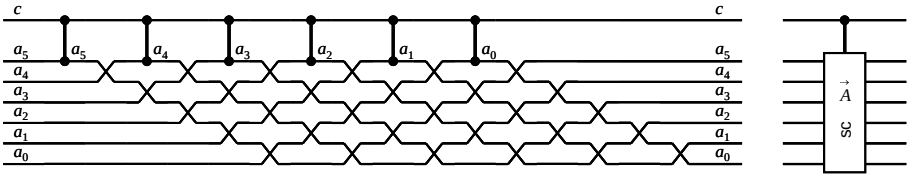


Figure 5.6: Shifted control circuit "sc". Two-qubits gates (denoted with vertical lines with dots at their ends) are applied consecutively between qubit c and a_j

.

To simplify the notation, we rename the n qubits carrying the multiplicand $|x\rangle$ in Figure 4.15 as a_2, a_1, a_0 and the qubits of the accumulator as $b_5, \ldots, b_0$. With this notation in mind, the five stages of the ΦMAC operations are as follows:

1. Rotation gates applying $V_l^{(j)}$ to the target qubit b_l and controlled by qubit a_j for $l = 0, \ldots, 2n-1$ and $j = 0, \ldots, n-1$.
2. Sequence of n CNOT gates having a common control qubit c which target the different n qubits $a_{n-1}, \ldots, a_0$.
3. Rotation gates applying $V_l^{(j)\dagger}$ to the target qubit b_l and controlled by qubit a_j for $l = 0, \ldots, 2n-1$ and $j = 0, \ldots, n-1$.
4. Sequence of n CNOT gates having a common control qubit c which target the different n qubits $a_{n-1}, \ldots, a_0$.
5. Sequence of rotation gates W_l for $l = 0, \ldots, 2n-1$ having a common control qubit c which target the different $2n$ qubits $b_{2n-1}, \ldots, b_0$

Stages 2,4 and 5 have the same pattern of operation: a common qubit c successively controls n or $2n$ qubits which are adjacent each other. This pattern of operation can be performed with local communications as shown in Figure 5.7 where CNOT gates are used in stages 2 and 4, while controlled W_l gates are used in stage 5, and W_l is given by Eq. (4.20)

Stages 1 and 3 have a different pattern of operation. Namely, the n qubits $a_{n-1}, a_{n-2} \ldots, a_0$ concurrently interact with the n qubits $b_{n-1\oplus l}, b_{n-2\oplus l}, \ldots, b_{0\oplus l}$ in $2n$ steps where $l = 0 \ldots 2n-1$ denotes the step. The $\oplus$ symbol denotes addition modulo $2n$. On other words, all the a_l qubits interact with n qubits b_j in a rotating pattern, e.g. for a_2 control qubit the sequence of the target qubits is $b_2, b_3, b_4, b_5, b_0, b_1$.

The above concurrent operation of gates in rotating pattern can be accomplished with local interaction and in linear depth with the circuit "rc" of Figure 5.8 for the case $n = 3$.

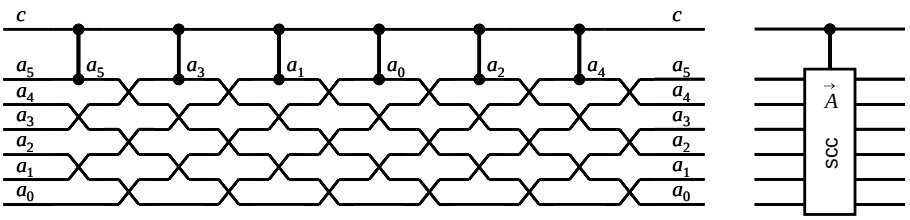


Figure 5.7: Shifted control circuit "scc" for commuting gates. The gates applied between qubit c and a_j mutually commute.

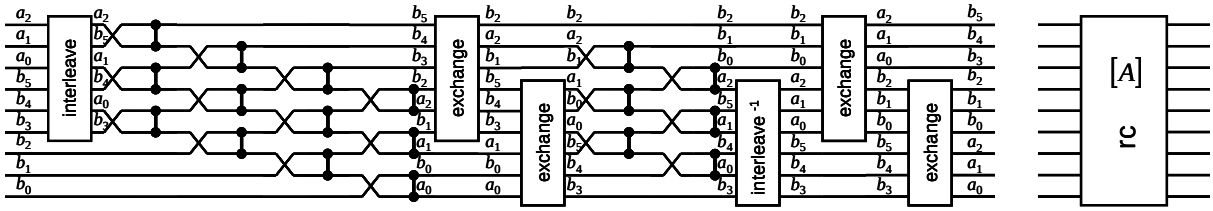


Figure 5.8: Rotating pattern control circuit "rc" using local interactions.

The first "interleave" block interleaves the upper b_5, b_4, b_3 qubits with the a_2, a_1, a_0 qubits. The first triad of two-qubit gates correspond to the gates $V_5^{(2)}, V_4^{(1)}, V_3^{(0)}$ of Figure 4.15. The three following SWAP gates allow the local interaction between the three pairs of qubits $a_2 - b_4$, $a_1 - b_3$ and $a_0 - b_2$, that is the local operation of gates $V_4^{(2)}, V_3^{(1)}, V_2^{(0)}$ of Figure 4.15, etc. This sequence of swapping and concurrent gates application cannot proceed further so as to allow the interaction of a_2 with b_1 and b_0. Also, it remains for qubit a_1 to interact with qubits b_0 and b_5, while for qubit a_0 to interact with qubits b_5 and b_4. The purpose of the next two "exchange" blocks is to reshuffle the qubits in order to achieve this. The inverse "interleave" blocks and the two "exchange" blocks at the end reorder all the qubits in their initial configuration.

The topology of Figure 5.8 can be used for both stages 1 and 3 of the ΦMAC with suitable choice of two-qubits controlled rotation gates. The depth of this rotating pattern control circuit is $12n - 10$.

The parameter $[A]$ of the symbol "rc" is a matrix of $n \times 2n$ blocks, where each block is a 4×4 matrix representing a two-qubits gate. In the case of stages 1 or 3, the element $[A]_{jl}$ corresponds to the matrix of controlled rotation described by $V_l^{(j)}$ of Eq. (4.18) or $V_l^{(j)\dagger}$ of Eq. (4.19), respectively.

Combining the previous results we can build a ΦMAC circuit that uses local communications as depicted in Figure 5.9. The first stage is implemented with a rotating pattern control circuit "rc" with parameter $[V]_{jl} = \mathrm{diag}(I, V_l^{(j)})$ as described above. The second stage is an "scc" block with parameter $\vec{X} = (CNOT, \ldots, CNOT)$. The third stage is an "rc" block with parameter $[V]_{jl} = \mathrm{diag}(I, V_l^{(j)\dagger})$ and the fourth stage is the same as the second. It remains to apply the fifth stage which can be implemented with an "scc" block with parameter $\vec{W} = (W_{2n-1}, \ldots, W_0)$ where W_l is given by Eq. (4.20). Yet, the qubits order must be rearranged so as the b qubits of Figure 5.9 are adjacent to the control qubit c. This is accomplished by the two "exchange" blocks that precede the last "scc" block. The two last "exchange" blocks finalize the order of the qubits in their initial position.

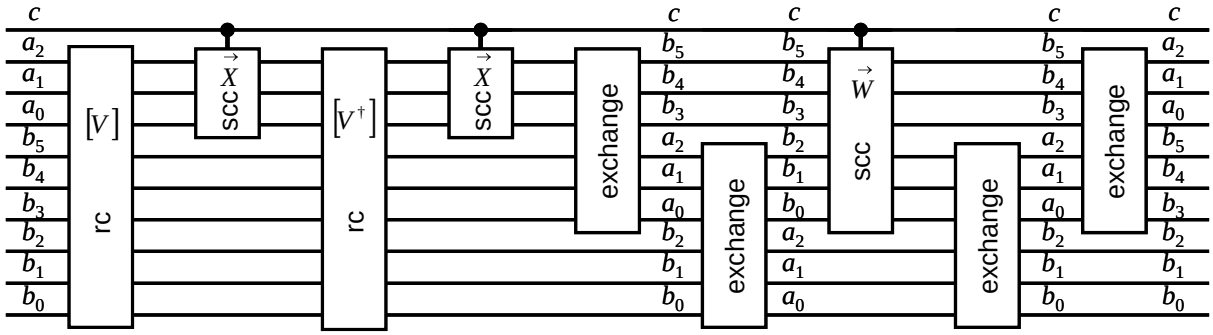


Figure 5.9: ΦMAC circuit using local communications.

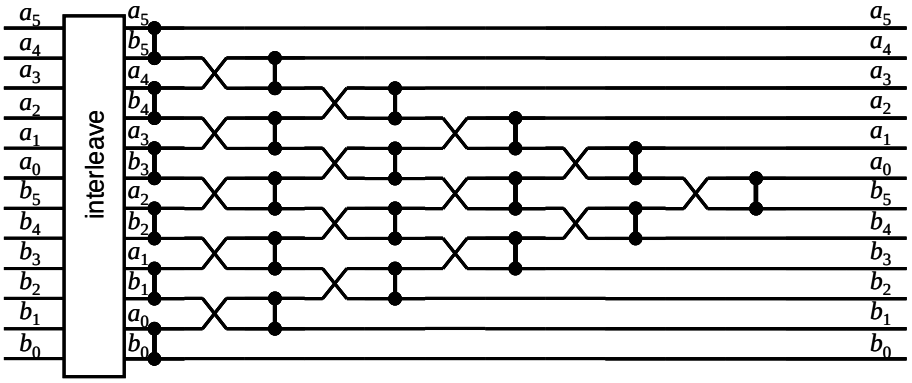


Figure 5.10: ΦADD on two 6 qubits integers using local communications.

The depth of the localized ΦMAC of Figure 5.9 is $40n-4$ compared to the depth $8n$ of the original ΦMAC of Figure 4.15, that is about five times deeper.

In order to retain the linear depth of the modular multipliers of Chapter 4 when implementing them in a 1D-NTC architecture, localization of other blocks except the ΦMAC is essential. The GMΦDIV block which is used in the modular multipliers consists of the ΦMAC and also of QFT,ΦADD (adder of two quantum integers) and CΦADDC (controlled adder of a quantum integer with a constant). Below we describe local interaction versions of the adder ΦADD and the QFT. An inspection of Figure 4.7 shows that the the case of the ΦADDC is handled with the topology "scc" of Figure 5.7. This scheme is slightly different form the one proposed in [130] as we want at the end to retain the qubits order. Thus, the depth of a localized ΦADDC becomes $2n$ instead of n.

Figure 5.10 is the localized version of the ΦADD of Figure 4.9 (see also [130]). Controlled rotation gates c-R_k are applied between pairs of qubits a_j and b_l for $k=1\ldots n$. For a particular k, $n+1-k$ pairs a_j and b_l interact, namely the pairs that their index j and l satisfy $j-l=k-1$ and $l=k-1$.

The "interleave" block in the circuit of Figure 5.10 makes adjacent all the pairs a_j and b_l with $j=l$ so as all the c-R_1 can be applied concurrently. Afterwards, SWAP gates make adjacent qubits a_j and b_l with $j=l+1$ for $j=0\ldots n-1$ so as to prepare them for the concurrent application of the gates c-R_2. This scheme is repeated until the final application of the gates c-R_n. The depth of the localized ΦADD is $4n-2$ compared to the depth n of the original ΦADD.

Finally, a localized version of the QFT [130] is shown in Figure 5.11. Hadamard gates are included in this design and qubits labeling is attached after each SWAP gate to help keep track the operation. Observe that this circuit achieves at the same time the required qubit reversal of QFT. The depth of the localized QFT is $3n-2$.

The localized version of the GMΦDIV dividers (Figures 4.16-4.17 or 4.24) is easily constructed by substituting the ΦMAC,ΦADD,ΦADDC and QFT blocks with their localized

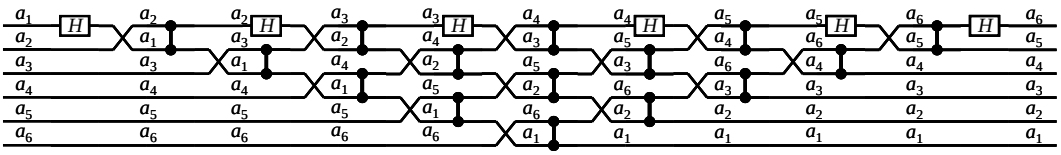


Figure 5.11: QFT on 6 qubits using local communications.

version exposed above. Also, some additional ”exchange” blocks are needed to make adjacent various registers that are distant during the computation. A rough estimation of the overall depth of the improved GMΦDIV2 divider for a divisor of n bits is about $400n$ steps of which about $40n$ steps are due to exchange operation of registers.

Similar calculations for the modular multiplier ΦMUL_MOD2 of Figure 4.23 gives a depth of about $1800n$. Consequently, the full localized version of the modular exponentiation circuit ΦEXP2 becomes $3600n^2$.

Note that the above depth calculations are over-estimated because they don't take into account possible depth absorption of SWAP gates with adjacent two-qubit gates using *canonical decomposition* [162, 163, 130]. An example is given in Figure 5.12. The canonical decomposition of both circuits has almost identical topology; the difference is the angle parameters used in the single and two-qubit gates of the decomposition. Thus, addition of a SWAP gate before or after a controlled rotation gate has inappreciable effect on the depth of the circuit. Such a configuration of SWAP gates is encountered in part of the localized ΦMAC in Figure 5.8, the localized ΦADD in Figure 5.10 and the localized QFT in Figure 5.11 .

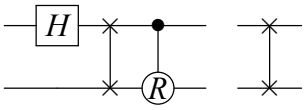


Figure 5.12: These two circuits have almost equal depth.

6. HIERARCHICAL SYNTHESIS OF QUANTUM AND REVERSIBLE ARCHITECTURES

A novel graph-based hierarchical synthesis methodology for arbitrary large and irregular quantum and reversible architectures is presented in this Chapter. An architecture is prescribed as a sequence of elementary operations that correspond to existing quantum or reversible components of a library. The library can be populated with new circuits synthesized by the same or other methods in a multilevel hierarchical synthesis setup. This Chapter closely follows the presentations in [164, 165].

6.1 Background and related work

While a reversible circuit operates on classical bits, e.g. on variables taking discrete values 0 or 1, a quantum circuit operates on qubits taking values in a continuous range (namely the surface of a sphere called Bloch's sphere). Moreover, the reversible logic gates are a subset of the quantum gates (the reversible gates can be described by matrices having elements the integers 0 or 1). Nevertheless, quantum circuit synthesis can exploit known reversible circuit techniques as many quantum algorithms use arithmetic and logic operations (they are Boolean). An example is Shor's algorithm whose main parts are: (a) a modular exponentiation computation and (b) a quantum Fourier transform (QFT). The former part can be described in integer arithmetic terms on the computational basis and, therefore its construction can take advantage of reversible synthesis techniques, something that cannot be applied to the latter part of the QFT. In the former case a reversible circuit implementing the function can be invoked and then a transformation to the quantum circuit can be applied using the available quantum gates.

In general, the reversible synthesis methods can be divided in two families: (a) optimal or asymptotically optimal and (b) heuristic. The former methods result in a circuit that minimizes a particular cost factor which is usually the number of gates. Optimal methods are practical for a few bits only (e.g. 3 or 4 bits) as they demand exponentially grown memory and time as a function of inputs [166, 167, 168]. Heuristic synthesis methods behave better referring to the bits handling capacity at the cost of relaxing the optimality requirement. Transformation [169, 170], search [171], cycle [172] and Binary Decision Diagrams (BDD) [173] based methods fall under the latter category. A thorough overview can be found in [30]. In general, most of the methods suffer from limited scalability: they do not handle large circuits of more than 100 bits due to restrictions of memory and runtime as they consume exponential resources in arbitrary examples cases.

Quantum synthesis differs from reversible synthesis in the specifications and the libraries used to synthesize the circuit. Boolean specifications in the computational basis are adequate when the target circuit is an arithmetic one or logical one due to the linearity and the superposition principle. Thus, reversible circuit methodologies can be used and then library transformation can be applied to convert from the reversible library to a quantum library. When the specifications are in the form of a unitary matrix U of dimension $2^n \times 2^n$ for a circuit consisting of n qubits then decomposition methods can be applied [53, 27, 174, 28]. In such methods the unitary U is decomposed in a sequence of single qubit and two qubit gates where the specific gates depend on the library. Gates number is exponential in n in general.

As discussed above the various existing quantum and reversible synthesis methods need exponential computing resources and thus they do not scale well for large circuits. A step towards synthesis of large circuits would be the combined use of hierarchical methods where the circuit, being reversible or quantum, is built level by level using already synthe-

sized functions of smaller blocks. These blocks can be ad-hoc designed if they correspond to well known circuits involving regularity (e.g. adders or other arithmetic circuits), automatically synthesized by a synthesis method chosen by the user, or synthesized by the proposed method if a bottom up design is needed to handle a complex specification. To our knowledge only a few hierarchical synthesis method have been presented in the literature [175, 176, 177, 178, 178]. The proposed hierarchical synthesis method applies to both reversible and quantum circuits and eliminates intermediate bits/qubits without excessive ancilla usage. It also supports unlimited circuit size handling capability on an existing library of components. It applies to the quantum case whenever a classically defined ”oracle” arithmetic function must be embodied in the quantum algorithm.

6.2 Methodology Basics

In this section, we present a set of inter-operating algorithmic routines which synthesize a complex reversible or quantum circuit using abstract functional blocks. Fig. 6.1 outlines the basic steps of the methodology. The abstract blocks are part of a library and are assumed to be already synthesized (using our method or other lower level methods). The specifications of the target architecture come in the form of sequence of arithmetic/logical instructions. The circuit is synthesized progressively in three steps. In the *forward synthesis* part a directed acyclic graph representing the required computations by the specifications is built by interconnecting various library blocks and possibly by adding ancillae (whenever temporary variables are used), without taking into account the resetting of the ancillae. Subsequently, possible *deadlocks* which prevent the next step are detected and eliminated. The last step, *reversing*, is the expansion of the graph so as to reset the ancilla states

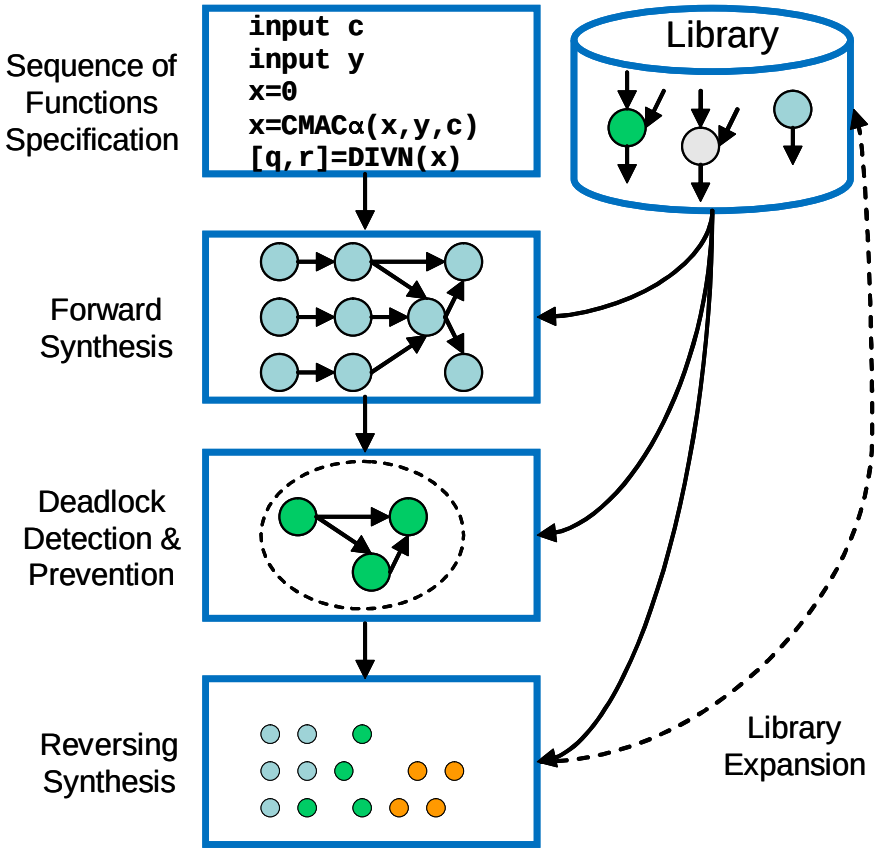


Figure 6.1: High level description of the proposed synthesis methodology.

6.2.1 Initial Specifications and Library

We consider sequences of arithmetic and logical operations describing the reversible circuit or the quantum oracle of the general form:

$$x = f_i(x, b) \tag{6.1}$$

Function f affects only one of its two input variables x and b. Index i is an identifier used to distinguish among the various available functions in the library. Variables x and b are integers of n_x and n_b bits, respectively. We call variable x the affected variable and b the control variable. There are special cases of elementary functions that fall under the description of Eq. (6.1). In the simplest case there is no control variable ($n_b = 0$) in the computation of a primitive assignment such as $x = NOT(x)$. In other cases, the bits of the variables are partitioned into sets, where each set has its own index as shown in equation (6.2).

$$\left[x_1^{(out)}, \ldots x_k^{(out)}\right] = f_i\left(\left[x_1^{(in)}, \ldots x_l^{(in)}\right], [b_1, \ldots b_m]\right) \tag{6.2}$$

In this case, output variable x (denoted as $x^{(out)}$) is partitioned in k subsets of bits, each one indexed as variable $x_i^{(out)}$, $i = 1, \ldots, k$ and consists of n_{x_i} bits. Similar description applies to variables $x^{(in)}$ and b. As an example consider the function of dividing a $2n$ bits affected input variable by a constant integer resulting in an n bits quotient and an n bits remainder whenever the quotient is less than $2n$. In this case $k = 2$ and $l = 1$ and the bits representing the dividend become the quotient and remainder bits of the output.

We assume that a library of quantum or reversible subcircuits implementing the classical functions in the form of Eq. (6.1) or (6.2) like the one proposed in [179] is available. This library can be also viewed as the instruction set of a *quantum arithmetic logical unit* (QALU) and the result of our synthesis procedure can be viewed as the sequence of executions of the quantum instructions to various quantum registers of the QALU. An example quantum library of arithmetic and logical functions is given in Table 6.1. Various quantum circuit representations of these functions can be found in the literature or can be synthesized

Table 6.1: Example functions of a Quantum Library.

Quantum Function	Affected Qubits			Control Qubits	
	Input State	Output State	Size	State	Size
INP_F	-	0 or x	n	-	0
NOT	x	$\bar{x}$	n	-	0
CNOT	x	$x \oplus b$	n	b	n
COPY	$x = 0$	$0 \oplus b = b$	n	b	n
ADDC$_a$	x	$x + a (\mathrm{mod} 2^n)$	n	-	0
CADDC$_a$	x	$x + ca (\mathrm{mod} 2^n)$	n	c	1
ADD	x	$x + b (\mathrm{mod} 2^n)$	n	b	n
CADD	x	$x + cb (\mathrm{mod} 2^n)$	n	b, c	$n, 1$
MAC$_a$	x	$x + ab (\mathrm{mod} 2^{2n})$	$2n$	b	n
CMAC$_a$	x	$x + cab (\mathrm{mod} 2^{2n})$	$2n$	b, c	$n, 1$
DIV$_a$	x	x/a, $x \bmod a$	$2n$	-	0
OUT_F	x	x	n	-	0

with known methods to further populate the library. The library can be arbitrarily extended by including more complex functions or even new functions synthesized by the algorithm described in this paper or other methods.

Each function shown in Table 6.1 transforms the quantum state (*input state*) of a qubits collection to another quantum state (*output state*). This transformation depends on the state of some other qubits which remain unaltered. We call the qubits that get transformed *affected* and the qubits that remain unaltered but influence the affected qubits *control* qubits, in correspondence to the affected and control variables of function f in equation (6.1), respectively. In Table 6.1, for each quantum function shown in the first column the following columns show the number of affected qubits (size), their initial state and the output (transformed) state. The last columns show the state and the number of the control qubits. Since the state of the control qubits remains unaltered, Table 6.1 does not distinguish between input and output states for these qubits. The inverses of the functions of Table 6.1 (which are inherently reversible) are not shown, but are also included in the library.

6.2.2 Quantum Dependence Graph

The synthesized quantum circuit is represented as a directional acyclic graph (*Quantum Dependence Graph* or QDG) consisting of nodes corresponding to the quantum subcircuits (blocks) of the library functions and of arcs corresponding to the individual qubits or groups of qubits (qubit buses) connecting these blocks.

Figure 6.2 clarifies with an example the notation and the labels used in a QDG. Figure 6.2.i is a quantum addition circuit block in standard notation implementing the function ADD(x, y). Qubits of input y remain unaffected at the output as the block is reversible and thus they correspond to control qubits. Figure 6.2.ii is the same block in a more compact form with the qubits organized in buses and connected in different ports. Figure 6.2.iii represents the same block as a QDG node and its incoming and outgoing arcs. Attached to the node are the `type` label which is equal to value ADD and the `id` label (it depends on the relative position in a particular QDG). The bottom left arc corresponds to the qubits carrying the x state and has three labels attached: The width of the arc (`width`) which is equal to the number of the qubits n, the port destination label `dest` which is equal to 1 as there is only one affected input port for this block and the port source information (`source`) which depends on other nodes of the QDG. Similar remarks hold for the rest of the labels

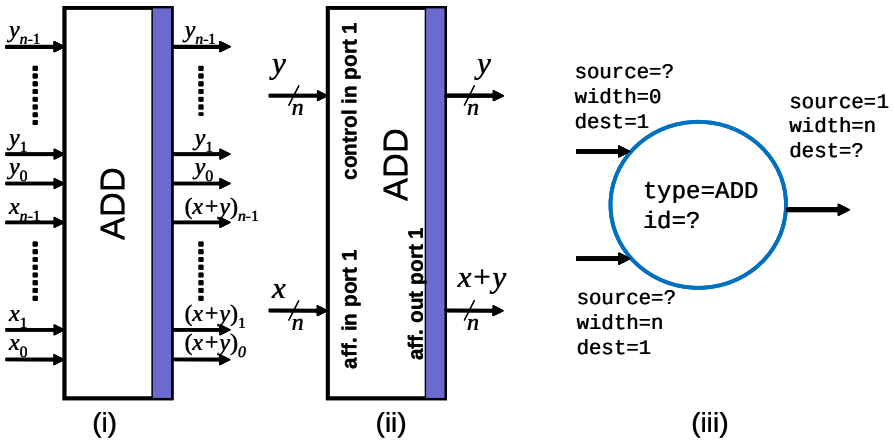


Figure 6.2: Representation of a quantum functional block in the QDG notation. (i) Functional block showing all the qubits taking part in the operation along with their input and output states, (ii) the same block with the qubits organized in buses connected to ports, and (iii) the abstract notation of the same block as a node with arcs and their labels. The question marks mean that the respective label depends on the specific connections of the node relative to the other nodes of the QDG.

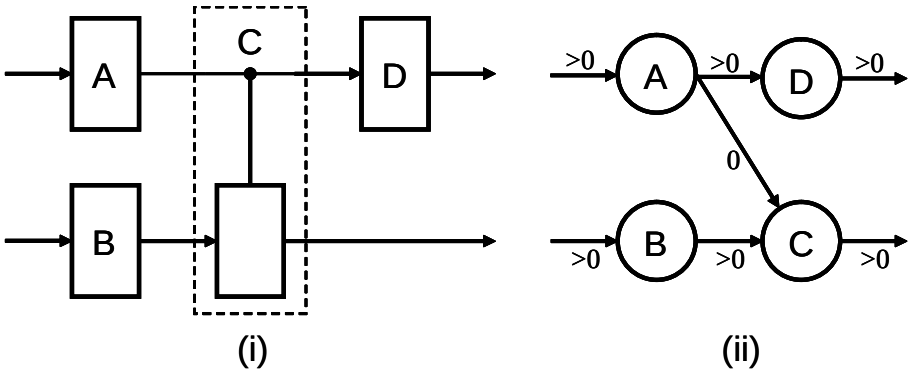


Figure 6.3: Mapping between the standard notation (i) and the QDG notation (ii). Affected arcs have width >0, while control arcs have width=0.

attached to the other two arcs (control input arc and affected output arc).

The labels `id` and `type` can be represented by integers, for each type there is its negative type which corresponds to the block performing the inverse function. Also, the constant parameters of some of the blocks (e.g. parameter a of the function MAC_a) are assumed to be included in the type label. Later, a third label named `anc` (integer with values 1 or 0) will be used also.

The affected qubits (or affected qubit buses) transformed by a node are represented by arcs incoming to and outgoing from that node. Each affected arc, either incoming or outgoing, must also include the port source (arc tail connection) and port destination (arc head connection) information, because some nodes may have more than one input and/or output qubits buses and we need to distinguish the various possible ports of each node.

Similarly, the input control qubit buses of a quantum function are represented as incoming arcs to the corresponding node. Control arcs always have a width of 0 (no matter their real qubits width) so as to be distinguished from affected arcs. The tail of control arcs emerge from affected qubits output ports of an ancestor node. As control qubits are not altered by any node they entered, there is no need to show their exit by an outgoing arc. Similarly with the case of the affected qubits arcs, we need to include in each control arc the port source and port destination information.

Input nodes of the graph represent initial qubit states (quantum variables passed to the quantum algorithm represented by the QDG) or ancilla qubits initially set to a zero state. For both cases, a node with the special type INP_F is used in the graph. Similarly, output nodes of the graph represent output qubits states (final results or ancilla states). The ancilla states correspond to garbage qubits states or ancilla qubits reset back to their initial zero state when the reversal procedure described later is applied. The output nodes are represented with a node of the special type OUT_F which acts as an identity node.

In some of the functions, the control qubits are grouped in different states variables. The same applies for the affected qubits in some of the functions. As an example, the controlled adder CADD has the two groups of control qubits b and c, of n qubits and one qubit, respectively. Also, the divider function DIV has $2n$ qubits wide input state, but the output state is grouped in two qubits buses of n qubits, namely the quotient and the remainder. In general, we allow such qubit grouping in the functions of the library because it facilitates the initial specifications.

The separation of the qubit buses into affected and control ones simplifies the internal representation of the circuit and the workings of the synthesis algorithm, especially the

deadlocks detection developed later. The restriction that a node control input does not exit the same node does not contradict the standard notation of a quantum circuit; it is just a remapping of notation as shown in Figure 6.3.

6.3 Forward QDG Synthesis

The first phase (see Fig. 6.1) of the QDG construction is dedicated exclusively to the forward computations, without taking into account the resetting of the possible ancillae qubits. This step is trivial and will be explained in short.

6.3.1 Representation of Classical Algorithm

Dependencies among the sequence of functions of the classical algorithm to be mapped as a quantum oracle exist when a variable in the list of affected output variables of a function is used as an input variable (affected or control) in a subsequent function. These dependencies will be reflected in the QDG through the use of an arc connecting two nodes. Initial values and input variables of the algorithm correspond to affected output ports of `INP_F` nodes, while the variables giving the final results (desired and garbage) of the algorithm correspond to affected input ports of `OUT_F` nodes. Intermediate variables (temporary) used in the algorithm for the calculations of the final results correspond mainly to ancilla qubits.

An arbitrary classical algorithm using elementary functions of the form of equation (2) can be equivalently described by arrays of integers and arrays of lists of size L, where L is the total number of functions comprising the algorithm. An integer array `type` describes the type of each function, arrays of lists `p`, `m` and `c` describe the lists of affected output, affected input and control input variables, respectively. Another array of integers, named `w` describes the number of bits used by each variable. Last, array `res` will discriminate which of the variables used in the algorithm are the desired final results and which are intermediate temporary variables. This last array definition is crucial for the final phase of the synthesis algorithm whose purpose is to reset intermediate garbage.

6.3.2 Forward Synthesis Algorithm

The main data structures used in the synthesis of the forward computations QDG are the graph structure itself, named `forwQDG`, and the arrays `type`,`p`,`m`,`c`,`w` and `res` describing the classical algorithm mentioned in the previous subsection.

The purpose of forward synthesis algorithm is to add nodes to the `forwQDG` (initially null), one for each function found in the classical algorithm and connect them with affected and control arcs based on the dependencies between the variables. In brief, the synthesis algorithm of the forward QDG consists of the steps shown in Table 6.2 and explained below. An example of a part of forward QDG built by such an algorithm is shown in Figure 4.

The synthesis algorithm executes the for loop of size L (lines 1,5). For each integer `l` (`l`$= 1 \ldots L$) the following steps build gradually the forward QDG:

Line 2 Add a new node in the forward QDG. This node has a type label equal to the type of the function (`type[l]`). A new node id is assigned sequentially for each node added.

Line 3 Scan the list of control input variables `c[l]` of this function. Then for each control variable in the list, find every function `k` that includes this variable in its output variable list `p[k]` and connect with an arc the two respective nodes of the QDG corresponding to these two functions, `l` and `k`. As the arc connecting the two

Table 6.2: Forward QDG Synthesis Algorithm

Operations
1: FOR each line l DO
2: Add node
3: Add incoming control arcs to node
4: Add incoming affected arcs to node
5: END FOR
6: Add terminal (`OUT_F`) nodes
7: Record the garbage terminal nodes in a list

nodes is related to a control input connection add a width label of value 0 on the arc. Add source and destination port labels reflecting the input and output ports that are connected by this arc. The position of the variable in the lists `p[k]` and `c[l]` is the source and destination port number, respectively.

Line 4 Scan the list of affected input variables `m[l]` of this function. Then for each affected input variable find every function that includes this variable in its output variable list `p[k]` and connect with an arc the two QDG nodes which correspond to these two lines, `l` and `k`. Add labels on this arc reflecting the number of the qubits carried by this variable (`w[m[l]]`) and also the source and destination port similarly to Step 2 above. If no line with such an output variable is found then add a new node of type `INP_F` and make the required connection with the relevant labels (This means that the input variable is an input argument to the classical oracle to be synthesized).

After the execution of the loop, two more steps are necessary to prepare the reversing phase of the synthesis:

Line 5 For all unmatched affected output variables (this means that the variable is a final desired result or an ancilla output) add a new node of type `OUTP_F` and make the required arc connection assigning the relevant width and port labels.

Line 6 Record in a list (`GarbageTermList`), the terminal node ids of the forward QDG which carry garbage results, that is non desired final results. This discrimination is based on the array `res` mentioned in the previous subsection.

6.4 Reversible QDG Synthesis

The final phase of our synthesis algorithm transforms and expands the forward QDG so as to reset all ancilla qubits back to their initial constant states.

6.4.1 Node Inversion

The un-computation of the ancilla qubits state back to a constant state can be achieved by successively inverting the states of ancilla qubits that appear at the output of a terminal node through all the nodes up the respective input nodes affecting these ancilla qubits. These terminal nodes have been recorded in `GarbageTermList`.

To understand the requirements and the procedure to invert the state of the affected output qubits of a node in a QDG, we refer to Figures 6.4 and 6.5 which represent a segment of an example forward QDG and its expansion (called `revQDG`), respectively. The purpose of the expanded QDG is to uncompute the garbage ancilla states.

In Figure 6.5, nodes A^{-1},B^{-1},C^{-1},D^{-1} and H^{-1} are the nodes required to invert the output

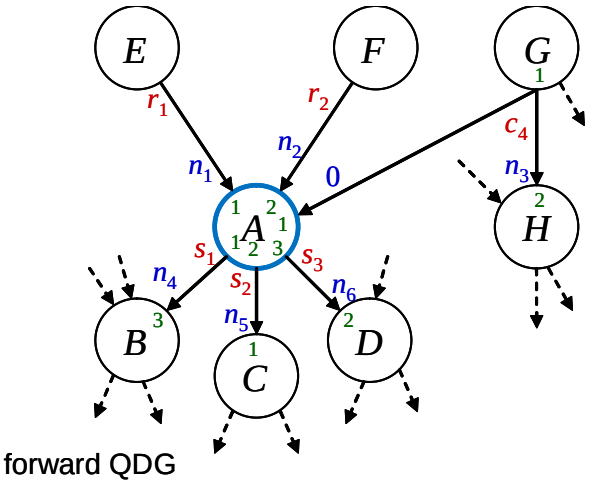


Figure 6.4: Part of an example forward QDG node. Attached at the tail of the solid arcs is the output state and at the head of the arcs is the width of the arc (0 for control arc). Inside the circles of the nodes the port numbers for each case of affected input, affected output and control input arcs are shown.

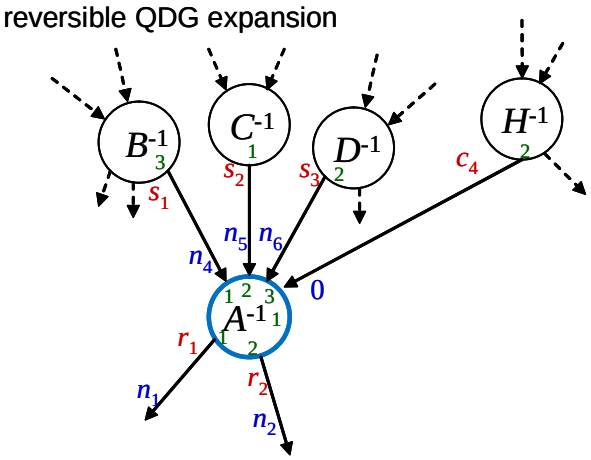


Figure 6.5: Inversion of node A of the example forward QDG shown in Fig. 6.4. Legend of arc and node labels is similar as that of Fig. 6.4.

states of node A of the forward QDG in Figure 6.4. Inverting states s_1, s_2, s_3 of the output qubits of node A means to transform them into the states r_1 and r_2. Figure 6.4 shows that node A receives as affected inputs two arcs (with widths n_1 and n_2) from nodes E and F being in states r_1 and r_2, respectively. These are controlled by state c_4 (arc of width 0 emerging from node G) and transformed into the output states s_1, s_2, s_3. The inverse transformation is realized by another node of the reverse QDG, namely node A^{-1}, which is the inverse of node A (as we have previously mentioned, the quantum library contains the inverse of every function as well). So, if the affected input ports of node A^{-1} are fed with the states s_1, s_2, s_3 and its control input is fed with the state c_4 it is obvious that the required inversed states r_1 and r_2 become available at the affected output ports of node A^{-1}.

This inversion implies that states s_1, s_2, s_3 and c_4 must be available. In the example we have shown for the forward QDG states s_1, s_2, s_3 have already been processed by the successor nodes of A (nodes B,C, D) and the state c_4 has been processed by node H. This necessitates the inversion of nodes B, C, D and H before the inversion of node A.

The incoming arcs connections to node A^{-1} are as follows. Node A has two ports of af-

fected input qubits, namely 1 and 2 of respective qubits width n_1 and n_2, and a unique control input port labeled as 1. Moreover, it has three affected output ports (1, 2 and 3) of widths n_4, n_5, and n_6, respectively. The ports of the affected input qubits of the inverted node A^{-1} are the ports of the affected output qubits of node A and vice versa. Control port 1 of A^{-1} corresponds to the same port (number 1) of node A. When connecting the incoming arcs of node A^{-1} the algorithm needs the extra information of which ports are engaged in these connections.

6.4.2 Global Considerations

The above per-node inversion procedure must be applied by taking global considerations into account. Some prerequisites and constraints are the following:

- *Selection of nodes which require inversion*: Only some of the forward computation QDG nodes need to be inverted. The reversing algorithm must select and label the nodes of the forward computation that need inversion (using label `anc` with values 0 or 1 attached at each node). The reversing algorithm begins from the nodes listed in `GarbageTermList` and recursively marks all the internal nodes of the forward QDG that have a path connection to these leaf nodes as the nodes that require inversion. These paths must comprise exclusively of affected arcs, i.e. it marks only the ancestors of the output nodes that directly transform the final ancilla state. The nodes which need inversion will be called *ancilla nodes* and their `anc` label is set to value 1 (the rest of the nodes have a value 0 and will be called *non ancilla nodes*). This prerequisite to mark the ancilla nodes of the forward QDG will be taken into account later, at line 1 of the reversing algorithm shown in Table 6.6.

- *Sequence of inversion*: the algorithm must check that each candidate node for inversion is ready and allowed for this operation. Only a subset of the ancilla nodes is able to be inverted at each instance. This is due to the existence of data dependencies between the various nodes. In the example of Figure 6.4, node A can be inverted only if its children nodes are already inverted (in case they were ancilla nodes) because node A^{-1} requires the states s_1, s_2, s_3. These states are not available as the forward computation has already transformed these states by applying nodes B, C and D. The readiness condition just described is given in line 8 of the reversing algorithm in Table 6.6.

  Even if an ancilla node is ready for inversion, a postponement of this action may be necessary. This may happen if this blocks the inversion of other nodes. E.g. the candidate node for inversion has a control arc connection towards another ancilla node which is not yet inverted. The purpose of updating the `GarbageTermList` in line 11 of Table 6.6 is exactly this.

- *Tracking of intermediate states*: The inversion algorithm needs to keep track of which forward QDG node output state corresponds to the new output states computed by each new inverted node. Referring again to Figure 6.5, when node A^{-1} is added to invert node A, the necessary information of where to find the states s_1, s_2, s_3 must have been recorded. These states have been computed in a previous step of the algorithm when the inversion of nodes B, C and D took place, by adding nodes B^{-1}, C^{-1} and D^{-1} and their corresponding arcs, so these latter nodes can supply the required states s_1, s_2, s_3.

  An array of lists named `revinfo` is used for this purpose. A list is assigned to each node of the `forwQDG`. Initially, the lists corresponding to the terminal nodes of the

forwQDG contain as records the same terminal nodes (meaning that the output states of the terminal nodes are available at the nodes themselves), while the revinfo lists of the internal nodes are empty (meaning that the states of the internal nodes of the forwQDG are no longer available as the forward computation has proceeded to the end). Additionally, the records of a list contain arcs information such as the source port, destination port and width. During the inversion of the nodes procedure, as the revQDG is expanded, states of internal nodes of the forwQDG gradually become available (new terminal nodes in the revQDG appear) while states of other nodes are no longer available (they become internal nodes of the revQDG). Therefore, the revinfo lists must be updated anytime a node is inverted and this update becomes part of the node inversion algorithm shown in Table 6.3.

- *Deadlocks resolution*: There are cases where it is impossible to invert a given forward computations QDG without applying a certain transformation on it. These cases are again related to the data dependencies imposed between two nodes which prevent the inversion of another node. Treatment of these situations is described in detail in the following subsection.

Table 6.3: Node Inversion Algorithm

Operations	
1:	ADD New Node of type A^{-1} in revQDG
2:	ADD affected arcs in revQDG from nodes of RevInfo(A) list towards node A^{-1}.
3:	FIND the ancestors F_1,...,F_m of node A that have affected arc connections with A.
4:	UPDATE the lists RevInfo(F_1),..., RevInfo(F_m) with the new added node A^{-1}.
5:	FIND all the nodes CA_1,..., CA_n, which control node A.
6:	ADD control arc connections in revQDG from nodes CA_1,..., CA_n to node A^{-1}.

6.4.3 Deadlocks Resolution

There are two possible cases that can lead to deadlock of the reversing algorithm. The two cases are depicted in Figures 6.6 and 6.7. In both cases we assume that we have already marked all the nodes of the QDG as ancilla or non ancilla nodes.

The sequence of deadlock resolution algorithms is to first apply the algorithm for the second type and afterwards the algorithm for the first type, as it is possible the revocation of a second type deadlock to generate a first type deadlock, but not vice versa.

Both deadlock resolution algorithm use additional ancilla bits/qubits and copy the output port states of the nodes that cause the deadlocks to these new ancillae. This is a bitwise copy operation which is simply performed with CNOT gates having as control the bit state to be copied and as target the new ancilla in the zero state.

Although a real copy operation is not permitted in the quantum context due to the ”no-cloning” theorem, the purpose here is to reset an intermediate ancilla state back to zero. If someone analyzes the global operation of the circuit he can see that as long as the garbage states ends up in state 0 for every basis input state then the same holds for every superposition of them. That is the garbage states become 0 and disentangled from

the other states. Consequently, the circuit operation is the desired one when analyzed globally (see Section 6.6 for more details).

In both deadlock cases I and II the drawback of adding more ancilla qubits can be counterbalanced by the reversing of other ancilla qubits which otherwise would be garbage qubits and could not be reused. This point will be further analyzed in Section 6.6.

Deadlock Type I

The handling of the first deadlock type is depicted in Figure 6.6. Nodes A and B are marked as ancilla and are connected by an affected arc (width greater than zero). Nodes C_1 and C_2 are marked as non-ancilla and the arcs connecting node A with them have also a width greater than zero (n_1 and n_2) and emerge from different ports of node A, namely p_1 and p_2. It is the affected arc between nodes A and B the fact that gives the ancilla property to node A and consequently the necessity to invert it; its other children connections are towards the non ancilla nodes C_1 and C_2.

The deadlock condition then arises due to the fact that a node like A has affected arc connections to both ancilla and non ancilla children. It is impossible to invert the ancilla node A as this requires the prior inversion of nodes C_1 and C_2, but these nodes must not be inverted as they are non ancilla nodes and their output results must remain unaltered up to the end of the computations. This kind of deadlock can be prevented by "copying" the qubits emerging from ports p_1 and p_2 and thus releasing node A so as its output states can be inverted as desired. The detailed required actions to revoke such a deadlock case are described in Table 6.4 and Figure 6.6.

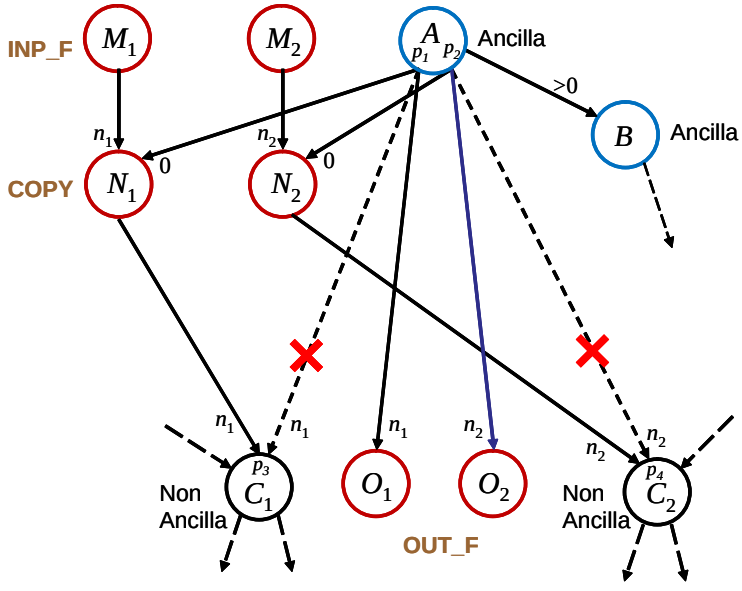


Figure 6.6: First type of deadlock resolution. Nodes A and B are ancilla, nodes C_1 and C_1 are non-ancilla and M_1,M_2,N_1,N_2,O_1,O_2 are the nodes added to prevent the deadlock. Next to each arc is shown its width. Ports are shown inside the circles of some nodes.

Deadlock Type II

The second type of deadlock is illustrated in Figure 6.7. Node A which is non-ancilla controls via an arc (width 0) emerging from port p_1 of ancilla node B. As the latter node is an ancilla node it must be inverted. This inversion need means either that the forward computations must not have proceeded beyond node A (towards nodes E, F, G and H) or that

Table 6.4: Detection and Resolution of Deadlock I Algorithm

Operations
FOR each ancilla node A of `forwQDG` DO
IF node A has affected arcs connections to both ancilla and non anciila children nodes THEN
FOR each affected outgoing arc i emerging from node A (port p_i) and leading to a non-ancilla node C_i DO
ADD a new node, M_i, of type INP_F initializing n_i qubits in zero state. The number n_i is the width of the arc $A \rightarrow C_i$
ADD a new node, N_i, of type COPY whose purpose is to copy the port p_i output state of node A
ADD an affectecd arc connection from node M_i to node N_i of width n_i (the number of qubits to be copied). The port information attached to this arc is trivial (1 for both the tail and head) as an INP_F node has only one output port and a COPY node has only one affected input port
ADD a control arc connection (width 0) from node A to node N_i. The tail of this arc is port p_i of node A and the head is port 1 as a COPY node has only one control input port
ADD an affected arc connection from node N_i to node C_i. Arc's destination port is the destination port of arc $A \rightarrow C_i$ and its width is n_i. The source port of this arc is again 1 as only one output port exists on any COPY node
REMOVE the arc connecting node A with node C_i
ADD an OUT_F node
ADD arc connection from port p_i of node A to node OUT_F
MARK node OUT_F as ancilla node
END FOR
END IF
END FOR

the output state of port p_1 of A must be available somewhere else in the QDG.

The deadlock condition arises whenever a path like $A \rightarrow E \rightarrow F \rightarrow G \rightarrow H$ (Path 1 in Figure 6.7) consists exclusively of affected qubits arcs (width greater than zero), the nodes belonging to the path are non-ancilla and simultaneously exists a second path from B to H (Path 2 in Figure 6.7) where the subpath $B \rightarrow D$ consists of affected arcs while the last arc $D \rightarrow H$ is supposed to be a control arc. The nodes belonging to path $B - D$ are assumed to be ancilla nodes. Both paths must emerge from the same output port (shown in Figure 6.7 as p_1) of node A.

If the last arc $D \rightarrow H$ wasn't a control arc then this case could be handled by the resolution of type I deadlock because in such case node D, being ancilla node, would have another outgoing arc of width greater than zero leading to another ancilla node. For the same reason, node A is supposed to be a non ancilla node, otherwise we would face an ancilla node having both ancilla and non ancilla children connected through affected arcs.

This second deadlock condition can be justified for the following reasons. If the inversion of node B is done prior the advancement of the forward computations beyond node A (towards node H) then the output state of node D will not be longer available and the

computation on node H could not be done. On the other hand as explained above, if the forward computation has advanced up to node H (so as node D can be inverted) then the inversion of node B cannot be done as the output state of node A is no longer available.

An algorithm for detection of second type of deadlocks has been developed and is briefly described in Table 6.5. A detected deadlock can be revoked with similar actions as those of the first deadlock case, that is addition of nodes M, N, O and some rearrangement of arcs as depicted in Figure 6.7. Detailed resolution operations are not exposed in Table 6.5 due to the similarity with first case.

The detection of the second type is as follows: Every non ancilla node A is checked for engagement in a possible deadlock (lines 1,17). For each such node a list, `L0`, consisting of its non ancilla children is built (line 2) and another list, `L1`, consisting of its ancilla children is also built (line 3). The purpose of the double loop defined in lines 4,17 and 5,16 is to check if two arcs emerge from the same port p of node A towards an ancilla and a non ancilla node (lines 6,15). This is a prerequisite for the deadlock of the second kind and this condition corresponds to the arcs $A \rightarrow E$ and $A \rightarrow B$ in Figure 6.7. If such a condition is fulfilled then another list, `nonAncS`, that contains non ancilla nodes is built (line 7). This list contains only the non ancilla successor nodes of node $C0$ and a modified *Depth First Search* procedure can be applied for this retrieval. This modified search procedure traverses only the affected arcs (the ones with their width greater than zero). Now, every path from node A to each node S of the list `nonAncS` corresponds a path similar to Path1 in Figure 6.7. The final check is to find if a second path exists from node $C1$ to any of the nodes recorded in list `nonAncS` (lines 10-13). This path must be composed of affected arcs only except the last one (line 11) and this could correspond to Path 2 in Figure 6.7. If this final condition is true then a procedure similar to that one exposed in Table 6.4 is applied to port p of node A which causes the second kind of deadlock (line 12).

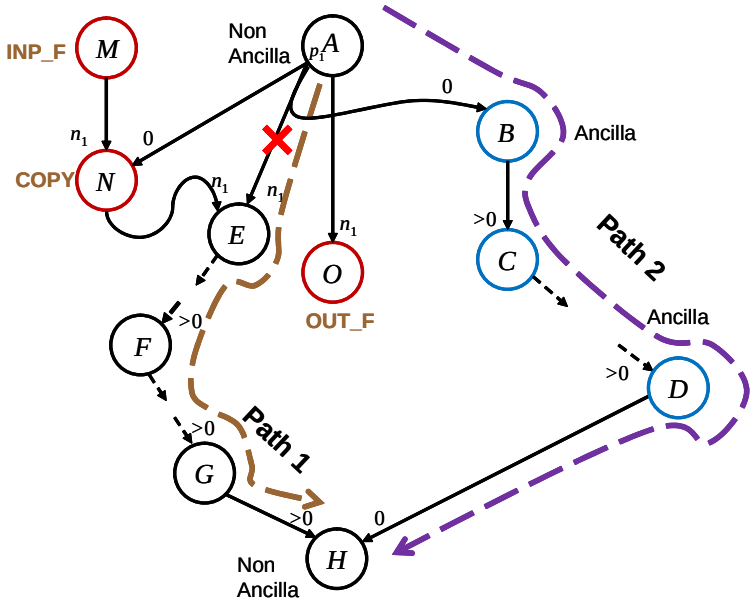


Figure 6.7: Second type of deadlock. Nodes A,E,F,G,H are non-ancilla whereas nodes B,C,D are ancilla. Nodes M,N,O added to prevent the deadlock. Width of each arc is shown.

Uniqueness of the two deadlock conditions

The previous two deadlock types are the only ones that can arise. This can be justified if we examine all the possible connection cases of an ancilla node, e.g. B, which must be

Table 6.5: Detection of Deadlock II Algorithm

	Operations
1:	FOR each non-ancilla node A of the `forwQDG` DO
2:	Generate a list `L0` with the non ancilla children of A
3:	Generate a list `L1` with the ancilla children of A
4:	FOR each node $C1$ of `forwQDG` in `L1` DO
5:	FOR each node $C0$ of `forwQDG` in `L0` DO
6:	IF source(arc($A, C1$))=source(arc($A, C0$)) THEN
7:	`nonAncS`=DFS1($C0$)
8:	p=source(arc($A, C0$))
9:	FOR each S in `nonAncS` DO
10:	path2=FindPath($C1, S$)
11:	IF path2 $\neq$ NULL AND all arcs of path2 are affected except the last one THEN
12:	Deadlock found at port p of node A
13:	END IF
14:	END FOR
15:	END IF
16:	END FOR
17:	END FOR
18:	END FOR

inverted. The necessary conditions to invert node B are: (1) its incoming control states be available at the instance of inversion and (2) its outgoing output states be also available, as explained previously.

The first condition means to investigate the possible cases of ancestor nodes of B that have control arcs connected to it. There are two cases: (1a) an ancestor node A is a non ancilla node and (1b) an ancestor node A is an ancilla node. Case 1a is covered by the type II deadlock. Case 1b means that at least one of the successors of A, e.g. C, with affected arc connection from node A is an ancilla node. If such a connection emerges from the same port as the arc $A \rightarrow B$ then node B can be inverted only if node C can be inverted so as this case is reduced to recursively check if node C is engaged in any deadlock. On the other hand if such a connection emerges from another port of node A then we can see that we fall back in a type I deadlock.

The second condition can be separated in the following subcases: (2a) All the successors of node B are ancilla nodes. This means that this condition can be reduced to assure recursively that the successors are not engaged in any deadlock. (2b) At least one of the successors is a non ancilla node and this case is handled again by the type I deadlock.

Therefore, all the necessary conditions to invert an ancilla node are covered by preventing just the two deadlock cases described previously.

6.4.4 Reversing Algorithm

The actions described previously to reset the garbage states are collected together in Table 6.6. The initialization actions already described are the marking of ancilla nodes of the forward QDG (line 1, subsection 6.4.2), the application of the two deadlock resolution algorithms (lines 2,3, subsections 6.4.3 and 6.4.3) and the initialization of the `revinfo` lists (line 4, subsection 6.4.2).

The forwQDG already modified by the two deadlock resolution procedures is then copied (line 5) to a second QDG called revQDG. New nodes will be progressively added to revQDG, while at the same time ancilla nodes will be deleted from forwQDG. The final synthesis product will be the revQDG graph.

The circular list, GarbageTermList, is used to store the ids of garbage terminal nodes of forwQDG. This list is initialized during the construction of the forward QDG (subsection 6.3) and contains all the terminal nodes of the graph that carry non desired results, that is it initially contains the terminal nodes carrying garbage. This list is updated during the reversion algorithm by removing ancilla node ids just inverted and by adding ancilla node ids which became terminal nodes after the removal of inverted nodes in the forwQDG. The algorithm scans in a circular manner this list until it becomes empty (lines 6,7,13).

Each node id found (curNode) in the circular list is checked if it is ready for inversion in the forwQDG (lines 8,12). This has been explained in *Sequence of Inversion* bullet in subsection 6.4.2. In such a case, a new node with inverse type and its relevant arcs are added in the revQDG (line 9). These steps have been described in detail in subsection 6.4.1.

We then remove this curNode from the forwQDG graph (line 10). This removal is necessary as it may release some other ancilla nodes of the forwQDG with their ids contained in the GarbageTermList to become ready for inversion in the next execution of the loop. Thus, this removal will be taken into account in the updating of the GarbageTermList in line 11. The procedure of updating this list is (i) to remove the curNode and (ii) to add all the ancilla parent nodes of curNode on the condition hey have no children (this is equivalent that they indeed have become terminal nodes after the removal of the curNode from the forwQDG).

A last post-processing step that rearranges some of the control arcs is not shown in Table 6.6. This rearrangement changes the head connections of some control arcs so as to emerge from the new added nodes instead of the original ones and it is based on the revinfo lists.

Table 6.6: Reversing Algorithm

	Operations
1:	MARK ancilla nodes of forwQDG
2:	CALL Deadlock 2 Detection and Resolution procedure
3:	CALL Deadlock 1 Detection and Resolution procedure
4:	INITIALIZE revinfo[] lists
5:	COPY forwQDG to revQDG
6:	WHILE GarbageTermList $\neq$ NULL DO
7:	FIND next curNode in GarbageTermList
8:	IF no children of curNode with affected arcs exist in forwQDG THEN
9:	CALL node inversion procedure (Table 6.3) for curNode of revQDG
10:	REMOVE curNode and its arcs from forwQDG
11:	UPDATE GarbageTermList
12:	END IF
13:	END WHILE

6.5 Synthesis Examples

This section presents two simple but complete arithmetic circuits synthesized by the proposed algorithm to clarify the previously described procedures. The first circuit is a controlled modular multiplier which is an integral part of the modular exponentiation compu-

tation for Shor's algorithm. Various proposals exist for the implementation of this circuit, most of them based on the construction of a modular adder [62, 129]. The example presented is based on a recent efficient design of Shor's algorithm [141] where the building blocks are a multiplier/accumulator by constant and a divider by constant. The example serves only to present the scalability of the proposed method to automatically build hierarchically large circuits given its specification in a classical algorithm and not to evaluate the resulting circuit in terms of quantum gates, circuit depth or qubits count, as these aspects depend on the components used in the library and consequently on the low level synthesis methods employed to generate them.

A controlled modular multiplier calculates the function $x = cay \bmod N$ where a and N are constants of n bits. Control variable c is 0 or 1, x is $2n$ bits wide and y is n bits wide. If a multiplier accumulator by constant a (CMAC_a) and a divider by constant N (DIV_N) are available in a synthesis library like that of Table 6.1 then the computation of the above function can be done as in Table 6.7.

The second column of Table 6.7 is the sequence of the elementary functions while the last four columns are the initial conditions passed in the synthesis algorithm in the form of array of lists as described in subsection 6.2.2. Additional to these initial conditions and not depicted in the table are the array of the variables width and the array defining the final results. These are initialized as follows : $w =$[n,n,2n,2n,n,n]andres=$[0,0,0,0,0,1]$; variables assigned the values 3 and 4 have a width of $2n$ qubits while the desired result is the remainder r which is numbered as the 6-th variable.

The synthesized QDG is depicted in Figure 6.8. It consists of the forward part at the left and the reverse part at the right. Three INP_F nodes are used in the forward part to represent the initial states c,y and $x = 0$ with width 1, n and $2n$, respectively. Nodes OUT_F with id 6 and 7 are the output nodes added by the forward synthesis algorithm and carry the garbage quotient state q and the desired state of the remainder $r = cay \bmod N$.

The purpose of the reverse part of the synthesis is to reset the garbage state q back to a constant zero state. This means reversing node with id 6 and all its ancestors nodes (1,2,3,4 and 5). All these nodes have been marked as ancilla nodes by the forward synthesis algorithm, but nodes 1,2 and 3 did not need reversion since they are INP_F nodes. A checking for possible deadlocks must be made before initiating the reverse algorithm. As can be seen in the figure, a deadlock of type I is found at node 5 as it is an ancilla node connected through affected arcs to ancilla node 6 and non-ancilla node 7. The actions of the deadlock algorithm is to add nodes with ids 8, 9 and 10, add some arcs as described in subsection 6.4.3 and remove the arc 5$\rightarrow$7. After these actions take place nodes 1, 2, 3, 4, 5, 6 and 10 become ancilla. Reversion algorithm results in the right part of Figure 6.8. Nodes with ids 11, 12, 13 and 14 are the inverses of 10, 6, 5 and 4, respectively.

Table 6.7: Specifications of a controlled modular multiplier

	Algorithm	Initial Conditions			
Line `l`	Function	`type[l]`	`p[l]`	`m[l]`	`c[l]`
1	input c	INP_F	1	-	-
2	input y	INP_F	2	-	-
3	input $x = 0$	INP_F	3	-	-
4	$x =\text{CMAC}_a(x, y, c)$	CMAC_a	4	3	1,2
5	$[q, r] =\text{DIV}_N(s)$	DIV_N	5,6	4	-

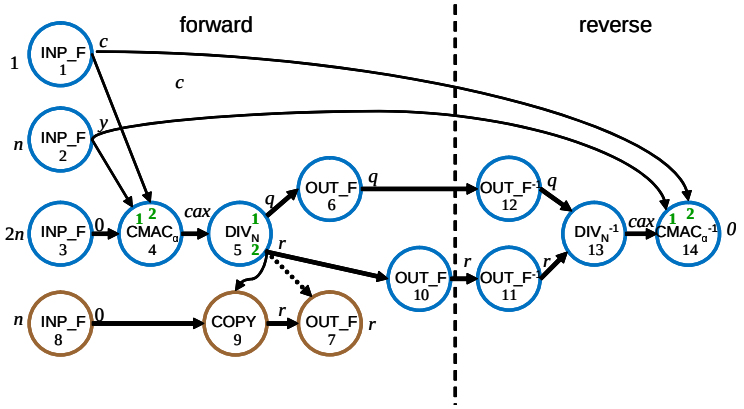


Figure 6.8: Quantum or reversible architecture result in the form of QDG (forward and reverse) for the controlled modular multiplier. Inside each node the function type and the id are shown. Next to each arc the state it carries is shown. Thick and thin arcs are affected and control arcs, respectively. Ports numbering is shown inside the node, when necessary.

Subsequently, the complete QDG calculates the desired function and the resulted circuit is identical to controlled modular multiplier accumulator proposed in Figure 4.22.

This example was a trivial one as it does not exhibit any advantage over the standard Bennett's method discussed in next session. In fact, the circuits derived by the two methods are identical. A second example follows which exhibits some advantages of the proposed method.

Assume that the target function to synthesize is defined as

$$f(x,y) = \begin{cases} y + b \cdot x, & y < a \\ y, & y \geq a \end{cases} \tag{6.3}$$

for some constants a and b, essentially being a conditional multiplier/accumulator conditioned on the value of one of the inputs y. Input arguments x and y and the constants a and b are n bits wide integers, while the output $f(x,y)$ is $2n$ bits wide and signed integers are assumed. The above function is not invertible in any of its arguments x or y, thus both the arguments must be supplied at the output of its reversible implementation.

It seems that the above function requires a controlled multiplier/accumulator block CMAC_b and a constant adder block ADD_{-a} operating as a comparator in order to implement the piecewise function f. The result of the comparison would be the most significant bit of the subtraction $y - a$ as signed arithmetic is assumed. We introduce two dummy blocks (they do not perform any computation) that are useful for extraction and combination of group of bits. The first one is a splitter (called SP) which partitions a group of n ordered bits into two groups of k and $n-k$ bits, by preserving their order. A splitter node with parameters k and $n-k$ has an input port of n bits and two output port of k and $n-k$ bits as depicted in Figure 6.9(i). The k most significant bits appear at output port 1 in the same order, while the $n-k$ least significant bits appear at the output port 2, again in the same order. Its operation can be rigorously defined as

$$[y,z] = SP_{k,n-k}(x), \quad x = (x_{n-1} \ldots x_0), \quad y = (x_{n-1} \ldots x_{n-k}), \quad z = (x_{n-k-1}, \ldots x_0) \tag{6.4}$$

The inverse block of the splitter $\text{SP}_{k,n-k}$ is the combiner $\text{CB}_{k,n-k}$ which is depicted in Figure 6.9(ii). Equipped with these functions and a COPY function we can specify the input for

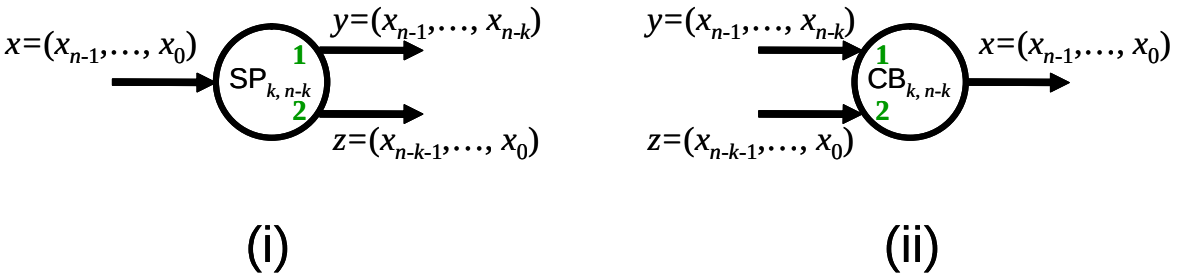


Figure 6.9: Splitter (i) and Combiner (ii) blocks.

Table 6.8: Specifications of a conditional multiplier/accumulator. Bit widths of the variables x, z, y, s are $n, n, n, 1$ respectively.

	Algorithm	Initial Conditions			
Line `l`	Function	`type[l]`	`p[l]`	`m[l]`	`c[l]`
1	input x	INP_F	1	-	-
2	input $z = 0$	INP_F	2	-	-
3	input y	INP_F	3	-	-
4	input $s = 0$	INP_F	4	-	-
5	$y =$ADD$_{-a}(y)$	ADD$_{-a}$	5	3	-
6	$[y_u, y_l] =$SP$_{1,n-1}(y)$	SP$_{1,n-1}$	6,7	5	-
7	$s =$COPY(s, y_u)	COPY	8	4	6
8	$y =$CB$_{1,n-1}(y_u, y_l)$	SP$_{1,n-1}$	9	6,7	-
9	$y =$ADD$_a(y)$	ADD$_a$	10	9	-
10	$y =$CB$_{n,n}(z, y)$	CB$_{n,n}$	11	2,10	-
11	$f =$CMAC$_b(y, x, s)$	CMAC$_b$	12	11	1,8

the synthesis of function $f(x, y)$ as shown in Table 6.8.

The purpose of lines 5,6 and 7 of Table 6.8 is to extract to variable s the sign bit of the comparison $y - a$. This sign bit will discriminate if the multiplier/accumulator is enabled or not. After the operation of lines 8 and 9, variable y has its initial value and it is almost ready to be processed by the CMAC function. Before the final processing by the CMAC an extension of its width from n bits to $2n$ bits is required. This is accomplished by combining the n bits variable z, initially in zero state, with y. Line 10 uses a combiner and results in a variable y of $2n$ bits wide with the same value as the initial value of the original y. The last line results in the final value $f = y + bx$ if $s = 1$ $(y < a)$, or just y if $s = 0$ $(y > a)$.

The forward synthesis QDG corresponding to Table 6.8 appears in Figure 6.10. Node CNOT which carries the sign bit requires inversion. This node is engaged in a deadlock II as it is controlled by non ancilla node (SP with id 6) and also controls the non ancilla node CMAC which affects the same qubits with SP.

The QDG after this deadlock II resolution is depicted in Figure 6.11 with the insertion of one ancilla qubit (node INP_F with id 15).

A deadlock I came to the front after this resolution. This happened because nodes SP (id 6) and ADD$_{-a}$ now became ancilla nodes. The deadlock I is due to node SP which feeds with affected qubits the non ancilla node CB (id 8). The QDG after this second deadlock is depicted in Figure 6.12. This resolution needed the insertion of $n - 1$ bits (node INP_F with id 18).

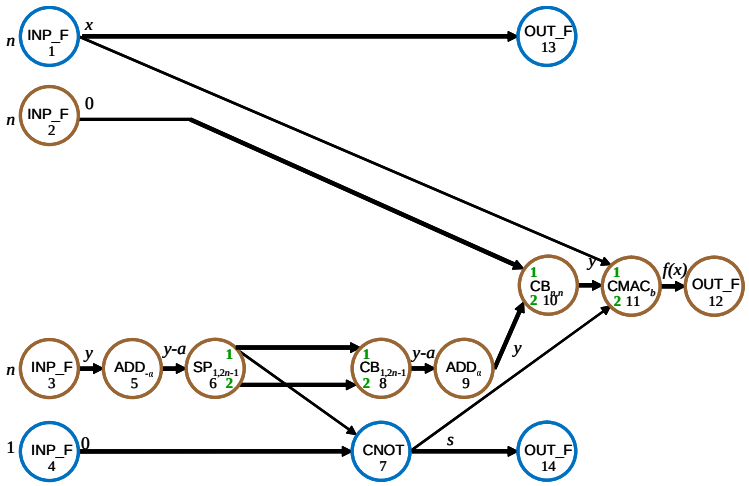


Figure 6.10: Forward Synthesis result for the conditional mutltiply/accumulate example.

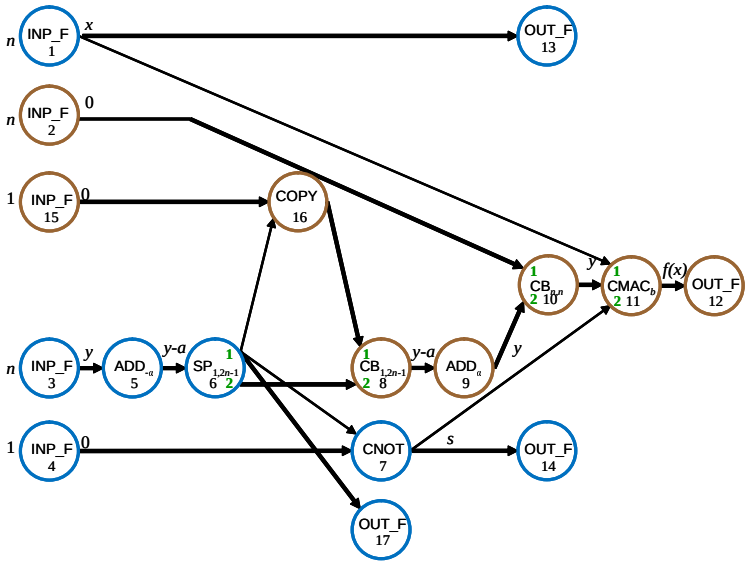


Figure 6.11: Deadlock II resolution for the conditional mutltiply/accumulate example.

The final stage, which is the reversion, results in the QDG of Figure 6.13. This reversion step resets the qubit carrying the sign and brings back the argument y. This procedure used the inverses of nodes OUT_F (it is an identity node) and CNOT which are themselves, the inverse of the subtractor (node id 5) which is an adder (node id 27) and the inverse of the splitter (node id 6) which is the combiner (node id 26).

The required effective blocks (dummy blocks like OUT_F, SP, CB are not counted as they don't have any cost) for the implementation of function f in a reversible manner with the presented method are three constant adders of n qubits, a CMAC and $n+1$ CNOT gates (hidden inside the COPY blocks). The dominant cost and depth contribution is due to the CMAC block. The standard implementation with Bennett's method discussed in next section, would require two CMAC blocks, four adders and n CNOT gates, roughly doubling the gate cost and the depth of the circuit. Moreover, the space requirement of the implementation just presented is $4n+1$ bits, while the standard method would require $5n+1$ bits.

Figure 6.14 better clarifies the above comparison between the standard Bennett method and the proposed method. Both subfigures use the standard block notation used throughout this thesis. Subfigure 6.14(ii) is stripped out from dummy blocks used in the synthesis

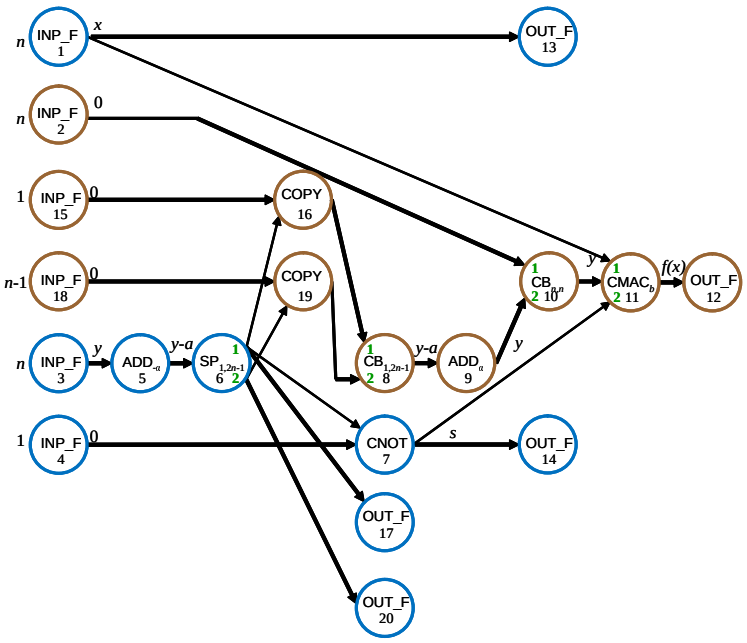


Figure 6.12: Deadlock I resolution for the conditional mutltiply/accumulate example.

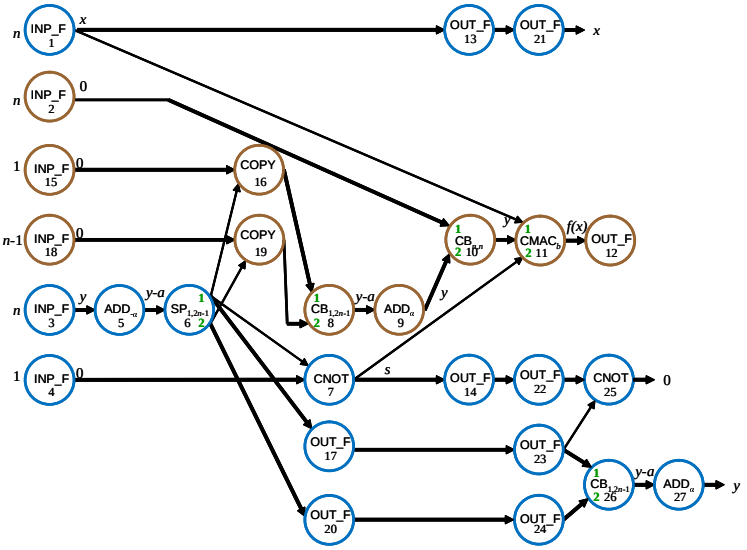


Figure 6.13: Complete synthesis of the conditional mutltiply/accumulate example after the final reversion procedure.

procedure like IN_F,OUT_F, SP and CB. The constant adders and the CMAC units are the ΦADDC and ΦMAC, respectively, introduced in Chapter 4, although other implementations could be used instead. To simplify the notation, the required QFT and inverse QFT blocks are absorbed in the ΦADDC and ΦMAC symbols. Thus, the depth of the complete ΦMAC is $8n + 2 \cdot 4n = 16n$ (it requires two QFT of width $2n$ and the depth of an QFT with width n is $2n$) while the depth of the complete ΦADDC is $4n + 1$. Taking these depths into account, we conclude that the depth of the second circuit is $28n + 6$ compared to $48n + 7$ of the first one. Similar calculation for the quantum cost gives that the proposed method derives a circuity with cost $9n^2 + 20n$ compared to $16n^2 + 32n$ of the standard method.

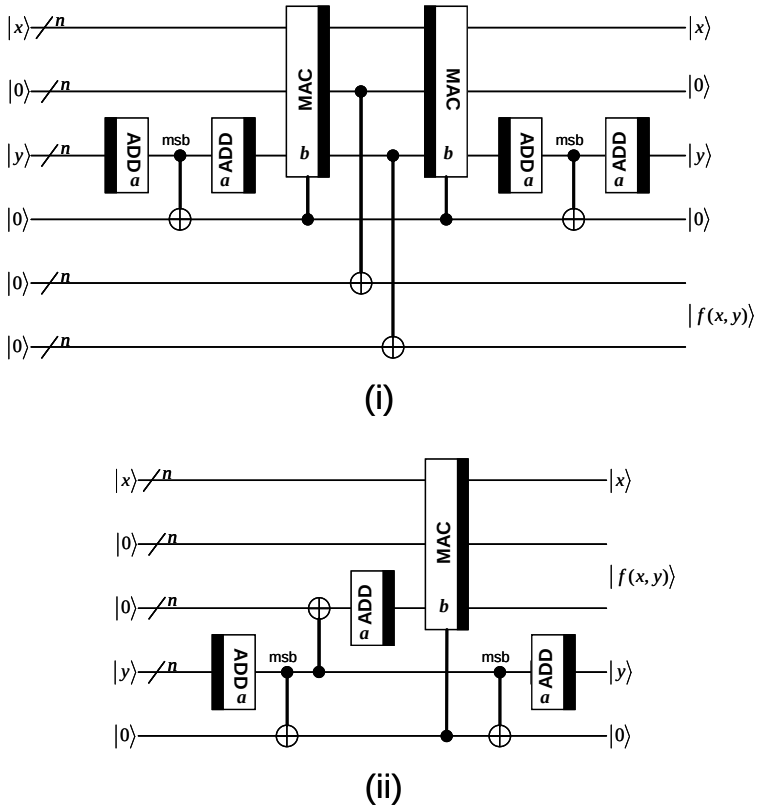


Figure 6.14: Comparison of two circuits computing the conditional multiply/accumulate example. (i) Circuit derived by the compute-copy-uncompute method and (ii) circuit derived by the proposed method.

6.6 Features and Comparison

In general, a direct comparison of a hierarchical synthesis method to flat (low-level) synthesis methods in terms of quantum cost, qubits count and circuit depth is not meaningful because a hierarchical synthesis is based on a previously synthesized library and the results depend on the particular low-level synthesis methods used to build the library. As for the execution time of the proposed hierarchical synthesis algorithm, it is obvious that the hierarchical technique of dividing a large circuit to smaller parts, gives a scalability advantage over a low-level strategy to handle a large and complicated circuit by a flat method.

A rough complexity analysis for the synthesis algorithm in terms of the number of lines L of the specifications follows. It is assumed that the number of input/output ports of a node is constant and much smaller than L (a reasonable assumption as shown in Table 6.2). The number of nodes of the forward QDG is L while that of the final reverse QDG is at most double. With the above assumptions we can estimate that the arcs number is $O(L)$. We concentrate on the reversing part as the forward part is easily shown to have a complexity $O(L)$ by investigating Table 6.2. The main part of the reversing algorithm consists of operations in lines 7-11 with constant complexity nested in a while loop scanning the ancilla nodes. Thus it has a $O(L)$ complexity. It can be seen that the most computation intensive part is the deadlock II resolution procedure. The algorithm of Table 6.5 consists of two nested loop each of complexity $O(L)$ (line 1 and 9); the other loops have a constant complexity due the above assumptions. Combining the nesting of the FindPaths procedure which has a linear complexity too, we conclude that the complexity of deadlock II detection is $O(L^3)$. This is the dominant complexity for the whole synthesis algorithm.

We present below a different kind of analysis that is related to the garbage generation and indirectly related to the ancilla requirement. A top level view of a reversible or quantum circuit U with its respective input and outputs signals is shown in the left part of Figure 6.15. Input bits/qubits are discriminated in argument input x and ancilla input initially in a constant state, usually zero. The ancilla qubits are used internally to assist the computation. On the other hand, output bits/qubits are discriminated in the desired output $f(x)$, that is the target of the computation, ancilla output which is usually part of the ancilla input being reset back to its initially state and the garbage output $g(x)$ which depends on the input argument x and thus is not constant as the ancilla output. The garbage output contains intermediate results of the computation. When the function embedded in U is not invertible (e.g. addition of two non-constant inetegers is not invertible) the elimination of the input argument is impossible [31].

The garbage output is an undesired effect of the computation which is dependent on the input argument. This means that in the case we refer to a classical reversible circuit it cannot be simply ”erased” as this would contradict the notion of a reversible circuit. On the other hand, in the quantum circuit case the garbage output is entangled with the desired output and a possible quantum measurement to ”erase” it would affect the useful desired output. The repeated use of a circuit such as that of U then would require an accumulation of ancilla wires usage. On the other hand, if all the ancilla input wires emerge as ancilla output wires then they could be reused on successively usage of the circuit block. Consequently, it is important to eliminate the ancilla usage as much as possible because this means lower cost in terms of wires which is an important factor especially in the quantum circuit design domain.

A well known technique (Bennett’s trick [31]) to eliminate the garbage, excluding the input argument, is depicted in Figure 6.15. The output wires of U are copied onto new ancilla wires and then the inverse circuit U^{-1} is applied to the outputs (desired, garbage and ancilla) of U. The final result of this processing eliminates any garbage $g(x)$ as shown below and leaves only the input argument x:

$$(x,0,0) \xrightarrow{U} (f(x),g(x),0,0) \xrightarrow{copy} (f(x),g(x),0,g(x)) \xrightarrow{U^{-1}} (x,0,f(x)) \tag{6.5}$$

In the above formula, x belongs to an orthonormal set (e.g. computational basis) and the same applies for the states $f(x)$ and $g(x)$ as f and g are unitary transformations in the quantum case. Although the no-cloning theorem does not allow a general copy operation for arbitrary states, it allows such an operation for orthonormal states, in our case $f(x)$. Consequently, as we can eliminate with this method the garbage state $g(x)$ for every input state x in the computational basis, the same holds for an arbitrary of input superposition of the computational basis.

Bennett’s method to eliminate intermediate garbage doubles the cost in the number of gates and depth as the inverse U^{-1} circuit must be added. Another feature is that as many ancilla wires as the desired output wires are required to hold the copy this output. This is an important disadvantage especially in the cases where the number of garbage wires to be eliminated are smaller the number of the desired output wires. This can happen when a circuit is broken down in multiple levels of hierarchy for reason already explained.

In contrast, the proposed method selectively copies only the wires that are engaged in the two kinds of deadlocks described in subsection 6.4.3. Essentially, it applies Bennett’s trick locally on wires that cause deadlocks to the inversion procedure, instead to apply it globally on the total number of desired output wires. A study of the conditions leading to

deadlocks shows that the total number of ancilla wires needed for the deadlocks resolution is always less than or equal compared to original globally applied Bennett's trick. Even if there is no gain in the ancilla usage, it is obvious that there is gain in terms of the circuit size and its depth, as there is no need for the application of the whole inverted circuit $U^{-1]}$ but only addition of locally inverted nodes.

The proposed synthesis algorithm has been implemented (initially in MatLab) and successfully applied on various examples, including complex and irregular circuits such as the divider by constant used in the implementation of Shor's factorization algorithm presented in Chapter 4. In this specific example no gain has been observed in terms of the qubits size ($6n$ qubits required for an n bits constant divider) due the presence of a deadlock. But compared to the standard Bennett's trick the quantum cost and depth is reduced by about 25%.

Studying other hierarchical methods appeared in the literature as part of integrated platforms we can see that the drawback of RevKit [178, 180] is the excessive generation of garbage bits for intermediate results which are not reset back to constant value [181]. The approaches of [175] (CTQG part of ScaffCC) don't reduce ancilla significantly and the connections between the modules must be done by the user in a description language (structural synthesis approach as opposed to our behavioral synthesis approach). On the other hand, Chisel-Q [177] and Quipper [176] exploit the globally applied Bennett's method on each block of the hierarchy and thus in general it requires more ancilla qubits.

6.7 Conclusions

We have presented a generic hierarchical method for the synthesis of arbitrary large and irregular arithmetic and logical quantum and reversible architectures. The architecture is specified as a sequence of elementary operations that correspond to existing quantum or reversible components of a library. The library can be populated with circuits synthesized by the proposed method, or by any other method, permitting multilevel hierarchical synthesis of any depth. As parts of the library could be used the synthesis output results of tools like [178, 180] for the reversible case or tools like [175, 176, 179, 177] for the quantum case by invoking these tools as back-end and passing them the parameters of the required parts (function type, input and output size). Another option could be the integration of the proposed method in the above mentioned tools.

Hierarchical synthesis methods for quantum and reversible architectures offer several advantages compared to flat gate level methods in the following aspects: (a) easier descrip-

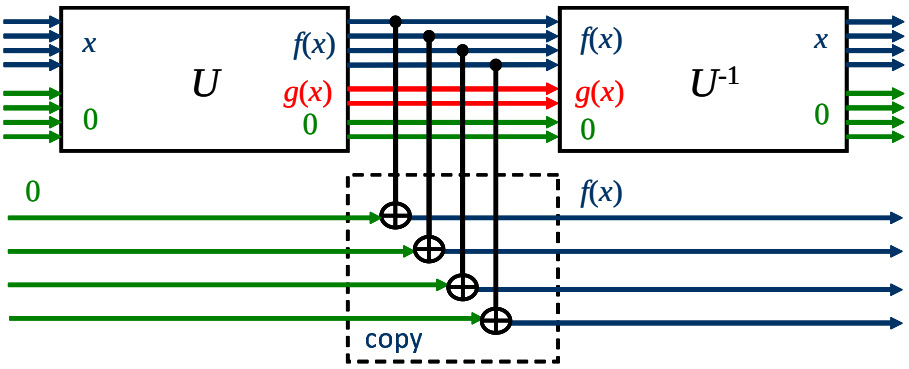


Figure 6.15: Input and output wires definitions of a reversible/quantum circuit U (input argument x, ancilla input and output 0, desired output $f(x)$ and garbage output $g(x)$) and garbage elimination (except the input argument) using Bennett's trick of copying the output and applying the inverse U^{-1}.

tion of complex circuits, (b) efficient handling of arbitrary large size circuits and (c) short synthesis run-time even for significantly large circuits. Our hierarchical synthesis method, when combined with other low-level synthesis methods can deliver architectures in short time, and compared to previous hierarchical synthesis approaches it has the important advantage that it does not pollute the synthesized architecture with excessive numbers of ancilla wires.

7. CONCLUSIONS AND FUTURE WORK

In this thesis we have proposed novel quantum arithmetic circuits based on the quantum Fourier transform representation of an integer or a superposition of integers; circuits are utilized for a novel, depth efficient realization of Shor's algorithm and can be also utilized in other algorithms realization such as the Quantum Phase Estimation or the Hidden Subgroup Problem.

We have also proposed a high level synthesis methodology (by generalizing the techniques used in the design of the quantum arithmetic circuits). We presented a formal procedure which uses directed acyclic graph descriptions and hierarchically synthesizes complex high level circuits based on a quantum library of components. This high synthesis procedure offers advantage over the standard Bennett's method application applied on a straightforward hierarchical synthesis.

7.1 QFT based arithmetic circuits

The first circuit is a controlled multiplier by constant and accumulator (ΦMAC) using $3n+1$ qubits and having a depth of $8n$, where n is the bit width of the multiplication operands. An uncontrolled version offering a depth of $2n$ is easily obtained.

The second circuit (GMΦDIV) is a divider by a constant integer which computes both the quotient and the remainder. This circuit is inspired by an algorithm for classical computation given by Granlund and Montgomery [122]. The divider circuit can be operated in two modes:

- Generic divider GMΦDIV1 with no restrictions on the range of the $2n$ qubits dividend and the constant n bits divisor. In this mode the depth is about $148n$ the space requirement is $12n+1$ qubits (of which $8n+1$ are ancilla qubits). Quantum cost counted as the number of single or two-qubit gates is about $176n^2$.

- Constrained divider GMΦDIV2 of $2n$ qubits dividend and n bits constant divisor. In this case the quotient must be a priori constrained to be less than 2^n. In this special mode of operation a depth of about $74n$ can be achieved, whereas the space requirement is $6n+1$ qubits (of which $4n+1$ are ancilla qubits). Quantum cost counted as the number of single or two-qubit gates is about $44n^2$.

The divider can be promoted to a controlled version with a slight modification using one more controlling qubit. The overhead in cost and depth is minor.

The multiplier/accumulator and the divider can be combined to construct a controlled modular multiplier. Two kinds of modular multipliers can be derived depending on the choice of which divider is used:

- A generic controlled modular multiplier (ΦMUL_MOD1) computing $|x\rangle \rightarrow |a^x \bmod N\rangle$ without restrictions in the range of integer x it gets as input and the constants a and N. A ΦMUL_MOD1 circuit for n qubits $|x\rangle$ and n bits width constants a and N, has a depth of about $650n$, requires $14n+2$ qubits (of which $8n+1$ are ancilla) and has a total quantum cost of about $750n^2$ gates.

- An optimized circuit (ΦMUL_MOD2) suitable for Shor's algorithm with improved depth of about $350n$, improved qubits requirement of $8n+2$ and a total quantum cost of about $200n^2$ gates.

The construction of the modular exponentiator block required by Shor's algorithm using the optimized controlled modular multiplier (ΦMUL_MOD2) achieves a depth of $700n^2$ using $8n+2$ qubits, where n is the bits width of the integer to be factored by the algorithm. Such a modular exponentiation quantum circuit is among the fastest of the literature requiring linear space, especially if we are restricted in physical implementations which require the localized interactions between the qubits. One of the reasons is the regular structure of the ΦMAC circuit that permits the localization of the interaction with no significant overhead in depth. The usage of arbitrary angle rotation gates disadvantage with respect to their fault tolerance capability can be addressed by approximating their angles in a way that the depth advantage is sustained and still the performance, in terms of probability of success, remains acceptable.

Quantum arithmetic circuits based on the QFT representation of integers, instead of the usual computational basis representation, is an alternative implementation that may offer various advantages if used properly. This is due to the fact that two of the main core blocks are the constant adder, which has a constant depth of 1 when the computation is carried out in a datapath that contains an already QFT transformed integer, and the controlled constant adder which has a linear depth of n. By keeping a sequence of computations in such a datapath without reverting back to the computational basis it is possible to maintain a linear depth which otherwise would be impossible. This can be achieved by exploiting properties of the controlled rotation gates such as commutativity, decomposition and suitable rearrangement so as to pipeline their execution. The initial direct QFT and the final inverse QFT does not alter the linear depth as both transforms can be performed in linear depth. Thus, a computation level of hierarchy can be climbed onto (e.g. in our case, addition to multiplication), without any respective time complexity increase.

Another advantage of using QFT based arithmetic is the lower space requirements. This is manifested in Beauregard's modular exponentiation [24] , where $2n+1$ qubits are adequate for the full Shor's algorithm. The reason is that no carry computations are needed in the QFT adder as this is done implicitly with the angle additions. While this advantage is not observed in the proposed modular exponentiation circuit due to the divider complexity, it remains in the multiplier/accumulator ΦMAC where no ancilla qubit is used. Also, robustness of such circuits to gate pruning and rotation angle approximation is observed in various instances.

All these remarks suggest that arithmetic circuits, like the proposed ones, are estimable as building blocks for larger and more complex arithmetic circuits.

7.2 Hierarchical Synthesis

The construction of a complex and irregular quantum arithmetic circuit, such as the proposed divider by constant GMΦDIV of Chapter 4, was the initial motivation to formalize the design procedure for such circuits. Usually, a classical algorithm must be transformed in quantum circuit whose inputs (initial state of the qubits) are fed with the arguments of the classical algorithm (or a superposition of them) and the outputs are the results of the classical algorithm (or a superposition of the results). Thus, the description of the quantum circuit can be given as a sequence of elementary arithmetic and logical operations such as addition, multiplications/accumulations, logical shifts, controlled operations conditioned on the result of a previous operation (if-then-else structures), etc. Provided that, these elementary operations have a quantum circuit counterpart in a library, it is possible to transform them into a larger quantum circuit that performs identical computation to the original classical algorithm in the computational basis. The crucial difference is that while in the classical case the values of the intermediate variables computed can be simply

ignored, in the quantum case these values must be reset back in their initial value. The intermediate variables of the classical algorithm correspond to the ancilla qubits of the derived quantum circuit. A well established method to reset the ancilla qubits back to their initial state is Bennett's trick, that is after the computation, the desired results are copied in the computational basis in a suitable register, and then the whole forward computation except the copying is reversed to give back the initial state of the circuit (uncomputation).

The developed method transforms the initial specifications of the quantum circuit which are given as arrays and arrays of list representing the classical sequence of operation into a directed acyclic graph called forward Quantum Dependence Graph (QDG). The nodes of the forward QDG correspond to the components of a quantum library and they suppose to implement the elementary arithmetic operations. These components could be known constructions from the literature (adders etc), synthesized by other low level synthesis method, or populated by the proposed method applied to a lower level. The arcs connecting the QDG nodes correspond to qubits or quantum registers and they are discriminated in arcs which are affected by their successor node and the ones that control their successor node. The final qubits state of the derived forward QDG describes the desired result along garbage results produced during the computation.

The method adopted to reset the garbage states is to apply uncomputation locally on each node that really needs such an inversion of computation, instead to apply it globally as Bennett's method suggests. Namely, nodes of the forward QDG that are effectively involved in garbage production are marked (these are the nodes which have paths with affected arcs towards final garbage states). These marked nodes of forward QDG are traversed backwards and an inverse of each node is appended to the QDG. The inverse nodes are part of the library as it contains quantum circuits whose inverses are assured to exist.

However, data dependencies between the nodes may not always allow such an inversion, in which case we have a deadlock. Two special procedures are applied to detect and resolve such deadlocks (type I and II deadlocks) before the uncomputation stage. Both procedures have the cost to introduce additional ancilla qubits but they never exceed the additional ancilla qubits that would be needed if Bennett's method would be applied.

The advantage of the overall proposed method over the usual hierarchical methods is that the quantum circuit represented by the final QDG may have smaller depth and quantum cost and may also require less ancilla qubits. The quantum synthesis of a piece-wise function like the one analyzed in section 6.5 is a striking example where a substantial reduction of depth and quantum cost is achieved. Also, in the same example a fair reduction qubits usage is observed.

On the other side, the employment of the proposed synthesis methodology on the construction of the GMFDIV does not exhibit significant improvements in depth and quantum cost compared to the standard method of uncomputation. The extreme case where both methods produce identical circuits is the construction of the controlled modular multiplier/accumulator of Figures 4.19 and 4.22 which correspond to the first synthesis example of section 6.5. The difference between the first and the second example of section 6.5 is that, in the former case a deadlock I is detected, while in the latter case a deadlock II is detected. Moreover, the deadlock II of the second example appears ”early” in the forward QDG, that is the distance of the first engaged node in the deadlock is near the input nodes, while the deadlock of the divider GMΦDIV appears near the end of its respective QDG. Also, the total depth and cost of each node engaged in the deadlock of the piece-wise function example is a significant proportion of the total QDG depth and cost. This is not

the case in the GMΦDIV circuit.

These observations lead to the conclusion that the advantage of the proposed synthesis method arises when deadlocks of type II appear in the forward QDG and this advantage is proportional to the ”size” of the deadlock and the ”earliest” positioning in the overall forward QDG.

7.3 Future Directions

The obvious follow up to the QFT arithmetic circuits would be to exploit them to derive more complex arithmetic circuits, useful for various quantum algorithms, like the constant divider was for Shor's algorithm. In the same branch of interest, the subject of the angle quantization discussed in section 5.1 can be further investigated through simulations. The bounds reported may be loose and better results may be obtained with numerical simulations. Numerical simulations of the full Shor's algorithm, like the ones performed in [158, 159], are difficult for the case of the proposed circuit because of the requirement of $8n + 2$ qubits. For example, to factor $N = 15$ we would need to simulate $8 \cdot 4 + 2 = 34$ qubits. The joint state vector of 34 qubits consists of $2^{34} \approx 16 \cdot 10^9$ complex elements leading to about 128Gbytes of memory when using single precision floating point, only for the state vector. Yet, partial simulations can be proven useful. A simulation to derive distances between the ΦMAC and an approximated ΦMAC are feasible ($3n + 1$ qubits), or even a similar simulation for the whole divider ($6n + 1$ qubits).

Regarding the hierarchical synthesis method, a next obvious step is to develop a complete software which would include front-end and back-end submodules. The front-end must be a compiler accepting the description of the classical algorithm in a suitable language and transforming it in the internal representation required by the synthesis algorithm as described in section 6.3. The back-end must combine the final QDG representation with information stored in the library so as to export the synthesized circuit in a low gate-level description such as in a quantum assembly format. Equipped with such an integrated tool, we could do a more systematic comparison with other high level synthesis tools, although the advantages of the proposed synthesis methodology are clear even without the tool.

A. APPENDIX ON SHOR'S ALGORITHM

This Appendix supplements Chapter 3 with definitions and proofs concerning (i) the probabilistic reduction of the integer factorization problem that Shor's algorithm solves to the problem of period finding, (ii) the method of continued fraction expansion to find the period from measurements acquired by the quantum period finding algorithm, (iii) the probability analysis of the quantum period finding algorithm.

It also deals with the QFT circuit construction of Figure 3.1, and the interpretation of the quantum period finding algorithm as an quantum phase estimation algorithm.

Most of the proofs in this Appendix can be found in [182] and [41].

A.1 Factorization reduction to order finding

Definition A.1. *(Order of element and group). The order* $\mathrm{ord}(a)$ *of an element* a *that belongs to the group* $\langle G, * \rangle$ *is the smallest positive integer* k *such that* $a^k = \underbrace{a * a * \cdots * a}_{k \text{ times}} = e$*,*

where e *is the identity element of the group.*

The order $|G|$ *of the group is the number of its elements*

It is obvious that in a finite order group, every element of the group has a finite order.

Proposition A.1. *(Multiplicative group modulo* N*). The set* $G_N = \{k = 1, \ldots, N-1 : \gcd(k, N) = 1\}$ *equipped with the operation of modular multiplication,* $a * b \doteq (ab) \bmod N$*, is a group. This group contains by its definition all the natural numbers less than and co-primes with* N*.*

Proof. It is sufficient to prove the following properties:

Closure By definition $\forall a, b \in G_N,\ \gcd(a, N) = 1, \gcd(b, N) = 1$. Thus, any non-trivial divisor d of N cannot divide a or b and consequently neither ab. Because $a * b = ab \bmod N$, there exists an integer q such that $ab = qN + a * b$. This means that d does not divide $a * b$ (as long as d is not a divisor of ab, but divides N by assumption). Thus it is proven that $\gcd(a * b, N) = 1$ and combined with the fact that $a * b < N$, the conclusion is that $a * b \in G_N$.

Associativity $\forall a, b \in G_N,\ (a*b)*c = (((ab) \bmod N)c) \bmod N = (abc) \bmod N = (a((bc) \bmod N)) \bmod N = a * (a * b)$

Identity The identity element of $\langle G, * \rangle$ is 1 because $\forall a \in G_N,\ 1 * a = (1 \cdot a) \bmod N = a \bmod N = a = (a \cdot 1) \bmod N = a * 1$

Inverse The operation $*$ is injective, that is $\forall a, b, c \in G_N$ it holds that $a*b = a*c \implies ab \bmod N = ac \bmod N \implies a(b-c) \bmod N = 0 \implies b = c$. The injective property, together with the fact that the group is finite means that the operation of a particular element a of the group with any element of the group performs a permutation of the elements. Thus $\forall a \in G_N, \exists a^{-1} \in G_N : a * a^{-1} = a^{-1}a = 1$.

□

According to Definition A.1 and because G_N is finite, $\forall a \in G_N$, the order $r = \mathrm{ord}(a)$ is the smallest positive integer r for which it holds $a^r \bmod N = 1$. If $f(x) \doteq a^x \bmod N$, $x \in \mathbb{N}$ is the *modular exponentiation* sequence, then it is easy to see that this sequence is periodic with fundamental period the order r of the element a, because $\forall x \in \mathbb{Z}$

$$\begin{aligned} f(x+r) &= a^{x+r} \bmod N \\ &= a^x a^r \bmod N \\ &= ((a^r \bmod N)a^x) \bmod N \\ &= ((a^r \bmod N)a^x \bmod N) \\ &= (1 \cdot a^x) \bmod N = f(x) \end{aligned} \tag{A.1}$$

Proposition A.2. *Let $r = \mathrm{ord}(a)$ the order of any element a of the multiplicative group G_N defined in Proposition A.1. If r is even and $(a^{\frac{r}{2}}+1) \bmod N \neq 0$ (N does not divide $(a^{\frac{r}{2}}+1)$, then $\gcd(a^{\frac{r}{2}}-1, N) \neq 1$ and $\gcd(a^{\frac{r}{2}}+1, N) \neq 1$. (This means that N has common factors with $a^{\frac{r}{2}}+1$ and $a^{\frac{r}{2}}-1$).*

Proof. N divides $a^r - 1$ because $r = \mathrm{ord}(a) \implies a^r \bmod N = 1 \implies (a^r - 1) \bmod N = 0$. Moreover, $a^r - 1 = (a^{\frac{r}{2}}-1)(a^{\frac{r}{2}}+1)$ and by assumption r is even, thus $(a^{\frac{r}{2}}-1)$ and $(a^{\frac{r}{2}}+1)$ are integers. Consequently, N divides the product $(a^{\frac{r}{2}}-1)(a^{\frac{r}{2}}+1)$ and thus there exists an integer k such that $kN = (a^{\frac{r}{2}}-1)(a^{\frac{r}{2}}+1)$. But N cannot divide $a^{\frac{r}{2}}-1$ as by definition r is the smallest positive integer for which N divides $a^r - 1$. Also, N does not divide $a^{\frac{r}{2}}+1$ by the assumption of the proposition. Proof of the Proposition follows by contradiction.

(i) Suppose that $\gcd(a^{\frac{r}{2}}-1, N) = 1$. Then, by Bézout's identity, there exist integers m and n such that

$$\begin{aligned} &m(a^{\frac{r}{2}}-1) + nN = 1 \implies \\ &m(a^{\frac{r}{2}}-1)(a^{\frac{r}{2}}+1) + nN(a^{\frac{r}{2}}+1) = (a^{\frac{r}{2}}+1) \implies \\ &mkN + nN(a^{\frac{r}{2}}+1) = (a^{\frac{r}{2}}+1) \implies \\ &\left(mk + n(a^{\frac{r}{2}}+1)\right) N = (a^{\frac{r}{2}}+1) \end{aligned}$$

which is impossible because contradicts the assumption that N does not divide $(a^{\frac{r}{2}}+1)$.

(ii) Similarly, suppose that $\gcd(a^{\frac{r}{2}}+1, N) = 1$. Then, there are integers m and n such that

$$\begin{aligned} &m(a^{\frac{r}{2}}+1) + nN = 1 \implies \\ &m(a^{\frac{r}{2}}+1)(a^{\frac{r}{2}}-1) + nN(a^{\frac{r}{2}}-1) = (a^{\frac{r}{2}}-1) \implies \\ &mkN + nN(a^{\frac{r}{2}}-1) = (a^{\frac{r}{2}}-1) \implies \\ &\left(mk + n(a^{\frac{r}{2}}-1)\right) N = (a^{\frac{r}{2}}-1) \end{aligned}$$

which contradicts too the assumption that N does not divide $(a^{\frac{r}{2}}-1)$. □

Corollary A.1. *Let N be a composite number and G_N the multiplicative group of integers modulo N. If the order $r = \mathrm{ord}(a)$ of some $a \in G_N$ is even and $(a^{\frac{r}{2}}+1) \bmod N \neq 0$, then $\gcd(a^{\frac{r}{2}}-1, N)$ is a non-trival factor of N.*

Proof. By Proposition A.2, $\gcd(a^{\frac{r}{2}}-1, N) \neq 1$, thus $\gcd(a^{\frac{r}{2}}-1, N)$ is a factor of N. Moreover $\gcd(a^{\frac{r}{2}}-1, N) \neq N$ because N does not divide $a^{\frac{r}{2}}-1$. □

It is the above Corollary that establishes the reduction of the integer factorization problem to the period finding algorithm of Table 3.1.

Corollary A.2. *Let N a composite number which can be factored into two prime factors p and q and G_N the multiplicative group of integers modulo N. If the order $r = \mathrm{ord}(a)$ of some $a \in G_N$ is even and $(a^{\frac{r}{2}} + 1) \bmod N \neq 0$, then the prime factors p and q are given by*

$$p = \gcd(a^{\frac{r}{2}} + 1, N) \tag{A.2}$$

$$q = \gcd(a^{\frac{r}{2}} - 1, N) \tag{A.3}$$

Proof. By Proposition A.2, N has common factors with $\gcd(a^{\frac{r}{2}} + 1, N)$ and $\gcd(a^{\frac{r}{2}} - 1, N)$. In other words, there are natural numbers $x > 1$ and $y > 1$, such that x divides both $a^{\frac{r}{2}} - 1$ and N, while y divides both $a^{\frac{r}{2}} + 1$ and N, or more compactly $x = \gcd(a^{\frac{r}{2}} - 1, N)$ and $y = \gcd(a^{\frac{r}{2}} + 1, N)$. Because N is a product of two primes p and q, then it is mandatory that $x = p$ and $y = q$. □

Definition A.2. *The Euler (or totient) function $\varphi : \mathbb{N}^* \to \mathbb{N}^*$ is defined so that $\varphi(n)$ is the number of positive integers not greater than n which are co-primes to n.*

From the above definition it is clear that $\varphi(p) = p - 1$ when p is a prime and that the order of the multiplicative group G_N is $|G_N| = \varphi(N - 1)$. Also, for a prime p, $\varphi(p^a) = p^{a-1}(p - 1) = p^a \left(1 - \frac{1}{p}\right)$. By convention, $\varphi(1) = \varphi(2) = 1$.

Theorem A.1. *(Chinese Remainder Theorem). Let $m_1, \ldots, m_n$ positive co-prime integers. The system of n equations $x = a_j \bmod m_j, \quad j = 1, \ldots, n$ has a unique solution in G_M where $M = m_1 m_2 \cdots m_n$.*

Proof. Let $M_j = M/m_j$, then $\gcd(m_j, M_j) = 1$ and thus there exists the inverse N_j of $M_j (\bmod m_j)$, that is $M_j N_j \bmod m_j = 1$. The integer $x \doteq \sum_{j=1}^{n} a_j M_j N_j$ is a solution because $x \bmod m_k = a_k M_k N_k \bmod m_k + \sum_{j \neq k}^{n} a_j M_j N_j \bmod m_k = a_k \cdot 1 + \sum_{j \neq k}^{n} a_j 0$, as long as $M_j N_j \bmod m_k = 0$ for $j \neq k$.

The uniqueness of the solution up to modulo $M = m_1 m_2 \cdots m_n$ follows. If x' is another solution then $\forall j$, $x - x' \pmod{m_j}$, that is m_j divides $x - x'$. Because $gcd(m_j, m_k) = 1, j \neq k$, it is clear that $m_1 m_2 \cdots m_n$ also divides $x - x'$ and consequently $x = x' \bmod M$. □

A direct consequence of the Chinese Remainder theorem is that there is a bijection between the multiplicative group G_M and the Cartesian product of the multiplicative groups $G_{m_1} \times \cdots \times G_{m_n}$. This bijection means that for co-primes $m_1, \ldots, m_n$, it holds $\varphi(m_1 \cdots m_n) = \varphi(m_1) \cdots (m_n)$.

The following two propositions establish the efficient probabilistic reduction of the integer factorization problem to the order finding problem, namely that with arbitrary high probability of success the integer factorization problem can be solved with a constant iterations number of the order finding problem.

Proposition A.3. *Let p be an odd prime and 2^d the largest power of two that divides $\varphi(p^a)$. Then, with probability $1/2$, integer 2^d divides the order r (modulo p^a) of a uniformly random chosen element of the multiplicative group G_{p^a}.*

Proof. When p is odd, $\varphi(p^a) = p^{a-1}(p - 1)$ is even, and thus the largest power of two that divides $\varphi(p^a)$ is a least 2 or $d \geq 1$. The multiplicative group G_{p^a} is cyclic because p^a is a prime power. For this reason, there exists an element g (the generator) of G_{p^a} such that any element x of the group can be written as $x = g^k \bmod p^a$ for some integer k. Let $x = g^k \bmod p^a$ is the random chosen element of G_{p^a} and r its order; $g^r = 1 \pmod{p^a}$.

If k is odd then $x^r = g^{kr} = 1 \pmod{p^a}$. The group G_{p^a} is a cyclic group and its order is $|G_{p^a}| = \varphi(p^a)$, so it holds that $g^{\varphi(p^a)} = 1 \pmod{p^a}$ with p^a the smallest positive integer with this property, and thus $\varphi(p^a)$ divides kr. By the assumption, 2^d divides $\varphi(p^a)$ and consequently 2^d divides kr and thus 2^d divides r because k is odd.

On the other side, if k is even, $\left(g^{\varphi(p^a)}\right)^{k/2} = \left(g^k\right)^{\varphi(p^a)/2} = 1 \pmod{p^a}$. Due to the definition of r, which is the lowest positive integer having the property $\left(g^k\right)^r = 1 \pmod{p^a}$ it holds that r divides $\varphi(p^a)/2$. By assumption, 2^d is the largest power of 2 that divides $\varphi(p^a)$, thus 2^d does not divide $\varphi(p^a)/2$ and consequently 2^d does not divide r.

In conclusion, the elements of the set G_{p^a} can be partitioned into two subsets of equal size, the ones that can be written as g^k with k odd and for which 2^d divides r, and the ones that can be written as g^k with k even and for which 2^d does not divide r. □

Proposition A.4. *Let $N = p_1^{m_1} p_2^{m_2} \cdots p_n^{m_n}$ be the unique factorization of an odd integer N into its prime factors. If x is an element chosen at random and uniformly from the multiplicative group G_N, and r is its order, then it holds*

$$\text{Prob}\left[r \bmod 2 = 0 \quad \text{AND} \quad x^{r/2} + 1 \bmod N \neq 0\right] \geq 1 - \frac{1}{2^{n-1}}$$

Proof. An equivalent relation for the above probability expression is that

$$\text{Prob}\left[r \bmod 2 \neq 0 \quad \text{OR} \quad x^{r/2} + 1 \bmod N = 0\right] \leq \frac{1}{2^{n-1}}$$

The bijection between the groups G_N and $G_{m_1} \times \cdots \times G_{m_n}$ deduced from the Chinese remainder Theorem A.1, permits to randomly choose n-tuples $(x_1, \ldots, x_n)$, from $G_{m_1} \times \cdots \times G_{m_n}$, instead of directly choosing x, and requiring $x = x_j \bmod p_j^{m_j}$. Let r_j be the order of x_j, and 2^{d_j} the largest power of 2 that divides the order r_j and 2^d the largest power of 2 that divides r.

Because r is the order of x, that is $x^r \bmod p_1^{m_1} p_2^{m_2} \cdots p_n^{m_n} = 1$, it holds that $p_1^{m_1} p_2^{m_2} \cdots p_n^{m_n}$ divides $x^r - 1$ and the same holds for each $p_j^{m_j}$ or equivalently, $x^r \bmod p_j^{m_j} = 1$. But, $x = x_j \bmod p_j^{m_j}$ which gives $x^r = x_j^r \bmod p_j^{m_j} = 1$.

If the order r is odd, by taking into account, that r_j is the least positive integer with the property $x_j^{r_j} \bmod p_j^{m_j} = 1$ the conclusion is that r_j divides r and thus r_j is odd too, leading to $d_j = 0$.

The other case occurs when r is even, and thus $r/2$ is an integer. Then $x^{r/2} + 1 \bmod N = 0$ means that $p_1^{m_1} \ldots p_n^{m_n}$ divides $x^{r/2} + 1$ and the same holds for $p_j^{m_j}$, that is $x^{r/2} \bmod p_j^{m_j} = -1$. Since $x = x_j \bmod p_j^{m_j}$, this means also that $x_j^{r/2} \bmod p_j^{m_j} = -1$. But r_j is the smallest positive integer with the property $x_j^{r_j} \bmod p_j^{m_j} = 1$, consequently r_j cannot divide $r/2$. It is already proven that r_j divides r and because 2^{d_j} divides r_j, the power of two 2^{d_j} divides r also. But, bu assumption 2^d is the largest power of two that divides r and consequently $d_j < d$. On the other side $2r_j$ cannot divide r and for this reason $d_j + 1 > d$. In conclusion, in this second case it holds $d_j = d$.

In both cases, r being odd or $x^{r/2} + 1 \bmod N = 0$, the largest power of two 2^{d_j} that divides the orders r_j of the randomly chosen $x_j \bmod p_j^{m_j}$ is the same for all j. This power of two 2^{d_j} divides also $\varphi(p_j^{m_j})$. By Proposition A.3, the largest power of two that divides $\varphi(p_j^{m_j})$ divides the order of any randomly chosen element of $G_{p_j^{m_j}}$ with probability $1/2$. Thus, the probability that 2^{d_j} divides the order of any randomly chosen element of some $G_{p_j^{m_j}}$ is at most $1/2$ and the probability that 2^{d_j} jointly divides the order of any randomly chosen element of all the

$G_{p_j^{m_j}}$ is at most $1/2^n$, except when $n = 1$. In this case, where N is a prime power, the probability is 1; it can never hold that r is even and $x^{r/2} + 1 \bmod N \neq 0$. The case of N being a prime power can be excluded from the analysis of Shor's algorithm because it can be handled by a classical algorithm to efficiently factorize N. To include this case the proposition is formulated with the fraction $1/2^{n-1}$ instead of $1/2^n$. □

A.2 Continued Fraction Expansion

Definition A.3. *The finite continued fraction expansion (CFE) defined by the positive integers sequence $c_0, c_1, \ldots, c_R$ is the rational number*

$$[c_0, c_1, \ldots, c_R] \doteq c_0 + \cfrac{1}{c_1 + \cfrac{1}{c_2 + \cfrac{1}{\cdots + \cfrac{1}{c_R}}}}$$

The j-th convergent ($j = 0, \ldots, R$) to the above continued fraction is the rational number $[c_0, c_1, \ldots, c_j]$

Proposition A.5. *Given a rational $\xi = \frac{p}{q}$, its continued fraction expansion representation $[c_0, c_1, \ldots, c_R]$ can be computed by the following algorithm which terminates after $O(\log p)$ iterations.*

$c_0 = \lfloor \xi \rfloor$
$\xi_0 = \xi - c_0$
$j = 0$
WHILE $\xi_j \neq 0$ DO
  $c_{j+1} = \lfloor \frac{1}{\xi_j} \rfloor$
  $\xi_{j+1} = \frac{1}{\xi_j} - c_{j+1}$
  $i = i + 1$
END WHILE

Proof. By the definition of the CFE, it is clear that c_0 is the quotient q_0 of the division p/q, while the other term, $\frac{1}{c_1 + \ldots}$, is $\frac{1}{q/r_0}$, where r_0 is the remainder of the division p/q and $\xi_0 = r_0/q$. Proceeding the same way, c_1 is the quotient of $q/r_0 = 1/\xi_0$ while the other term $\frac{1}{c_2 + \ldots}$, is $\frac{1}{r_0/r_1}$, where r_1 is the remainder of the division $q/r_0 = 1/\xi_0$. Thus, the coefficients c_j are essentially the partial quotients of the Euclid algorithm initialized with p and q. The algorithm of Proposition A.5 is the Euclidean algorithm. The time complexity of Euclid algorithm is known to be $O(\log p)$. □

The following proposition permits to retrieve all the convergents of a rational ξ from its CFE representation, in the form $\xi_j = \frac{p_j}{q_j}$ where $\gcd(p_j, q_j) = 1$ (numerator and denominator are coprimes).

Proposition A.6. *Each convergent ξ_j of $\xi = [c_0, c_1, \ldots, c_n]$ can be written in the form $\frac{p_j}{q_j}$, where p_j and q_j are coprimes, and the integers p_j and q_j can be found be the following recursive relation:*

$p_0 = c_0$
$q_0 = 1$
$p_1 = c_1 c_0 + 1$
$q_1 = a_1$
FOR $j = 2$ TO n DO
  $p_j = c_j p_{j-1} + p_{j-2}$
  $q_j = c_j q_{j-1} + q_{j-2}$
END FOR

Proof. It is clear that the above algorithm is valid for $n = 0, 1, 2$. Using the compact form of the CFE representation it can be stated that

$$[c_0, c_1, \ldots, c_n] = [c_0, c_1, \ldots, c_{n-2}, c_{n-1} + 1/c_n] \tag{A.4}$$

The left hand side is a CFE of n terms while the right hand side is a CFE of $n-1$ terms. Assuming that the algorithm is valid for $n-1$, it will be proven that it is valid for n by induction. Let $p_j^{'}/q_j^{'}$ be the sequence of convergents of the right hand side for $j = 0 \ldots n-1$ as given be the algorithm. It will be proven that for the sequence of the respective convergents of the left hand side p_j/q_j, it holds that $p_n/q_n = p_{n-1}^{'}/q_{n-1}^{'}$.

The terms of the two CFE forms of Eq. (A.4) have equal coefficients from c_0 up to c_{n-2}. Consequently, taking into account the algorithm the following relations are true

$$\begin{aligned} \frac{p_{n-1}^{'}}{p_{n-1}^{'}} &= \frac{(c_{n-1} + 1/c_n)p_{n-2} + p_{n-3}}{(c_{n-1} + 1/c_n)q_{n-2} + q_{n-3}} \\ &= \frac{p_{n-1} + p_{n-2}/c_n}{q_{n-1} + q_{n-2}/c_n} \\ &= \frac{c_n p_{n-1} + p_{n-2}}{c_n q_{n-1} + q_{n-2}} \\ &= \frac{p_n}{q_n} \end{aligned} \tag{A.5}$$

This means that $\frac{p_n}{q_n} = [c_0, c_1, \ldots, c_n]$ □

Theorem A.2. *Let $\xi \in \mathbb{R}$ and $a, b \in \mathbb{Z}$ with $b > 0$. The rational $\frac{a}{b}$ is a convergent of the CFE of ξ if the following condition holds*

$$\left| \xi - \frac{a}{b} \right| \leq \frac{1}{2b^2}$$

See [182] for a proof.

Corollary A.3. *If a measurement result k is obtained by the Shor's algorithm circuit of Figure 3.5 with $L = 2^{2n}$ and order r of a random a, and additionally this measurement result k is the closest to an integer multiple of L/r, that is $\left|k - m\frac{L}{r}\right| \leq \frac{1}{2}$, $m \in \{0, \ldots, r-1\}$, then the rational number $\frac{m}{r}$ is a convergent of k/L.*

Proof. Taking into account that $N^2 \leq L$ and $r \leq N$ then

$$\left|k - m\frac{L}{r}\right| \leq \frac{1}{2} \implies \left|\frac{k}{L} - \frac{m}{r}\right| \leq \frac{1}{2L} \leq \frac{1}{2N^2} \leq \frac{1}{2r^2}$$

Thus, the condition of Theorem A.2 is satisfied for $\xi = \frac{k}{L}$. □

The usage of $2n$ qubits at the top register of 3.5, where $n = \lceil N \rceil$, is justified by the required condition of Theorem A.2. Corollary A.3 permits the extraction of a rational m/r, where r is the period, given a measurement result k on the condition that this result is the closest to a multiple of L/r. This extraction is done in two steps. First, the convergents of L/r are found using the algorithm of Proposition A.5. These convergents are of the form $c_0, c_1, \ldots, c_j$. Then, these convergents are calculated in the equivalent form p_j/q_j using the algorithm of Proposition A.6. The numerator p_j and the denominator q_j extracted by this procedure are co-primes ($\gcd(p_j, q_j)$). By Corollary A.3, one of these convergents is m/r and thus for $r = q_j$ to be valid, it must hold that $\gcd(m, r)$. The candidate periods q_j are verified by using $a^{q_j} \bmod N = 1$. If this verification fails for all q_j then the whole quantum computation is repeated as shown in the flowchart of Figure 3.7.

A.3 Success Probability of Quantum Period Finding

The quantum period finding algorithm successfully finds the period r searched for if two conditions are met: (i) The measurement result k at the last step of quantum computation is the closest to an integer m multiple of L/r, and (ii) the integers m and r are co-primes. The probability of success is analyzed in this section.

Definition A.4. *The set $\mathcal{K}_g$ of "good" measurements is the set of the measurement indices k that are the closest ones to the integers multiples of $\frac{L}{r}$*

$$\mathcal{K}_g \doteq \left\{k = 0 \ldots L-1 : \left|k - m\frac{L}{r}\right| \leq \frac{1}{2},\, m = 0 \ldots r-1\right\}$$

Because the integer m ranges between 0 and $r-1$, its range can be reformulated as $m = \text{round}(\frac{k}{L}r)$ for $k = 0 \ldots L-1$. By defining the residue $\{rk\}_L \doteq rk - L\text{round}(\frac{rk}{L})$, the definition of the set $\mathcal{K}_g$ can be reformulated as follows

$$\begin{aligned}\mathcal{K}_g = &\left\{k = 0 \ldots L-1 : |rk - mL| \leq \frac{r}{2},\, m = 0 \ldots r-1\right\} = \\ &\left\{k = 0 \ldots L-1 : \left|rk - \text{round}(\frac{k}{L}r)L\right| \leq \frac{r}{2}\right\} = \\ &\left\{k = 0 \ldots L-1 : |\{rk\}_L| \leq \frac{r}{2}\right\}\end{aligned} \tag{A.6}$$

The following Proposition gives a lower bound for the probability to measure a result that belongs to the "good" set of measurements $\mathcal{K}_g$.

Proposition A.7. *The probability to obtain a measurement k which belongs to the set of "good" measurements $\mathcal{K}_g$ is lower bounded as follows*

$$\text{Prob}\left[k | k \in \mathcal{K}_g\right] \geq \begin{cases} \frac{4}{\pi^2 r}\left(1 - \frac{1}{L}\right)^2, & 0 < \{rk\}_L \leq \frac{r}{2}\left(1 - \frac{1}{L}\right) \\ \frac{1}{r}, & \{rk\}_L = 0 \end{cases} \tag{A.7}$$

Proof. The relations $N^2 \leq L \leq 2N^2$ and $0 \leq r < N$ are used to derive the following inequalities, where $s = \lfloor L/r \rfloor$,

$$\begin{aligned}\left|\frac{\pi\{rk\}_L}{L}(s+1)\right| &\leq \frac{\pi r}{2L}\left(1-\frac{1}{L}\right)(s+1)\\ &\leq \frac{\pi}{2}\left(1-\frac{1}{L}\right)\left(\frac{L+r}{L}\right)\\ &\leq \frac{\pi}{2}\left(1-\frac{1}{L}\right)\left(1+\frac{L}{L^2}\right)\\ &\leq \frac{\pi}{2}\end{aligned} \tag{A.8}$$

A direct consequence is that

$$\left|\frac{\pi\{rk\}_L}{L}s\right| \leq \frac{\pi}{2} \tag{A.9}$$

Also, because $rk = \{rk\}_L + L\mathrm{round}(\frac{rk}{L})$ the following equality is valid

$$\sin\left(\frac{\pi rk}{L}x\right) = \sin\left(\frac{\pi\{rk\}_L}{L}x\right),\ \forall x \in \mathbb{Z} \tag{A.10}$$

Using the trigonometric inequality and Eq. (3.21)

$$\frac{4}{\pi^2}\theta^2 \leq \sin^2(\theta) \leq \theta^2, \quad |\theta| < \frac{\pi}{2} \tag{A.11}$$

we conclude that if $0 < \{rk\}_L \leq \frac{r}{2}\left(1-\frac{1}{L}\right)$ then (we remind that $L = rs + t$)

$$\begin{aligned}\mathrm{Prob}[k] &= \frac{t\sin^2\left(\frac{\pi\{rk\}_L}{L}(s+1)\right) + (r-t)\sin^2\left(\frac{\pi\{rk\}_L}{L}s\right)}{L^2\sin^2\left(\frac{\pi\{rk\}_L}{L}\right)}\\ &\geq \frac{t\frac{4}{\pi^2}\left(\frac{\pi\{rk\}_L}{L}(s+1)\right)^2 + (r-t)\frac{4}{\pi^2}\left(\frac{\pi\{rk\}_L}{L}s\right)^2}{L^2\left(\frac{\pi\{rk\}_L}{L}\right)^2}\\ &\geq \frac{4}{\pi^2}\cdot\frac{rs^2}{L^2}\\ &= \frac{4}{\pi^2}\cdot\frac{1}{r}\left(\frac{L-t}{L}\right)^2\\ &= \frac{4}{\pi^2}\cdot\frac{1}{r}\left(1-\frac{t}{L}\right)^2\\ &\geq \frac{4}{\pi^2}\cdot\frac{1}{r}\left(1-\frac{1}{N}\right)^2\end{aligned} \tag{A.12}$$

Thus, first case of Proposition A.7 has been proved. The second case $\{rk\}_L = 0$ can be easily proven by taking into account that $0 \leq t \leq r-1$ and that the probability of Eq. (3.21) becomes

$$\begin{aligned} \text{Prob}\left[k\right] &= \frac{tL^2 + (r-t)\,r^2 s^2}{L^2 r^2} \\ &= \frac{1}{r} \cdot \frac{L^2 + t(r-t)}{L^2} \\ &\geq \frac{1}{r} \end{aligned} \tag{A.13}$$

□

The number of indices k that fulfill the condition $|\{rk\}_L| \leq \frac{r}{2}$ is r, as implied by Eq. (A.6). Combined with the fact that the number N to be factored is very large and thus $\left(1 - \frac{1}{L}\right)^2 \approx 1$ the total probability to obtain a "good" measurement is lower bounded by

$$\text{Prob}\left[k \in \mathcal{K}_g\right] \geq \frac{4}{\pi^2} \approx 0.4 \tag{A.14}$$

In conclusion, the total probability to obtain a measurement which is the closest to an integer multiple of $\frac{L}{r}$, something that could permit the extraction of the period r using CFE, is at least 40%.

It remains to calculate the final probability that the CFE procedure correctly gives the period r on the condition that a "good" measurement is obtained, that is the probability

$$\text{Prob}\left[k \in \mathcal{K}_g : \gcd(m, r) = 1\right],$$

where m is a function of index k as implied in Definition A.4, namely $m = A(k) = \text{round}(\frac{k}{L}r)$.

Because the number of the "good" measurements is r there is a "one-to-one" relation between the "good" k's and $m = A(k)$. This fact can be combined with the definition of the Euler's function $\varphi(n)$ which gives the number of natural numbers which are co-primes and less than n. Thus, the probability to obtain a "good" measurement and at the same time this measurement lead to a correct period estimation is given by

$$\text{Prob}\left[k \in \mathcal{K}_g : \gcd(A(k), r) = 1\right] \geq \frac{4}{\pi^2} \cdot \frac{\varphi(r)}{r} \left(1 - \frac{1}{N}\right)^2 \tag{A.15}$$

The following result from Number Theory is used, without proof.

Theorem A.3.

$$\liminf \frac{\varphi(N)}{N/\ln\ln N} = e^{-\gamma} \tag{A.16}$$

where γ is Euler's constant and it is $\gamma = 0.5777215\ldots$ and $e^{-\gamma} = 0.5614594\ldots$

Corollary A.4.

$$\text{Prob}\left[k \in \mathcal{K}_g : \gcd(A(k), r) = 1\right] \geq \frac{4}{\pi^2 \ln 2} \cdot \frac{e^{-\gamma} - \varepsilon(r)}{\ln\log N} \left(1 - \frac{1}{N}\right)^2 \tag{A.17}$$

where $\varepsilon(r)$ is a monotonically decreasing sequence converging to zero.

Proof. Equivalent to Theorem A.3 is the result that $\frac{\varphi(r)}{r/\ln\ln r} = e^{-\gamma} - \varepsilon(r)$, where $\varepsilon(r)$ is a monotonically decreasing sequence converging to zero. consequently,

$$\frac{\varphi(r)}{r} \geq \frac{e^{-\gamma}-\varepsilon(r)}{\ln\ln r} \geq \frac{e^{-\gamma}-\varepsilon(r)}{\ln\ln N} = \frac{e^{-\gamma}-\varepsilon(r)}{\ln\ln 2 + \ln\log N} \geq \frac{e^{-\gamma}-\varepsilon(r)}{\ln 2}\cdot\frac{1}{\log\log N} \tag{A.18}$$

□

Corollary A.4 can be stated in the equivalent form

$$\text{Prob}\left[k \in \mathcal{K}_g : \gcd(A(k), r) = 1\right] = \Omega\left(\frac{1}{\log\log N}\right) \tag{A.19}$$

In conclusion, the Quantum Period Finding algorithm gives the period r with a constant lower bounded probability of error using $O(\log\log N)$ iterations, where N is the number to be factored, or equivalently, using $O(\log n)$ iterations, where n is the number of bits of N.

A.4 Quantum Fourier Transform Circuit

The derivation of Eq. (3.5) is justified by the following relations

$$\begin{aligned}
|j\rangle = |j_1 \ldots j_n\rangle \xrightarrow{QFT_{2^n}} & \frac{1}{\sqrt{2^n}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} e^{\frac{i2\pi}{2^n} j \sum_{l=1}^{n} k_l 2^{n-l}} |k_1 \ldots k_n\rangle \\
= & \frac{1}{\sqrt{2^n}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{i2\pi j k_l 2^{-l}} |k_l\rangle \\
= & \frac{1}{\sqrt{2^n}} \bigotimes_{l=1}^{n} \sum_{k_l=0}^{1} e^{i2\pi j k_l 2^{-l}} |k_l\rangle \\
= \frac{1}{\sqrt{2^n}} \left(|0\rangle + e^{i2\pi(0.j_n)}|1\rangle\right) & \left(|0\rangle + e^{i2\pi(0.j_{n-1}j_n)}|1\rangle\right) \cdots \left(|0\rangle + e^{i2\pi(0.j_1 j_2 \ldots j_{n-1} j_n)}|1\rangle\right)
\end{aligned} \tag{A.20}$$

The first line in the above set of equations follows from the QFT definition of Eq. (3.1) and the fact that $\sum_{l=1}^{n} k_l 2^{n-l} = k$, assuming that $k = (k_1 k_2 \ldots k_n)$ in binary notation. The second line is just a rewriting of $|k\rangle = |k_1 k_2 \ldots k_n\rangle = |k_1\rangle \otimes |k_2\rangle \otimes \cdots \otimes |k_n\rangle$. The third line follows from the linearity property of the tensor product in Eq. (2.13). The last line is just the expanded form of the third line and shows that in the particular QFT action on computational basis states the resulting state is of product form.

The first factor of the product form $\frac{1}{\sqrt{2}}\left(|0\rangle + e^{i2\pi(0.j_n)}|1\rangle\right)$ is a reformulation of the expression $\frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^{j_n}|1\rangle\right)$. Taking into account that $|j\rangle$ belongs to the computational basis and thus $j_n = 0$ or $j_n = 1$, this reformulation gives the state $\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)$ if $|j_n\rangle = |0\rangle$ and the state $\frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right)$ if $|j_n\rangle = |1\rangle$. This is exactly the effect of a Hadamard gate applied on $|j_n\rangle$, as exposed in Eq. (2.25). This operation is depicted on the lower qubit of Figure 3.1.

The second factor of the product form is $\frac{1}{\sqrt{2}}\left(|0\rangle + e^{i2\pi(0.j_{n-1}j_n)}|1\rangle\right)$. The phase factor $e^{i2\pi(0.j_{n-1}j_n)}$ can be decomposed as $e^{i2\pi(0.j_{n-1})+i2\pi(0.0j_n)} = e^{i2\pi(0.j_{n-1})}e^{i2\pi(0.0j_n)}$. Similarly to the previous analysis, the state $\frac{1}{\sqrt{2}}\left(|0\rangle + e^{i2\pi(0.j_{n-1})}|1\rangle\right)$ can be constructed by applying a Hadamard gate on qubit $|j_{n-1}\rangle$. The additional phase factor $e^{i2\pi(0.0j_n)}$ can be introduced by a $c\text{-}R_z(\frac{\pi}{2})$ controlled by the state $|j_n\rangle$ because $2\pi(0.0j_n) = \frac{\pi}{2}$ if $j_n = 1$, otherwise $2\pi(0.0j_n) = 0$. The consecutive

application of the H and c-$R_z(\frac{\pi}{2})$ gates is depicted in the second qubit line of Figure 3.1 which is initially in state $|j_{n-1}\rangle$.

Proceeding with similar analysis for the remaining factors of the product state in Eq. (A.20), it can be shown that the operation of the circuit in Figure 3.1 performs the desired QFT on the computational basis states $|j\rangle, j = 0 \ldots 2^{n-1}$. Linearity of the operations and the completeness of the computational basis assures its validity for any state $|\psi\rangle \in \mathcal{H}^N$ as described by Eq. (3.1).

The total number of gates of the QFT circuit is $\frac{n(n+1)}{2}$, while its apparent depth, assuming a sequential operation of one gate after the other, is the same. Yet, it can be shown that by a suitable rearrangement, the gates involved in operations of different qubits can operate in parallel, lowering the depth of the QFT circuit to only $2n-1$ steps.

A.5 Quantum Phase Estimation

The decomposition of the U_f operator in a sequence of $CU_{a^{2^j}}$ operators shown in Figure 3.8 offers an alternative interpretation of the quantum period finding algorithm as a phase estimation algorithm, which can give more insight into other relevant algorithms. The $CU_{a^{2^j}}$ controlled operator can be decomposed in block form as

$$CU_{a^{2^j}} = \begin{bmatrix} I & 0 \\ 0 & U_{a^{2^j}} \end{bmatrix} \tag{A.21}$$

where $U_{a^{2^j}}$ is the operator defined by $U_{a^{2^j}}(|x\rangle) = |a^{2^i} x \bmod N\rangle$. Equation (3.26) can be reformulated as

$$CU_{a^{2^j}}\left(|c\rangle|y\rangle\right) = |c\rangle \left(U_{a^{2^j}}\right)^c |y\rangle \tag{A.22}$$

Observe that $U_{a^{2^j}} = (U_a)^{2^j}$ where U_a is the modular multiplication operator defined by $U_a(|x\rangle) = |ax \bmod N\rangle$. Let $|u\rangle$ be an eigenstate of U_a. The corresponding eigenvalue is of the form $e^{i\varphi}$, where $0 \leq \varphi < 2\pi$, because U_a is unitary (a is selected to be co-prime with N) and thus $U_{a^{2^j}}|u\rangle = (U_a)^{2^j}|u\rangle = e^{i\varphi 2^j}|u\rangle$. In Figure 3.8 the controlled modular multipliers $CU_{a^{2^j}}$ are controlled by the j-th qubit of the top register. If the top register is in the computational basis state $|k\rangle = |k_{2n-1} \ldots k_0\rangle$ and the bottom register is in an eigenstate $|u\rangle$ of U_a then by using Eq. (A.22) we conclude that the state of both registers after the application of the $2n$ controlled modular multipliers will be

$$\begin{aligned} U_f(|k\rangle|u\rangle) =& |k\rangle \prod_{j=0}^{2n-1} \left(U_{a^{2^j}}\right)^{k_j} |u\rangle \\ =& |k\rangle \prod_{j=0}^{2n-1} \left(e^{i\varphi 2^j k_j}\right) |u\rangle \\ =& |k\rangle e^{i\varphi \sum_{j=0}^{2n-1} 2^j k_j} |u\rangle \\ =& |k\rangle e^{i\varphi k} |u\rangle \end{aligned} \tag{A.23}$$

The phase factor $e^{i\varphi k}$ can be viewed as being kicked-back to the top register as already noted in section 2.4.2.

The top register in Figure 3.8 is initialized in an equiprobable superposition $\frac{1}{\sqrt{Q}} \sum_{k=0}^{Q-1} |k\rangle$

and thus on the condition that the bottom register is initially set in an eigenstate $|u\rangle$, the state of the top register just before the inverse QFT would be

$$U_f\left(\frac{1}{\sqrt{Q}}\sum_{k=0}^{Q-1}|k\rangle|u\rangle\right)=\frac{1}{\sqrt{Q}}\sum_{k=0}^{Q-1}e^{i\varphi k}|k\rangle|u\rangle \tag{A.24}$$

If the phase φ can be described by exactly $2n$ bits in binary fractional notation as $\varphi = 2\pi(0.j_{2n_1}\ldots j_0) = 2\pi j/Q$ (which means that $2\pi\varphi$ divides Q), then Eq. (A.24) takes the form

$$U_f\left(\frac{1}{\sqrt{Q}}\sum_{k=0}^{Q-1}|k\rangle|u\rangle\right)=\frac{1}{\sqrt{Q}}\sum_{k=0}^{Q-1}e^{\frac{i2\pi}{Q}jk}|k\rangle|u\rangle \tag{A.25}$$

The application of the inverse QFT at the top register then, is by definition exactly $|j\rangle$. Thus the circuit of Figure 3.8 can find the phase $\varphi = 2\pi j/Q$ if the bottom register is initialized in an eigenstate of U_a.

The initialization of the bottom register in the quantum period finding application is the state $|1\rangle$. For this particular application it holds that $U_a^r = 1$ where r is the order of a searched for, because $U_a^r|x\rangle = |xa^r \bmod N = |x\rangle$. Consequently, the r-th roots of unity $e^{i2\pi t/r} = e^{i\varphi_t}$ for $t = 0\ldots r-1$ are eigenvalues of U_a. Furthermore, it can be proven that the eigenstates $|u_t\rangle$ which correspond to these eigenvalues are

$$|u_t\rangle=\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}e^{-i2\pi\frac{t}{r}s}|a^s \bmod N\rangle \tag{A.26}$$

and that the bottom register's initial state $|1\rangle$ is the sum of the eigenstates u_t which correspond to these eigenvalues:

$$|1\rangle=\sum_{t=0}^{r-1}|u_t\rangle \tag{A.27}$$

In this particular case, Eq. (A.24) becomes

$$\begin{aligned}U_f\left(\frac{1}{\sqrt{Q}}\sum_{k=0}^{Q-1}|k\rangle|1\rangle\right)&=\frac{1}{\sqrt{r}}\frac{1}{\sqrt{Q}}\sum_{t=0}^{r-1}\sum_{k=0}^{Q-1}e^{i\varphi_t k}|k\rangle|u_t\rangle\\&=\frac{1}{\sqrt{r}}\frac{1}{\sqrt{Q}}\sum_{t=0}^{r-1}\sum_{k=0}^{Q-1}e^{\frac{i2\pi}{Q}\frac{Q}{r}tk}|k\rangle|u_t\rangle\end{aligned} \tag{A.28}$$

When the order r divides Q, then $j = Q/r$ is an integer (meaning that $1/r$ can be written in binary fractional form of $2n$ bits as $(0.j_{2n-1}\ldots j_0)$). The application of the inverse QFT at the top register then leads to the joint state

$$|\psi\rangle=\frac{1}{\sqrt{r}}\sum_{t=0}^{r-1}e^{\frac{i2\pi}{Q}\left|\frac{Q}{r}t\right\rangle}|u_t\rangle \tag{A.29}$$

The final measurement of the top register gives with certainty one of the values $\frac{Q}{r}t$ for $t = 0 \ldots r-1$ which are related to the eigenvalues phases with the relation $\varphi_t = \frac{2\pi}{Q}\frac{Q}{r}t$. When r does not divide Q, then close estimations of φ_t are performed as already pointed out in section 3.4.

Thus the circuit of Figure 3.8 can be viewed as a quantum phase estimator of the phases φ_t corresponding to the eigenvalues $e^{i\varphi_t}$ of the modular multiplier U_a.

REFERENCES

[1] H. Araki and E. H. Lieb, “Entropy inequalities,” *Communications in Mathematical Physics*, vol. 18, no. 2, pp. 160–170, 1970.

[2] E. H. Lieb and M. B. Ruskai, “Proof of the strong subadditivity of quantum-mechanical entropy,” *Journal of Mathematical Physics*, vol. 14, no. 12, pp. 1938–1941, 1973.

[3] G. Lindblad, “Completely positive maps and entropy inequalities,” *Communications in Mathematical Physics*, vol. 40, no. 2, pp. 147–151, 1975.

[4] A. S. Kholevo, “Bounds for the quantity of information transmitted by a quantum communication channel,” *Problemy Peredachi Informatsii (Problems of Information Transmission)*, vol. 9, no. 3, pp. 3–11, 1973.

[5] A. Kholevo, “On the capacity of a quantum communication channel.” *Problemy Peredachi Informatsii (Problems of Information Transmission)*, vol. 15, no. 4, pp. 3–11, 1979.

[6] J. D. Bekenstein, “Universal upper bound on the entropy-to-energy ratio for bounded systems,” *Physical Review D*, vol. 23, pp. 287–298, Jan. 1981.

[7] D. Deutsch, “Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer,” *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 400, no. 1818, pp. 97–117, Jul. 1985.

[8] D. Deutsch, “Quantum Computational Networks,” *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 425, no. 1868, pp. 73–90, Sep. 1989.

[9] D. Simon, “On the power of quantum computation,” in *Proc. 35th Annual IEEE Symposium on Foundations of Computer Science, (FOCS’94)*, Nov. 1994, pp. 116–123.

[10] P. W. Shor, “Algorithms for Quantum Computation: Discrete Logarithms and Factoring,” in *Proc. 35th Annual IEEE Symposium on Foundations of Computer Science, (FOCS’94)*, Nov. 1994, pp. 124–134.

[11] L. K. Grover, “A Fast Quantum Mechanical Algorithm for Database Search,” in *Proc. 28th Annual ACM Symposium on Theory of Computing (STOC’96)*, May 1996, pp. 212–219.

[12] S. Lloyd, “Universal Quantum Simulators,” *Science*, vol. 273, no. 5278, pp. 1073–1078, Aug. 1996.

[13] J. I. Cirac and P. Zoller, “Quantum Computations with Cold Trapped Ions,” *Physical Review Letters*, vol. 74, pp. 4091–4094, May 1995.

[14] D. Vion, A. Aassime, A. Cottet, P. Joyez, H. Pothier, C. Urbina, D. Esteve, and M. H. Devoret, “Manipulating the Quantum State of an Electrical Circuit,” *Science*, vol. 296, no. 5569, pp. 886–889, May 2002.

[15] T. Monz, D. Nigg, E. A. Martinez, M. F. Brandl, P. Schindler, R. Rines, S. X. Wang, I. L. Chuang, and R. Blatt, “Realization of a scalable Shor algorithm,” *Science*, vol. 351, no. 6277, pp. 1068–1070, Mar. 2016.

[16] R. Barends, L. Lamata, J. Kelly, L. García-Álvarez, A. G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, J. Y. Mutus, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, I.-C. Hoi, C. Neill, P. J. J. O'Malley, C. Quintana, P. Roushan, A. Vainsencher, J. Wenner, E. Solano, and J. M. Martinis, "Digital quantum simulation of fermionic models with a superconducting circuit," *Nature Communications*, vol. 6, pp. 7654:1–7654:7, Jul. 2015.

[17] A. Politi, J. C. F. Matthews, and J. L. O'Brien, "Shor's quantum factoring algorithm on a photonic chip," *Science*, vol. 325, no. 5945, pp. 1221–1221, Sep. 2009.

[18] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang, "Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance," *Nature*, vol. 414, no. 6866, pp. 883–887, Dec. 2001.

[19] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Physical Review A*, vol. 52, pp. R2493–R2496, Oct. 1995.

[20] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Physical Review A*, vol. 54, pp. 1098–1105, Aug. 1996.

[21] A. Steane, "Multiple-Particle Interference and Quantum Error Correction," *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 452, no. 1954, pp. 2551–2577, Nov. 1996.

[22] D. Aharonov and M. Ben-Or, "Fault-tolerant Quantum Computation with Constant Error," in *Proc. 29th Annual ACM Symposium on Theory of Computing (STOC'97)*, May 1997, pp. 176–188.

[23] T. G. Draper, "Addition on a Quantum Computer," *eprint arXiv:quant-ph/0008033*, Aug. 2000.

[24] S. Beauregard, "Circuit for Shor's Algorithm Using $2n + 3$ Qubits," *Quantum Information & Computation*, vol. 3, no. 2, pp. 175–185, Mar. 2003.

[25] A. Khosropour, H. Aghababa, and B. Forouzandeh, "Quantum Division Circuit Based on Restoring Division Algorithm," in *Proc. 8th International Conference on Information Technology: New Generations (ITNG '11)*, 2011, pp. 1037–1040.

[26] B.-S. Choi and R. Van Meter, "On the Effect of Quantum Interaction Distance on Quantum Addition Circuits," *ACM J. Emerging Technologies in Computing Systems*, vol. 7, no. 3, pp. 11:1–11:17, Aug. 2011.

[27] G. Cybenko, "Reducing Quantum Computations to Elementary Unitary Operations," *J. Computing in Science and Engineering*, vol. 3, no. 2, pp. 27–32, Mar. 1996.

[28] V. V. Shende, S. S. Bullock, and I. L. Markov, "Synthesis of quantum-logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 6, pp. 1000–1010, Jun. 2006.

[29] T. Toffoli, "Reversible computing," Massachusetts Institute of Technology, Laboratory for Computer Science, Tech. Rep. MIT/LCS/TM-151, Feb. 1980.

[30] M. Saeedi and I. L. Markov, "Synthesis and Optimization of Reversible Circuits - A Survey," *ACM Computing Surveys*, vol. 45, no. 2, pp. 21:1–21:34, Feb. 2013.

[31] C. H. Bennett, “Logical Reversibility of Computation,” *IBM J. Research and Development*, vol. 17, no. 6, pp. 525–532, Nov. 1973.

[32] G. E. Moore, “Cramming more components onto integrated circuits,” *Electronics*, vol. 38, no. 8, pp. 114–117, Apr. 1965.

[33] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc, “Design of ion-implanted MOSFET’s with very small physical dimensions,” *IEEE Journal of Solid-State Circuits*, vol. 9, no. 5, pp. 256–268, Oct. 1974.

[34] M. Sipser, *Introduction to the Theory of Computation*, 2nd ed. Thomson Course Technology, 2006.

[35] C. H. Papadimitriou, *Computational Complexity*, 1st ed. Addison-Wesley, 1994.

[36] E. Bernstein and U. Vazirani, “Quantum Complexity Theory,” in *Proc. 25th Annual ACM Symposium on Theory of Computing (STOC’93)*, May 1993, pp. 11–20.

[37] J. E. Savage, “Computational Work and Time on Finite Machines,” *J. ACM*, vol. 19, no. 4, pp. 660–674, Oct. 1972.

[38] C. P. Schnorr, “The Network Complexity and the Turing Machine Complexity of Finite Functions,” *Acta Informatica*, vol. 7, no. 1, pp. 95–107, Mar. 1976.

[39] N. Pippenger and M. J. Fischer, “Relations Among Complexity Measures,” *J. ACM*, vol. 26, no. 2, pp. 361–381, Apr. 1979.

[40] R. Landauer, “Irreversibility and Heat Generation in the Computing Process,” *IBM J. Research and Development*, vol. 5, no. 3, pp. 183–191, Jul. 1961.

[41] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th ed. Oxford University Press, 2011.

[42] S. Kotiyal, H. Thapliyal, and N. Ranganathan, “Mach-Zehnder Interferometer Based Design of All Optical Reversible Binary Adder,” in *Proc. Conference on Design, Automation and Test in Europe 2012 (DATE’12)*, Mar. 2012, pp. 721–726.

[43] D. H. Wood and J. Chen, “Fredkin gate circuits via recombination enzymes,” in *Proc. Congress on Evolutionary Computation 2004 (CEC2004)*, Jun. 2004, pp. 1896–2000.

[44] R. P. Feynman, “Simulating physics with computers,” *International Journal of Theoretical Physics*, vol. 21, no. 6, pp. 467–488, Jun. 1982.

[45] P. Kaye, R. Laflamme, and M. Mosca, *An Introduction to Quantum Computing*, 1st ed. Oxford University Press, 2007.

[46] G. Esposito, G. Marmo, G. Miele, and G. Sudarshan, *Advanced concepts in quantum mechanics*, 1st ed. Cambridge University Press, 2014.

[47] A. Einstein, B. Podolsky, and N. Rosen, “Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?” *Physical Review*, vol. 47, pp. 777–780, May 1935.

[48] A. K. Ekert, “Quantum cryptography based on Bell’s theorem,” *Physical Review Letters*, vol. 67, pp. 661–663, Aug. 1991.

[49] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, “Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels,” *Physical Review Letters*, vol. 70, pp. 1895–1899, Mar. 1993.

[50] F. Gaitan, *Quantum Error Correction and Fault Tolerant Quantum Computing*, 1st ed. CRC Press, 2008.

[51] A. C.-C. Yao, “Quantum circuit complexity,” in *Proc. 34th Annual IEEE Symposium on Foundations of Computer Science (FOCS’93)*, Nov. 1993, pp. 352–361.

[52] D. P. DiVincenzo, “Two-bit gates are universal for quantum computation,” *Physical Review A*, vol. 51, pp. 1015–1022, Feb. 1995.

[53] 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, pp. 3457–3467, Nov. 1995.

[54] S. Lloyd, “Almost Any Quantum Logic Gate is Universal,” *Physical Review Letters*, vol. 75, pp. 346–349, Jul. 1995.

[55] D. Deutsch, A. Barenco, and A. Ekert, “Universality in Quantum Computation,” *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 449, no. 1937, pp. 669–677, Jun. 1995.

[56] Kitaev, A. Y. and Shen, A. H. and Vyalyi, M. N., *Classical and Quantum Computation*. American Mathematical Society, 2002.

[57] C. M. Dawson and M. A. Nielsen, “The Solovay-Kitaev Algorithm,” *Quantum Information & Computation*, vol. 6, no. 1, pp. 81–95, Jan. 2006.

[58] V. Kliuchnikov, D. Maslov, and M. Mosca, “Fast and Efficient Exact Synthesis of Single-qubit Unitaries Generated by Clifford and T Gates,” *Quantum Information & Computation*, vol. 13, no. 7-8, pp. 607–630, Jul. 2013.

[59] P. Selinger, “Efficient Clifford+T Approximation of Single-qubit Operators,” *Quantum Information & Computation*, vol. 15, no. 1-2, pp. 159–180, Jan. 2015.

[60] V. Kliuchnikov, D. Maslov, and M. Mosca, “Practical Approximation of Single-Qubit Unitaries by Single-Qubit Quantum Clifford and T Circuits,” *IEEE Transactions on Computers*, vol. 65, no. 1, pp. 161–172, Jan. 2016.

[61] Vedral, V. and Barenco, A. and Ekert, A., “Quantum networks for elementary arithmetic operations,” *Physical Review A*, vol. 54, pp. 147–153, Jul. 1996.

[62] D. Beckman, A. N. Chari, S. Devabhaktuni, and J. Preskill, “Efficient networks for quantum factoring,” *Physical Review A*, vol. 54, pp. 1034–1063, Aug. 1996.

[63] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, “Quantum algorithms revisited,” *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1969, pp. 339–354, Jan. 1998.

[64] D. Deutsch and R. Jozsa, “Rapid Solution of Problems by Quantum Computation,” *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 439, no. 1907, pp. 553–558, Dec. 1992.

[65] A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. A. Spielman, "Exponential Algorithmic Speedup by a Quantum Walk," in *Proc. 35th Annual ACM Symposium on Theory of Computing (STOC'03)*, Jun. 2003, pp. 59–68.

[66] A. Childs, L. Schulman, and U. V. Vazirani, "Quantum Algorithms for Hidden Nonlinear Structures," in *Proc. 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, Oct. 2007, pp. 395–404.

[67] A. M. Childs and J. Goldstone, "Spatial search by quantum walk," *Physical Review A*, vol. 70, p. 022314, Aug. 2004.

[68] A. Ambainis, "Quantum Walk Algorithm for Element Distinctness," *SIAM J. Computing*, vol. 37, no. 1, pp. 210–239, Apr. 2007.

[69] H. Buhrman and R. Špalek, "Quantum Verification of Matrix Products," in *Proc. 7th Annual ACM-SIAM Symposium on Discrete Algorithm (S0DA'06)*, Jan. 2006, pp. 880–889.

[70] J. Kempe, "Quantum random walks : An introductory overview," *Contemporary Physics*, vol. 44, no. 4, pp. 307–327, 2003.

[71] A. Ambainis, "Quantum walks and their algorithmic applications," *International Journal of Quantum Information*, vol. 01, no. 04, pp. 507–518, Dec. 2003.

[72] S. Jordan. (2016, Apr.) Quantum Algorithm Zoo. Accessed 25 May 2016. [Online]. Available: http://math.nist.gov/quantum/zoo/

[73] D. S. Abrams and S. Lloyd, "Simulation of Many-Body Fermi Systems on a Universal Quantum Computer," *Physical Review Letters*, vol. 79, pp. 2586–2589, Sep. 1997.

[74] C. Zalka, "Simulating quantum systems on a quantum computer," *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1969, pp. 313–322, Jan. 1998.

[75] G. Ortiz, J. E. Gubernatis, E. Knill, and R. Laflamme, "Quantum algorithms for fermionic simulations," *Physical Review A*, vol. 64, p. 022319, Jul. 2001.

[76] R. Somma, G. Ortiz, J. E. Gubernatis, E. Knill, and R. Laflamme, "Simulating physical phenomena by quantum networks," *Physical Review A*, vol. 65, p. 042323, Apr. 2002.

[77] S. Raeisi, N. Wiebe, and B. C. Sanders, "Quantum-circuit design for efficient simulations of many-body quantum dynamics," *New Journal of Physics*, vol. 14, no. 10, p. 103017, 2012.

[78] K. L. Brown, W. J. Munro, and V. M. Kendon, "Using Quantum Computers for Quantum Simulation," *Entropy*, vol. 12, no. 11, p. 2268, Nov. 2010.

[79] I. M. Georgescu, S. Ashhab, and F. Nori, "Quantum simulation," *Review of Modern Physics*, vol. 86, pp. 153–185, Mar. 2014.

[80] D. P. DiVincenzo, "The Physical Implementation of Quantum Computation," *Fortschritte der Physik*, vol. 48, no. 9-11, pp. 771–783, Sep. 2000.

[81] P. Aliferis, D. Gottesman, and J. Preskill, “Quantum Accuracy Threshold for Concatenated Distance-3 Codes,” *Quantum Information & Compututation*, vol. 6, no. 2, pp. 97–165, Mar. 2006.

[82] A. Paetznick and B. W. Reichardt, “Fault-tolerant Ancilla Preparation and Noise Threshold Lower Bounds for the 23-qubit Golay Code,” *Quantum Information & Computation*, vol. 12, no. 11-12, pp. 1034–1080, Nov. 2012.

[83] A. Kitaev, “Quantum Error Correction with Imperfect Gates,” in *Quantum Communication, Computing, and Measurement*, O. Hirota, A. Holevo, and C. Caves, Eds. Springer, 1997, pp. 181–188.

[84] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, “Topological quantum memory,” *J. Mathematical Physics*, vol. 43, pp. 4452–4505, Aug. 2002.

[85] D. S. Wang, A. G. Fowler, and L. C. L. Hollenberg, “Surface code quantum computing with error rates over 1%,” *Physical Review A*, vol. 83, p. 020302, Feb. 2011.

[86] M. Suchara, J. Kubiatowicz, A. Faruque, F. Chong, C.-Y. Lai, and G. Paz, “QuRE: The Quantum Resource Estimator toolbox,” in *Proc. 31st IEEE International Conference on Computer Design (ICCD’13)*, Oct. 2013, pp. 419–426.

[87] M. Ahsan, C. B-S., and K. J., “Performance simulator based on hardware resources constraints for ion trap quantum computer,” in *Proc. 31st IEEE International Conference on Computer Design (ICCD’13)*, Oct. 2013, pp. 411–418.

[88] M. Ahsan, R. V. Meter, and J. Kim, “Designing a Million-Qubit Quantum Computer Using a Resource Performance Simulator,” *ACM J. Emerging Technologies in Computing Systems*, vol. 12, no. 4, pp. 39:1–39:25, Dec. 2015.

[89] D. L. Moehring, C. Highstrete, D. Stick, K. M. Fortier, R. Haltli, C. Tigges, and M. G. Blain, “Design, fabrication and experimental demonstration of junction surface ion traps,” *New Journal of Physics*, vol. 13, no. 7, p. 075018, Jul. 2011.

[90] Aude Craik, D. P. L. and Linke, N. M. and Harty, T. P. and Ballance, C. J. and Lucas, D. M. and Steane, A. M. and Allcock, D. T. C., “Microwave control electrodes for scalable, parallel, single-qubit operations in a surface-electrode ion trap,” *Applied Physics B*, vol. 114, no. 1, pp. 3–10, Jan. 2014.

[91] T. P. Harty, D. T. C. Allcock, C. J. Ballance, L. Guidoni, H. A. Janacek, N. M. Linke, D. N. Stacey, and D. M. Lucas, “High-Fidelity Preparation, Gates, Memory, and Readout of a Trapped-Ion Quantum Bit,” *Physical Review Letters*, vol. 113, p. 220501, Nov. 2014.

[92] D. Leibfried, E. Knill, S. Seidelin, J. Britton, R. B. Blakestad, J. Chiaverini, D. B. Hume, W. M. Itano, J. D. Jost, C. Langer, R. Ozeri, R. Reichle, and D. J. Wineland, “Creation of a six-atom ’Schrödinger cat’ state,” *Nature*, vol. 438, no. 7068, pp. 639–642, Dec. 2005.

[93] H. Häffner, W. Hänsel, C. F. Roos, J. Benhelm, D. Chek-al-kar, M. Chwalla, T. Körber, U. D. Rapol, M. Riebe, P. O. Schmidt, C. Becher, O. Gühne, W. Dür, and R. Blatt, “Scalable multiparticle entanglement of trapped ions,” *Nature*, vol. 438, no. 7068, pp. 643–646, Dec. 2005.

[94] D. Stick, W. K. Hensinger, S. Olmschenk, M. J. Madsen, K. Schwab, and C. Monroe, “Ion trap in a semiconductor chip,” *Nature Physics*, vol. 2, no. 1, pp. 36–39, Jan. 2006.

[95] B. P. Lanyon, C. Hempel, D. Nigg, M. Müller, R. Gerritsma, F. Zähringer, P. Schindler, J. T. Barreiro, M. Rambach, G. Kirchmair, M. Hennrich, P. Zoller, R. Blatt, and C. F. Roos, “Universal Digital Quantum Simulation with Trapped Ions,” *Science*, vol. 334, no. 6052, pp. 57–61, Oct. 2011.

[96] P. Schindler, D. Nigg, T. Monz, J. Barreiro, E. Martinez, S. Wang, S. Quint, M. Brandl, V. Nebendahl, C. Roos, M. Chwalla, M. Hennrich, and R. Blatt, “A quantum information processor with trapped ions,” *New Journal of Physics*, vol. 15, no. 12, p. 123012, Oct. 2013.

[97] T. Monz, P. Schindler, J. T. Barreiro, M. Chwalla, D. Nigg, W. A. Coish, M. Harlander, W. Hänsel,W., M. Hennrich, and R. Blatt, “14-Qubit Entanglement: Creation and Coherence,” *Physical Review Letters*, vol. 106, p. 130506, Mar. 2011.

[98] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. C. White, J. Mutus, A. G. Fowler, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, C. Neill, P. O’Malley, P. Roushan, A. Vainsencher, J. Wenner, A. N. Korotkov, A. N. Cleland, and J. M. Martinis, “Superconducting quantum circuits at the surface code threshold for fault tolerance,” *Nature*, vol. 508, no. 7497, pp. 500–503, Apr. 2014.

[99] E. Lucero, R. Barends, Y. Chen, J. Kelly, M. Mariantoni, A. Megrant, P. O’Malley, D. Sank, A. Vainsencher, J. Wenner, T. White, Y. Yin, A. N. Cleland, and J. M. Martinis, “Computing prime factors with a Josephson phase qubit quantum processor,” *Nature Physics*, vol. 8, no. 10, pp. 719–723, Oct. 2012.

[100] J. Kelly, R. Barends, A. G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, J. Y. Mutus, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, I.-C. Hoi, C. Neill, P. J. J. O’Malley, C. Quintana, P. Roushan, A. Vainsencher, J. Wenner, A. N. Cleland, and J. M. Martinis, “State preservation by repetitive error detection in a superconducting quantum circuit,” *Nature*, vol. 519, no. 7541, pp. 66–69, Mar. 2015.

[101] E. Knill, R. Laflamme, and G. J. Milburn, “A scheme for efficient quantum computation with linear optics,” *Nature*, vol. 409, no. 6816, pp. 46–52, Jan. 2001.

[102] B. P. Lanyon, T. J. Weinhold, N. K. Langford, M. Barbieri, D. F. V. James, A. Gilchrist, and A. G. White, “Experimental Demonstration of a Compiled Version of Shor’s Algorithm with Quantum Entanglement,” *Physical Review Letters*, vol. 99, p. 250505, Dec. 2007.

[103] C. Lu, D. E. Browne, T. Yang, and J.-W. Pan, “Demonstration of a Compiled Version of Shor’s Quantum Factoring Algorithm Using Photonic Qubits,” *Physical Review Letters*, vol. 99, p. 250504, Dec. 2007.

[104] E. Martín-López, A. Laing, T. Lawson, R. Alvarez, X.-Q. Zhou, and J. L. O’Brien, “Experimental realization of Shor’s quantum factoring algorithm using qubit recycling,” *Nature Photonics*, vol. 6, no. 11, pp. 773–776, Nov. 2012.

[105] J. A. Jones, M. Mosca, and R. H. Hansen, “Implementation of a quantum search algorithm on a quantum computer,” *Nature*, vol. 393, no. 6683, pp. 344–346, May 1998.

[106] S. Haroche, “Quantum Information with Atoms and Photons in a Cavity: Entanglement, Complementarity and Decoherence Studies,” in *Condensation and Coherence in Condensed Matter*, T. Claeson and P. Delsing, Eds. Physica Scripta and Wold Scientific, 2011, pp. 128–132.

[107] R. Miller, T. E. Northup, K. M. Birnbaum, A. Boca, A. D. Boozer, and H. J. Kimble, “Trapped atoms in cavity QED: coupling quantized light and matter,” *Journal of Physics B: Atomic, Molecular and Optical Physics*, vol. 38, no. 9, p. S551, Apr. 2005.

[108] M. Veldhorst, C. H. Yang, J. C. C. Hwang, W. Huang, J. P. Dehollain, J. T. Muhonen, S. Simmons, A. Laucht, F. E. Hudson, K. M. Itoh, A. Morello, and A. S. Dzurak, “A two-qubit logic gate in silicon,” *Nature*, vol. 526, no. 7573, pp. 410–414, Oct. 2015.

[109] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, “Quantum Computation by Adiabatic Evolution,” Massachusetts Institute of Technology, Center for Theoretical Physics and Department of Mathematics, Tech. Rep. MIT/CTP/2936, Jan. 2000.

[110] T. Lanting, A. J. Przybysz, A. Y. Smirnov, F. M. Spedalieri, M. H. Amin, A. J. Berkley, R. Harris, F. Altomare, S. Boixo, P. Bunyk, N. Dickson, C. Enderud, J. P. Hilton, E. Hoskinson, M. W. Johnson, E. Ladizinsky, N. Ladizinsky, R. Neufeld, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, S. Uchaikin, A. B. Wilson, and G. Rose, “Entanglement in a Quantum Annealing Processor,” *Physical Review X*, vol. 4, p. 021041, May 2014.

[111] C. C. McGeoch and C. Wang, “Experimental Evaluation of an Adiabatic Quantum System for Combinatorial Optimization,” in *Proc. 10th ACM International Conference on Computing Frontiers (CF,13)*, May 2013, pp. 23:1–23:11.

[112] S. Boixo, T. F. Ronnow, S. V. Isakov, Z. Wang, D. Wecker, D. A. Lidar, J. M. Martinis, and M. Troyer, “Evidence for quantum annealing with more than one hundred qubits,” *Nature Physics*, vol. 10, no. 3, pp. 218–224, Mar. 2014.

[113] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer, “Defining and detecting quantum speedup,” *Science*, vol. 345, no. 6195, pp. 420–424, Jul. 2014.

[114] P. W. Shor, “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer,” *SIAM J. Computing*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997.

[115] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*, 1st ed. CRC Press, 1996.

[116] R. B. Griffiths and C.-S. Niu, “Semiclassical Fourier Transform for Quantum Computation,” *Physical Review Letters*, vol. 76, pp. 3228–3231, Apr. 1996.

[117] A. Ekert and R. Jozsa, “Quantum computation and Shor’s factoring algorithm,” *Review of Modern Physics*, vol. 68, pp. 733–753, Jul. 1996.

[118] A. Kitaev, “Quantum measurements and the Abelian Stabilizer Problem,” Electronic Colloquium on Computational Complexity, Tech. Rep. TRS96-003, Jan. 1996.

[119] D. Boneh and R. J. Lipton, “Quantum cryptanalysis of hidden linear functions (extended abstract),” in *Proc. 15th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO’95)*, Aug. 1995, pp. 424–437.

[120] M. Mosca and A. Ekert, “The hidden subgroup problem and eigenvalue estimation on a quantum computer,” in *Quantum Computing and Quantum Communications: First NASA International Conference, (QCQC’98), Selected Papers*, C. P. Williams, Ed. Springer, 1999, pp. 174–188.

[121] C. Lomont, “The Hidden Subgroup Problem - Review and Open Problems,” *eprint arXiv:quant-ph/0411037*, Nov. 2004.

[122] T. Granlund and P. L. Montgomery, “Division by Invariant Integers Using Multiplication,” in *Proc. ACM SIGPLAN Conference on Programming Language Design and Implementation 1994 (PLDI’94)*, Jun. 1994, pp. 61–72.

[123] 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, Sep. 2005.

[124] S. Parker and M. B. Plenio, “Efficient Factorization with a Single Pure Qubit and $\log N$ Mixed Qubits,” *Physical Review Letters*, vol. 85, pp. 3049–3052, Oct. 2000.

[125] S. Cuccaro, T. Draper, S. Kutin, and D. Petrie Moulton, “A new quantum ripple-carry addition circuit,” *eprint arXiv:quant-ph/0410184*, Oct. 2004.

[126] T. G. Draper, S. A. Kutin, E. M. Rains, and K. M. Svore, “A Logarithmic-depth Quantum Carry-lookahead Adder,” *Quantum Information & Computation*, vol. 6, no. 4, pp. 351–369, Jul. 2006.

[127] P. Gossett, “Quantum Carry-Save Arithmetic,” *eprint arXiv:quant-ph/9808061*, Aug. 1998.

[128] C. Zalka, “Fast versions of Shor’s quantum factoring algorithm,” *eprint arXiv:quant-ph/9806084*, Jun. 1998.

[129] R. Van Meter and K. M. Itoh, “Fast quantum modular exponentiation,” *Physical Review A*, vol. 71, p. 052320, May 2005.

[130] A. G. Fowler, S. J. Devitt, and L. C. L. Hollenberg, “Implementation of Shor’s Algorithm on a Linear Nearest Neighbour Qubit Array,” *Quantum Information & Computation*, vol. 4, no. 4, pp. 237–251, Jul. 2004.

[131] S. A. Kutin, “Shor’s algorithm on a nearest-neighbor machine,” in *Asian Conference on Quantum Information Science (AQIS’07)*, Sep. 2007, eprint arXiv:quant-ph/0609001.

[132] B.-S. Choi and R. Van Meter, “A $\Theta(\sqrt{n})$-depth Quantum Adder on the 2D NTC Quantum Computer Architecture,” *ACM J. Emerging Technologies in Computing Systems*, vol. 8, no. 3, pp. 24:1–24:22, Aug. 2012.

[133] P. Pham and K. M. Svore, “A 2D Nearest-Neighbor Quantum Architecture for Factoring in Polylogarithmic Depth,” *Quantum Information & Computation*, vol. 13, no. 11-12, pp. 937–962, Nov. 2013.

[134] B. P. Lanyon, M. Barbieri, M. P. Almeida, T. Jennewein, T. C. Ralph, K. J. Resch, G. J. Pryde, J. L. O'Brien, A. Gilchrist, and A. G. White, "Simplifying quantum logic using higher-dimensional Hilbert spaces," *Nature Physics*, vol. 5, no. 2, pp. 134–140, Feb. 2009.

[135] T. C. Ralph, K. J. Resch, and A. Gilchrist, "Efficient Toffoli gates using qudits," *Physical Review A*, vol. 75, p. 022313, Feb. 2007.

[136] T. Monz, K. Kim, W. Hänsel, M. Riebe, A. S. Villar, P. Schindler, M. Chwalla, M. Hennrich, and R. Blatt, "Realization of the Quantum Toffoli Gate with Trapped Ions," *Physical Review Letters*, vol. 102, p. 040501, Jan. 2009.

[137] A. Fedorov, L. Steffen, M. Baur, M. P. da Silva, and A. Wallraff, "Implementation of a Toffoli gate with superconducting circuits," *Nature*, vol. 481, no. 7380, pp. 170–172, Jan. 2012.

[138] P. Kaye, "Optimized Quantum Implementation of Elliptic Curve Arithmetic over Binary Fields," *Quantum Information & Computation*, vol. 5, no. 6, pp. 474–491, Sep. 2005.

[139] B. Amento, M. Rötteler, and R. Steinwandt, "Efficient Quantum Circuits for Binary Elliptic Curve Arithmetic: Reducing T-gate Complexity," *Quantum Information & Computation*, vol. 13, no. 7-8, pp. 631–644, Jul. 2013.

[140] N. Moller and T. Granlund, "Improved Division by Invariant Integers," *IEEE Transactions on Computers*, vol. 60, no. 2, pp. 165–175, Feb. 2011.

[141] A. Pavlidis and D. Gizopoulos, "Fast Quantum Modular Exponentiation Architecture for Shor's Factoring Algorithm," *Quantum Information & Computation*, vol. 14, no. 7&8, pp. 649–682, May 2014.

[142] R. Van Meter,III, "Architecture of a Quantum Multicomputer Optimized for Shor's Factoring Algorithm," Ph.D. dissertation, Keio University, Jul. 2006.

[143] Y. Takahashi and N. Kunihiro, "A quantum circuit for shor's factoring algorithm using $2n+2$ qubits," *Quantum Information & Computation*, vol. 6, no. 2, pp. 184–192, Mar. 2006.

[144] A. M. Steane, "Overhead and noise threshold of fault-tolerant quantum error correction," *Phys. Rev. A*, vol. 68, p. 042322, Oct 2003.

[145] A. G. Fowler and L. C. L. Hollenberg, "Scalability of Shor's algorithm with a limited set of rotation gates," *Physical Review A*, vol. 70, p. 032329, Sep. 2004.

[146] A. G. Fowler and L. C. L. Hollenberg, "Erratum: Scalability of Shor's algorithm with a limited set of rotation gates [Phys. Rev. A 70, 032329 (2004)]," *Physical Review A*, vol. 75, p. 029905, Feb. 2007.

[147] T. T. Pham, R. Van Meter, and C. Horsman, "Optimization of the Solovay-Kitaev algorithm," *Physical Review A*, vol. 87, p. 052332, May 2013.

[148] N. C. Jones, J. D. Whitfield, P. L. McMahon, M.-H. Yung, R. V. Meter, A. Aspuru-Guzik, and Y. Yamamoto, "Faster quantum chemistry simulation on fault-tolerant quantum computers," *New Journal of Physics*, vol. 14, no. 11, p. 115023, Nov. 2012.

[149] A. Bocharov, Y. Gurevich, and K. M. Svore, "Efficient decomposition of single-qubit gates into V basis circuits," *Physical Review A*, vol. 88, p. 012313, Jul. 2013.

[150] R. Raussendorf, D. E. Browne, and H. J. Briegel, "Measurement-based quantum computation on cluster states," *Physical Review A*, vol. 68, p. 022312, Aug. 2003.

[151] A. Trisetyarso and R. Van Meter, "Circuit Design for A Measurement-Based Quantum Carry-Lookahead Adder," *International Journal of Quantum Information*, vol. 08, no. 05, pp. 843–867, Aug. 2010.

[152] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Physical Review A*, vol. 86, p. 032324, Sep. 2012.

[153] R. Van Meter, T. D. Ladd, A. G. Fowler, and Y. Yamamoto, "Distributed Quantum Computation Architecture Using Semiconductor Nanophotonics," *International Journal of Quantum Information*, vol. 8, no. 1&2, pp. 295–323, Feb. 2010.

[154] N. C. Jones, R. Van Meter, A. G. Fowler, P. L. McMahon, J. Kim, T. D. Ladd, and Y. Yamamoto, "Layered Architecture for Quantum Computing," *Physical Review X*, vol. 2, p. 031007, Jul. 2012.

[155] A. Barenco, A. Ekert, K.-A. Suominen, and P. Törmä, "Approximate quantum Fourier transform and decoherence," *Physical Review A*, vol. 54, pp. 139–146, Jul. 1996.

[156] D. Coppersmith, "An approximate Fourier transform useful in quantum factoring," IBM Research Division,T.J.Watson Research Center, Tech. Rep. RC 19642, Jan. 1994.

[157] Y. S. Nam and R. Blümel, "Performance scaling of Shor's algorithm with a banded quantum Fourier transform," *Physical Review A*, vol. 86, p. 044303, Oct. 2012.

[158] Y. S. Nam and R. Blümel, "Robustness and performance scaling of a quantum computer with respect to a class of static defects," *Physical Review A*, vol. 88, p. 062310, Dec. 2013.

[159] Y. S. Nam and R. Blümel, "Streamlining Shor's algorithm for potential hardware savings," *Physical Review A*, vol. 87, p. 060304, Jun. 2013.

[160] Y. S. Nam and R. Blümel, "Analytical formulas for the performance scaling of quantum processors with a large number of defective gates," *Physical Review A*, vol. 92, p. 042301, Oct. 2015.

[161] E. Knill, "Approximation by quantum circuits," Los Alamos National Laboratory, Tech. Rep. LA-UR-95-2225, Aug. 1995.

[162] B. Kraus and J. I. Cirac, "Optimal creation of entanglement using a two-qubit gate," *Physical Review A*, vol. 63, p. 062309, May 2001.

[163] M. J. Bremner, C. M. Dawson, J. L. Dodd, A. Gilchrist, A. W. Harrow, D. Mortimer, M. A. Nielsen, and T. J. Osborne, "Practical Scheme for Quantum Computation with Any Two-Qubit Entangling Gate," *Physical Review Letters*, vol. 89, p. 247902, Nov. 2002.

[164] A. Pavlidis and D. Gizopoulos, “Hierarchical synthesis of quantum and reversible architectures,” in *Proc. 12th ACM International Conference on Computing Frontiers (CF’15)*, 2015, pp. 13:1–13:8.

[165] A. Pavlidis and D. Gizopoulos, “Hierarchical synthesis of quantum and reversible architectures,” *International Journal of Parallel Programming*, vol. 44, no. 5, pp. 1028–1053, Oct. 2016.

[166] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, “Synthesis of reversible logic circuits,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 6, pp. 710–722, Jun. 2003.

[167] A. K. Prasad, V. V. Shende, I. L. Markov, J. P. Hayes, and K. N. Patel, “Data Structures and Algorithms for Simplifying Reversible Circuits,” *ACM J. Emerging Technologies in Computing Systems*, vol. 2, no. 4, pp. 277–293, Oct. 2006.

[168] O. Golubitsky and D. Maslov, “A Study of Optimal 4-Bit Reversible Toffoli Circuits and Their Synthesis,” *IEEE Transactions on Computers*, vol. 61, no. 9, pp. 1341–1353, Sep. 2012.

[169] D. M. Miller, D. Maslov, and G. W. Dueck, “A transformation based algorithm for reversible logic synthesis,” in *Proc. 40th Annual Design Automation Conference (DAC’03)*, Jun. 2003, pp. 318–323.

[170] D. Maslov, G. W. Dueck, and D. M. Miller, “Techniques for the Synthesis of Reversible Toffoli Networks,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 12, no. 4, pp. 42:1–42:28, Sep. 2007.

[171] P. Gupta, A. Agrawal, and N. K. Jha, “An Algorithm for Synthesis of Reversible Logic Circuits,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 11, pp. 2317–2330, Nov. 2006.

[172] M. Saeedi, M. S. Zamani, M. Sedighi, and Z. Sasanian, “Reversible Circuit Synthesis Using a Cycle-based Approach,” *ACM J. Emerging Technologies in Computing Systems*, vol. 6, no. 4, pp. 13:1–13:26, Dec. 2010.

[173] R. Wille and R. Drechsler, “BDD-based Synthesis of Reversible Logic for Large Functions,” in *Proc. 46th Annual Design Automation Conference (DAC’09)*, 2009, pp. 270–275.

[174] J. J. Vartiainen, M. Möttönen, and M. M. Salomaa, “Efficient decomposition of quantum gates,” *Physical Review Letters*, vol. 92, p. 177902, Apr. 2004.

[175] A. JavadiAbhari, S. Patil, D. Kudrow, J. Heckey, A. Lvov, F. T. Chong, and M. Martonosi, “ScaffCC: A Framework for Compilation and Analysis of Quantum Computing Programs,” in *Proc. 11th ACM International Conference on Computing Frontiers (CF’14)*, May 2014, pp. 1:1–1:10.

[176] A. S. Green, P. L. Lumsdaine, N. J. Ross, P. Selinger, and B. Valiron, “Quipper: A Scalable Quantum Programming Language,” in *Proc. 34th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI’13)*, Jun. 2013, pp. 333–342.

[177] X. Liu and J. Kubiatowicz, “Chisel-Q: Designing quantum circuits with a scala embedded language,” in *Proc. 31st IEEE International Conference on Computer Design (ICCD’13)*, Oct. 2013, pp. 427–434.

[178] R. Wille and R. Drechsler, *Towards a Design Flow for Reversible Logic*. Springer, 2010.

[179] C.-C. Lin, A. Chakrabarti, and N. K. Jha, “QLib: Quantum Module Library,” *ACM J. Emerging Technologies in Computing Systems*, vol. 11, no. 1, pp. 7:1–7:20, Sep. 2014.

[180] R. Wille, S. Offermann, and R. Drechsler, “SyReC: A programming language for synthesis of reversible circuits,” in *2010 Forum on Specification Design Languages (FDL 2010)*, Sep. 2010, pp. 1–6.

[181] R. Drechsler and R. Wille, “Reversible Circuits: Recent Accomplishments and Future Challenges for an Emerging Technology,” in *Proc. 16th International Conference on Progress in VLSI Design and Test 2012 (VDAT’12)*, 2012, pp. 383–392.

[182] G.H.Hardy and E.M.Wright, *An Introduction to the Theory of Numbers*, 4th ed. Clarendon Press, 1960.