Kvantecomputere og fejlretning på kvantecomputere

Frederik Treue

DIKU 2006

Indhold

1 Indledning

1.1 Formål

Denne rapport har to grundlæggende formål: Dels at dokumentere en forståelse af kvantecomputere og dels at opstille en funktion for det minimale antal "basale instruktioner" pr. tidsenhed, som en kvantecomputer nødvendigvis må kunne udføre for at implementere Shor's algoritme når implementationen skal tage hensyn til støj i kvantecomputeren, samt at forklare disse emner på en måde, så det kan forstås af en datalogi studerende uden forudgående kendskab til fysik (men dog med kendskab til matematisk analyse og algebra).

Det første delmål dokumenteres i

- afsnit 2 med en basal redegørelse for, hvorledes en kvantecomputer fungerer, samt en fastlæggelse af de grundlæggende begreber.
- afsnit 3 hvor et netværk for Shor's algoritme opstilles og der argumenteres for korrektheden af dette, under antagelse af at der ikke eksisterer nogen form for støj i systemet.
- afsnit 4 hvor de nødvendige dele af teorien for "kvante fejl retning"(QEC[1]) og fejltolerant beregning beskrives, og Shor's algoritme implicit bliver gjort resistent over for støj.

Det sidste delmål dokumenteres i afsnit 5 udfra teorien udledt i afsnit 4.

1.2 Oversigt over afsnit

Vi præsenterer her for overskuelighedens skyld en oversigt over formålene med hvert af hovedafsnittene i denne rapport:
Afsnit 2 introduceres læseren til de grundlæggende koncepter i kvantecomputeren.
Afsnit 3 forklarer Shor's algoritme grundigt, men uden hensyntagen til støj i systemet.
Afsnit 4 forklarer og løser problemerne med støj i kvantecomputeren på et abstrakt niveau. Resultatet heraf bliver, at implementationen af en fejltollerant version af Shor's algoritme bliver triviel.
Afsnit 5 forklarer begrebet grænseanalyse, som er det værktøj der kan frembringe den funktion er ønsket i formålet.

1.3 Afgrænsninger

Jeg vil ikke beskæftige mig med den fysiske implementation af kvantecomputeren. Jeg antager derfor ikke noget om hvilken implementation jeg arbejder med.

Den støj som skal behandles i rapporten antages at være begrænset til hukommelsen af computeren - operationer på kvantecomputeres tilstand (kvantegates) antages at forløbe fejlfrit. Desuden antages det at der ikke er nogen

[1]Quantum Error Correction

kobling imellem fejlrisikoen på de individuelle qubit - mao. er risikoen for at der optræder støj på en qubit uafhængig af risikoen for støj på de andre qubits.

2 Basale begreber og vedtægter

2.1 Tilstande

I den klassiske computer repræsenteres tilstanden ved en vektor af bits, dvs. elementer som *enten* er 0 *eller* 1. Kvantecomputeren kan delvist forstås som en slags generalisering af dette: Vi specificerer, ligesom i den klassiske computer, tilstanden ved en vektor af elementer (kaldet qubits for QUantum BITS), men disse elementer kan nu antage superpositioner af $|0>$ og $|1>$. Dvs. en qubit har generelt tilstanden

$$c_0|0> + c_1|1>, |c_0|^2 + |c_1|^2 = 1$$

hvilket vil virke gammelkendt for fysikere [2], men for dataloger uden forudgående kendskab til kvantemekanik er det simpleste at tænke på en qubit som en normaliseret vektor i et hilbertrum. Det kan her betale sig at minde om/indføre Dirac notation - faktisk er den allerede indført med f.eks. $|1>$, som er en abstrakt beskrivelse af en fysisk tilstand eller, ækvivalent hermed, en vektor i et hilbertrum - denne beskrivelse kaldes konventionelt en "ket". Den hermittisk konjugerede til en ket, betegnet f.eks. $<1|$, kaldes en "bra", og det indre produkt af disse, $<1|1>$, kaldes følgeligt for en "bra(c)ket".
Det må understreges at der ikke er mere information i en qubit end i en bit - de to koefficienter i 2.1 ændrer ikke på dette. Ganske vist kan vi have et kontinuum af mulige tilstande for en enkelt qubit, men vi kan ikke måle disse koefficienter umiddelbart - en kvantemekanisk måling svarer til at projicere tilstanden i 2.1 ned på den orthonormale basis $\{|0>, |1>\}$ og en kvantecomputer vil så give en værdi som svarer til tilstanden $|0>$ ($a_0 =< 0|M|\Psi >$), med en sandsynlighed, $|c_0|^2$, hvor M er den operator som har egenskaben at $M|0> = a_0|0>$ og $M|1> = a_1|1>$ og $|\Psi>$ er tilstanden af kvantecomputeren inden denne måles), og med tilsvarende sandsynligheden $|c_1|^2$ en værdi som svarer til tilstanden $|1>$ ($a_1 =< 1|M|\Psi >$). Dette illustrerer den basalt probabilistiske natur i en kvantecomputer - selvom det er muligt at lave tilstande, hvorom den for alle qubits gælder, at enten er $c_1 = 1$ eller $c_0 = 1$, vil man i det generelle tilfælde have en *superposition* af $|1>$ og $|0>$ i hvilke tilfælde vi ikke på forhånd kan sige, hvilken værdi (a_0 eller a_1) kvantecomputeren vil returnere.
Bemærk også, at en måling af tilstanden i fysisk forstand projicerer enhver tilstand af qubit'en ned på enten $|0>$ eller $|1>$. Dette betyder, at en måling af en tilstand *uværligt* vil ødelægge denne tilstand - dette er en af de basale grunde til at problemerne med at rette fejl i en kvantecomputer er meget større end de er på en klassisk computer.
Man er foruden ovenforstående også nødt til at forstå begrebet "entangelment" (sammenfiltring). Det simpleste eksempel på dette fåes med to qubits i tilstanden $\frac{1}{\sqrt{2}} * (|01> + |10>)$, hvor kettene beskriver qubit 1's tilstand på første plads og qubit 2's tilstand på anden plads. Hvis vi i denne tilstand måler på qubit 1's tilstand, så projiceres tilstanden ned på sættet $\{|01>, |10>\}$, og vi er

[2] $|0>$ og $|1>$ er simpelhent orthonormale tilstandsfunktioner for et (abstrakt) system, som udgør kvantecomputeren - et "kanonisk eksempel er et spin-$\frac{1}{2}$ system, hvor $|0>$ svarer til spin op og $|1>$ svarer til spin ned.

altså efter målingen helt sikre på qubit 2's tilstand, selvom vi aldrig har målt den direkte. Denne situation står i kontrast til situationen, hvor vi kun ved at qubit 1 er $|0>$ eller $|1>$ og qubit 2 er $|0>$ eller $|1>$, nemlig tilstanden $\frac{1}{\sqrt{4}}(|00>+|01>+|10>+|11>)$.
I ovenstående har vi hele tiden underforstået at beregningsbasen (dvs. den basis af tilstande, som vi projicere ned på, når vi måler) har været $\{|0>,|1>\}$. Dette er ikke den eneste mulighed - vi kunne ligesågodt vælge $\{\frac{1}{\sqrt{2}}(|0>+|1>$ $), \frac{1}{\sqrt{2}}(|0>-|1>)\}$. Det eneste vi kræver er, at sættet er orthonormalt, og at det udspænder hele udfaldsrummet. I de netværk, som bliver præsenteret i denne rapport, vil vi engang imellem skifte basis for at opnå en fordelagtig effekt.

2.2 Operationer

Et af de basale resultater i kvantefysikken er, at tidsudvikling af tilstande (som f.eks. $|0>$ eller $|1>$) altid kan beskrives ved en unitær operator. Idet det eneste vi kan gøre ved en kvantecomputers tilstand (udover at måle den), er at anvende en given tidsudvikling på tilstanden, må det kræves, at alle beregninger på kvantecomputeren kan beskrives som en unitær transformation af en på forhånd bestemt starttilstand. Hvis kvantecomputeren har n qubits er klart, at tilstanden kan beskrives som en vektor i et 2^n-dimentionalt hilbertrum (da hver qubit har 2 forskellige grundtilstande), hvorfor det også er klart, at kvantecomputerens operation kan beskrives som en $2^n * 2^n$ unitær matrix. Det er dog i praksis usandsynligt at vi kan implementere en given unitær matrice "på en gang" i en kvantecomputer - vi vil normalt have en række basale operationer (dvs. unitære matricer) til rådighed, som vi kombinerer til den egentlige operation. Det er heldigvis bevist at næsten ethvert sæt af unitære matricer, som virker på to qubits (dvs. har elementer forskellige fra 0 udenfor diagonalen i 4 rækker/kolonner) er komplet, i den forstand at enhver unitær matrix kan laves ved at sammenkoble en række af disse basale operationer.

En vigtig konsekvens ved unitariteten af kvantecomputerens operation er, at en kvantecomputer altid skal være reversibel, idet den omvendte operation jo også er en unitær operation (hvis U er unitær, er U^{-1} det selvfølgeligt også), og alle unitære operationer kan implementeres på en kvantecomputer - faktisk vil den omvendte operation jo bare være de basale instruktioners omvendte taget i omvendt rækkefølge, idet $(U_1U_2U_3\dots U_n)^{-1} = U_n^{-1}U_{n-1}^{-1}U_{n-2}^{-1}\dots U_1^{-1}$. Dette fører til nogle konsekvenser for hvordan kvantecomputeren behandler den data, som vi anvender den på: lad os antage at vi ønsker at konstruere en kvantecomputer, som kan beregne funktionen $f(x) : \{0,1\}^n \to \{0,1\}^n$. Der er grundlæggende to muligheder: enten er $f(x)$ invertibel og vi kan derfor finde en unitær transformation U således at $U|x>=|f(x)>$ (i dette og i det følgende beskriver $|x>$ en vektor i det relevante 2^n-dimentionelle hilbertrum, og ikke bare en enkelt qubits tilstand). Men hvis $f(x)$ *ikke* er invertibel, kan der ikke eksistere en unitær operator U som implementerer denne funktion umiddelbart, idet der så ville være en operator U^{-1} som har effekten $U^{-1}|f(x)>=U^{-1}U|x>=|x>=|f^{-1}(f(x))>$ og dermed implementerer $f^{-1}(x)$, i modstrid med at denne ikke eksister. Dette problem kan dog relativt let løses ved at implementere transfor-

mationen $U(|x>_{ind} \otimes |0>_{ud}) = |x>_{ind} \otimes |f(x)>_{ud}$[3], dvs. ved at beholde den oprindelige inddata hele vejen igennem netværket. Dermed er der intet krav om, at U^{-1} implementerer $f^{-1}(x)$ idet den omvendte transformation bare skal nulstille $|>_{ud}$.

Idet operationen af en kvantecomputer kan beskrives som en lineær operator (da den unitære operator altid kan skrives som en endeligdimentional matrix), kan vi let indse hvordan en kvantecomputer opererer på en superposition: Idet vi beskriver en generel tilstand af en n-qubit kvantecomputer som $|\Psi> = \frac{1}{\sqrt{N}} \sum_{i=0}^{2^n-1} c_i|i>$ hvor $|i>$ skal forstås som den "rene" tilstand som svarer til tallet i, dvs. hvor tilstandsvektoren er den binære repræsentation af tallet i, og hvor N blot er en normerings konstant som sørger for at $<\Psi|\Psi> = 1$, ser vi, at $U|\Psi> = \frac{1}{\sqrt{N}} \sum_{i=0}^{2^n-1} c_i U|i>$. Dette har som konsekvens, at hvis vi kan vise at et givet kvantenetværk implementerer operationen U på enhver ren tilstand, så implementerer det også U på enhver superposistion af tilstandene $|i>$.

2.2.1 Basale operationer

I bestemmelsen af tidskompleksisteten af kvantealgoritmerne må vi vedtage nogle basale operatorer, som antages at kunne implementeres i konstant tid. Hvilke operatorer dette præcist vil være, afhænger af implementationen af den kvantecomputeren, som jeg, jf. afsnit 1.3, ikke vil antage noget om, hvorfor jeg relativt arbitrært vælger de operationer som Barenco et al. definerer i [1] afsnit IV, samt en controled-NOT (CNOT) gate, som de "basale" operationer. En CNOT gate er en gate, som opererer på to qubits således, at hvis den første ("kontrol-qubit'en") er i tilstand $|1>$, negeres den anden ("mål-qubit'en"), og hvis kontrol-qubit'en er i tilstand $|0>$ er gaten triviel, dvs. identitets afbilledingen. Vi vil igennem hele denne rapport anvende matricer til at beskrive forskellige gates, som vi vil opskrive i basen $\{|0\ldots00>, |0\ldots01>, |0\ldots10>, |0\ldots11>, \ldots, |1\ldots10>, |1\ldots11>\}$, hvor vi naturligvis kun medtager det relevante antal qubits, f.eks. 2 for en CNOT gate. På figur 1 ses matricen for en CNOT gate i denne basis. De netværk, som vi vil specificere i de følgende afsnit,

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

Figur 1: Matricen for en $CNOT$ gate

vil bestå af ganske få forskellige operationer, som vi her vil bygge udfra et konstant antal af de ovenstående basale gates. Formålet med at gøre det fra starten er, at vi så vil kunne betragte dem som "basale" gates, uden at bekymre os om, hvordan de implementeres i fuld detalje. Vi for brug for den såkaldte Hadamard

[3] hvor $|>_{ind}$ repræsenterer en $2*n$ dimentionel vektor og $|>_{ud}$ repræsenterer en $2l$-dimentionel vektor, hvor $f(x)$ kan skrives med l qubits for alle mulige værdier af x

gate, som defineres ved at $H|0>=\frac{1}{\sqrt{2}}(|0>+|1>)$ og $H|1>=\frac{1}{\sqrt{2}}(|0>-|1>)$ - denne kan ses som en transformation imellem den normale beregningsbasis og den alternative basis som blev specificeret sidst i afsnit 2.1. Vi ser at, idet vi anvender navnene fra [1], $H = R_y(\frac{\pi}{2})\sigma_x$, hvilket i matrixform illustreres i figur 2 - σ_x er her en af Pauli matricerne, som vi vil anvende gennem hele rapporten, og som vi derfor vil definere her, én gang for alle:

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad \sigma_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{1}$$

Vi vil derudover få brug for en faseskifts operator, som er defineret som

$$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} \cos(\pi/4) & \sin(\pi/4) \\ -\sin(\pi/4) & \cos(\pi/4) \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Figur 2: Matricen for en hadamard gate

$P(d) = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i*\pi}{2^d}} \end{bmatrix}$, hvor d er et heltal, som vil blive forklaret i 3.1.1. Denne transformation er let at konstruere, igen udfra de basale gates specificeret i [1]: $P(d) = R_z(-\frac{\pi}{2^d})\Phi(\frac{\pi}{2*2^d})$

I de afsnit 3 vil vi også anvende C^nNOT gates, dvs. CNOT gates som blot har flere kontrol qubit's, som alle skal være i tilstand $|1>$ for at negere målqubit'en. I matrix notationen vil det sige, at gaten er identisk med identitetsmatricen, undtaget i nederste højre hjørne, hvor de sidste 4 elementer er skiftet ud med en σ_x matrice. Disse C^nNOT gates har Barenco et al. beskrevet i [1], og vi anvender deres konstruktionsanvisning. Det bør dog allerede her nævnes, at sættet $\{CNOT, P(2), H\}$ er komplet, jf. [7], hvilket vil sige, at de effektivt kan implementere enhver kvantealgoritme, uanset hvilke operationer denne måtte bruge.
For simplicitet vil jeg lade tidskompleksiteten af alle gates beskrevet i dette afsnit være $\Theta(1)$.

2.3 Kvantecomputeren ift. den klassiske turing maskine

I afsnit 2.1 blev det klart at der ikke ligger mere information i en qubit, end i en klassisk bit, idet vi, når vi måler tilstanden af kvantecomputeren, kun får én af de egentilstande, som kvantecomputerens tilstand evt. er en superposition af, som uddata. Hvori ligger en kvantecomputers styrke ift den klassiske turingmaskine så? Svaret er, at en kvantecomputer tillader os at bestemme egenskaber ved en given operation, svarende til en given funktion $f(x) : \{0,1\}^n \rightarrow \{0,1\}^m$, selvom vi ikke kan opnå at bestemme funktionens værdi for alle mulige inddata. En illustration af dette, som evt. kan tjene til at give læseren en intuition af, hvordan ovenstående kan bruges, gives i appendiks A, hvor vi vil analysere den simpleste kendte kvantealgoritme, nemlig Deutsch's algoritme.

3 Shor's algoritme

I dette afsnit vil vi se, hvorledes vi kan bruge foregående afsnits begreber til at konstruere et kvantenetværk, som kan faktorere et tal N, som kan skrives med n bit, i $\Theta(n^3)$, i det idealiserede tilfælde hvor der ikke er nogen former for støj i kvantesystemet. Grunden til at dette er interessant er specielt, at RSA-krypteringens sikkerhed bygger på, at man ikke har klassiske algoritmer som kan gøre dette i sub-exponentiel tid, hvorfor en kvantecomputer vil kunne bruges til at overvinde denne kryptering. Denne algoritme er først implementeret af Peter Shor i [12], men følgende gennemgang er inspireret af [2] afsnit 6.9 og 6.10. Hvor andre kilder er brugt, er det specificeret.
I appendix B argumenterer vi for at problemet at faktorere N klassisk kan omformes til at finde perioden af funktionen $f(x) = a^x \mod N$, hvor $a < N$ er et naturligt tal, hvorom det gælder at $SFD(a, N) = 1$ (SFD $\equiv$ Største Fælles Deviser). Emnet for dette afsnit vil derfor være, at konstruere en kvantealgoritme som kan finde perioden af $f(x) = a^x \mod N$, og det viser sig, at dette naturligt er en todelt opgave:

- Bestemmelse af Fourier transformationen af en funktion.
- Et netværk, som effektivt laver transformationen $|x> |0> \rightarrow |x> |a^x \mod N>$

Disse problemer blive behandlet i de to følgende underafsnit. Slutteligt (i afsnit 3.3) konkluderer jeg med at bevise, at Shor's algoritme faktisk kan implementeres i $\Theta(n^3)$.

3.1 Kvante Fourier transformation

Vi ønsker, jf. appendiks B, hurtigt at kunne finde perioden af funktionen $f(x) = a^x \mod N$, dvs. det tal r, som opfylder at $f(r) = a^r \mod N = 1$, hvor vi med "hurtigt" mener at tidskompleksiteten skal være polynomiel i n. Måden, vi vil gøre det på, er essentielt at foretage følgende skridt: Vi starter ud med to registre af størrelse n, et register til "inddata" og et til "uddata", begge initialiseret til $|0>$. Vi roterer så "inddata"-registret til tilstanden $\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i>$ (hvor vi husker fra afsnit 2 at $|i>$ er den egentilstand i beregningsbasen som svarer til at qubit'ene, læst som en binær vektor, er tallet i). Dette skridt er simpelt at implementere, vi anvender blot Hadamard transformationen H (se afsnit 2.2.1) på alle qubits, så vi opnår tilstanden

$$(\tfrac{1}{\sqrt{2}}(|0>_0 + |1>_0)) \otimes (\tfrac{1}{\sqrt{2}}(|0>_1 + |1>_1)) \otimes \ldots (\tfrac{1}{\sqrt{2}}(|0>_{n-1} + |1>_{n-1})) = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i> \tag{2}$$

Næste skridt er at beregne $f(x)$ af inddata registret, og placere resultatet i uddataregistret, dvs. anvende transformationen

$$(\frac{1}{\sqrt{2^n}} \sum_{i=0}^{n-1} |i>_{ind}) |0>_{ud} \rightarrow \frac{1}{\sqrt{2^n}} \sum_{i=0}^{n-1} (|i>_{ind} |f(i)>_{ud}) \tag{3}$$

Implementationen af dette skridt beskrives i næste afsnit, så vi vil her blot antage, at dette skridt er blevet udført.
Bemærk, at vi har en sammenfiltret tilstand: Hvis vi måler uddataregistret til $f(j)$ kollapser inddata til tilstandsunderrummet udspændt af de tilstande $|i>$, hvorom det gælder at $f(i) = f(j)$ - der er flere forskellige værdier af i, idet $f(x)$ jo er periodisk. Vi får ikke brug for uddataregistret i den videre beregning (og jeg vil derfor ikke medtage den i mine tilstandskets herefter), så faktisk kunne vi måle uddataregistret, for at få inddataregistret til at kollapse til et givet tilstandsunderrum, men da dette vil forøge antallet af nødvendige operationer i kvantenetværket vil vi undlade at gøre dette. Vi kan dog gennemføre beskrivelsen af algoritmen udfra den antagelse, at vi er i et givet tilstandsunderrum, idet det jo er nok at vide at vi kan finde perioden af $f(x)$ vha. tilstande fra et givet tilstandsunderrum i inddataregistret, så kan vi konkludere at vi vil kunne finde perioden udfra en hvilken som helst superposition af tilstande fra forskellige tilstandsunderrum (idet perioden af $f(x)$ selvfølgelig ikke afhænger af hvad for et tilstandsunderrum vi betragter).
Antag altså, at vi har specificeret at inddataregistret er i tilstanden $|x_0 + k*r>$, $k \in \{0, 1, \ldots, (\frac{2^n}{r}) - 1\}$, hvor r er perioden af $f(x)$, og $x_0 \in \{0, 1, \ldots, r-1\}$. dvs. vi er i tilstandsunderrummet specificeret ved x_0. Vi vil nu forsøge at finde r vha. en Fourier transformation af inddataregistret, og derefter måle tilstanden af inddataregistret. Der er to delproblemer: at finde en (hurtig) implementation af den sidste transformation, og bevise at vi efter en måling (med en hvis sandsynlighed) kan bruges til at beregne r. Disse vil blive behandlet hver for sig i de næste to underafsnit.

3.1.1 En implementation af Fourier transformationen på kvantecomputeren

Vi ønsker at implementere transformationen F så

$$\begin{aligned} F(\sum_x f(x)|x>) &= \sum_x F(f(x)|x>) = \\ &\sum_x \frac{1}{\sqrt{2^n}} \sum_y (e^{\frac{2*\pi*i*x*y}{2^n}} f(x)|y>) = \\ &\sum_y \frac{1}{\sqrt{2^n}} \sum_x (e^{\frac{2*\pi*i*x*y}{2^n}} f(x))|y> \end{aligned} \tag{4}$$

hvor vi har, at $f(x)$ er en delta kornecker funktion af x og $x_0 + k*r, k \in \{0, 1, \ldots, \frac{2^n}{r} - 1\}$, ganget med en normeringskonstant $\frac{1}{\sqrt{a}}$ hvor a er antallet perioder af $f(x)$. For at løse dette problem vil vi først lave den observation, at, hvis vi betragter x og y som bitstrenge, hhv $x_{n-1}, x_{n-2}, \ldots, x_0$ og $y_{n-1}, y_{n-2}, \ldots, y_0$ hvor x_m hhv. y_m er koefficienten på 2^m i den binære repræsentation af x og y, så kan vi ignorere alle led i produktet af de to bitstrenge $(x_i * y_j)$ hvor $i+j \geq n$, idet disse led vil have en koefficient 2^{i+j} som 2^n går op i. Dermed vil $\frac{x_i*y_j*2^{i+j}}{2^n} \in \mathbb{N}$ og $e^{\frac{2*\pi*i*x_i*y_j*2^{i+j}}{2^n}} = (e^{2*\pi*i})^{\frac{2^{i+j}*x_i*y_j}{2^n}} = 1^{\frac{x_i*y_j*2^{i+j}}{2^n}} = 1$. Dvs., idet vi lader summen over x i formel 4 løbe over værdierne som opfylder $x \mod k = x_0$ (dette er

korrekt da $f(x) = 0$ for alle andre værdier af x) og dernæst erstatter $f(x)$ med $\frac{1}{\sqrt{a}}$ (korrekt, idet vi netop har de værdier af x tilbage, hvor dette er sandt), får vi, at vi skal implementere

$$\begin{aligned}
&\frac{1}{\sqrt{a}}F|x>= \frac{1}{\sqrt{a}}\frac{1}{\sqrt{2^n}}\sum_y (e^{\frac{2*\pi*i*x*y}{2^n}})|y>= \\
&\quad \frac{1}{\sqrt{a}}\frac{1}{\sqrt{2^n}}\sum_y (e^{\frac{2*\pi*i*\sum_{j=0}^{n-1}(2^j*y_j*\sum_{l=0}^{n-j-1}2^l*x_l)}{2^n}})|y_{n-1}y_{n-2}\cdots y_0>= \\
&\quad \frac{1}{\sqrt{a}}\frac{1}{\sqrt{2^n}}\sum_y (\prod_{j=0}^{n-1}\prod_{l=0}^{n-j-1} e^{\frac{2*\pi*i*(y_j*x_l*2^{j+l})}{2^n}}|y_j>) = \\
&\quad \frac{1}{\sqrt{a}}\frac{1}{\sqrt{2^n}}(\prod_{j=0}^{n-1}\sum_{y_j=0}^{1}\prod_{l=0}^{n-j-1} e^{\frac{2*\pi*i*(y_j*x_l)}{2^{n-j-l}}}|y_j>)
\end{aligned} \tag{5}$$

hvor det første produkt i de sidste to linjer skal forstås som et tensor produkt, idet det løber over y's index. Denne transformation kan uden videre implementeres effektivt på en kvantecomputer:

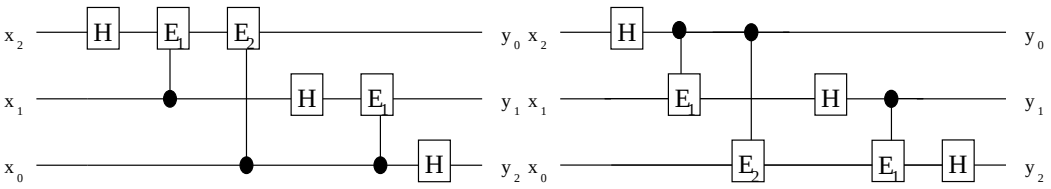
- Tensorproduktet evaluerer til at vi skal udføre operationen på hver qubit.
- Summen over y_j genkender vi som en Hadamard transformation på x_{n-j} idet exponenten for $y_j = 0$ altid (dvs. for alle faktorer af produktet over l) er 0 og for $y_j = 1$ er den sidste faktor af produktet over l netop $e^{\frac{2*\pi*i*x_l}{2^{n-j-(n-j-1)}}} = e^{\pi*i*x_l} = (-1)^{x_l}$
- De sidste $n-j-1$ faktorer af produktet over l kan hver implementeres som en transformation af formen $\begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i*\pi}{2^d}} \end{bmatrix}$, hvor d er defineret som $n-j-l$, hvis konstruktionen vi har beskrevet i 2.2.1. Nedenfor er en procedure beskrevet som (i vores tilfælde) kan afskaffe behovet for en kontrolleret version af faseforskydningsoperationen, og vi vil derfor ikke gå videre ind i konstruktionen af en sådan, men blot nævne at metoden som er beskrevet i [1] afsnit V ville kunne anvendes.

I næste afsnit skal vi se, at vi for at finde perioden af $f(x)$ skal måle tilstanden af kvanteregistret umiddelbart efter Fourier transformationen, og vi vil derfor anvende en optimering som foreslået i [2] kap 6 s. 74: eftersom en kontrolleret faseforskydningsoperation har følgende matrix repræsentation

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i*e^{\frac{i*\pi}{2^d}}} \end{bmatrix}$$

, ser vi, at den er symmetrisk under ombytning af kontrol og mål qubit. Vi kan derfor vælge at anskue implementationen der har hadamard transformationen

som den *sidste* transformation af hver enkelt qubit, istedet for den *første*, og samtidigt observere at alle kontrol qubits for alle kontrollerede faseforskydnings gates i netværket er "færdige" i den forstand, at de ikke længere ændres af det resterende netværk - den nye anskuelse af netværket er i figur 3 til højre. Optimeringen af netværket er nu, at vi, istedet for at anvende en hadamard transformeret qubit som kontrol qubit i en kontrolleret faseforskydnings gate, blot måler dens tilstand, og derefter anvender de rette faseforskydnings gates på de resterende qubits hvis qubittens tilstand var $|1>$, og lader være hvis tilstanden var $|0>$. På denne måde kan vi fuldstændigt eliminere alle kontrollerede gates fra implementationen.



Figur 3: Til venstre er det umiddelbare netværk for kvante Fourier transformationen. Til højre er det ækvivalente netværk, som kan optimeres hvis der foretages en måling af tilstanden umiddelbart efter netværket er gennemløbet

3.1.2 Bestemmelse af perioden på $a^x \mod N$

Vi har nu set hvordan en kvantecomputer hurtigt kan Fourier transformere funktionen $f(x) = a^x \mod N$, og vi påstår nu, at dette kan bruges til at bestemme ordnen af $f(x)$ med en sandsynlighed på $\frac{4}{\pi^2}$. Husk, at vi har en tilstand på formen $\frac{1}{\sqrt{a}} * \sum_{k=0}^{a-1} |x_0 + k * r >_{ind} |f(x_0) >_{ud}$, idet vi ser på situationen hvor $f(x_0)$ er bestemt til noget fast for at forsimple beregningerne (jf. ovenfor), og husker at a er antallet af perioder som $f(x)$ når at gennemløbe når x gennemløber sine værdier. Vi anvender nu Fourier transformation på inddataregistret som dermed bliver

$$\frac{1}{\sqrt{a * 2^n}} * \sum_{y=0}^{2^{n-1}} \sum_{k=0}^{a-1} (e^{\frac{2*\pi*i*y*(x_0+k*r)}{2^n}} |y>) =$$

$$\frac{1}{\sqrt{a * 2^n}} * \sum_{y=0}^{2^{n-1}} (e^{\frac{2*\pi*i*y*x_0}{2^n}}) \sum_{k=0}^{a-1} (e^{\frac{2*\pi*i*y*k*r}{2^n}} |y>) \quad (6)$$

Idet vi ønsker at evaluere sandsynligheden for at måle visse værdier af y, opstiller vi nu sandsynligheden for at måle en givet værdi af y, hvilket, jf. afsnit

2.1, er normkvadratet af koefficienten for denne værdi af y, dvs:

$$|\frac{1}{\sqrt{a*2^n}}*(e^{\frac{2*\pi*i*y*x_0}{2^n}})\sum_{k=0}^{a-1}(e^{\frac{2*\pi*i*y*k*r}{2^n}})|^2=\frac{a}{2^n}|\frac{1}{a}\sum_{k=0}^{a-1}e^{\frac{2*\pi*i*y*k*r}{2^n}}|^2= \tag{7}$$

$$\frac{a}{2^n}|\frac{1}{a}\sum_{k=0}^{a-1}e^{(2*\pi*i)*p}*e^{\frac{2*\pi*i*k*(y*r \mod 2^n)}{2^n}}|^2=\frac{a}{2^n}|\frac{1}{a}\sum_{k=0}^{a-1}e^{\frac{2*\pi*i*k*(y*r \mod 2^n)}{2^n}}|^2=$$

$$\frac{a}{2^n}|\frac{1}{a}\sum_{k=0}^{a-1}e^{i*k*\phi_y}|^2$$

, hvor p er et heltal, og $\phi_y \equiv .\frac{2*\pi*((y*r) \mod 2^n)}{2^n}$

Det, vi reelt måler, er værdien af y, og vi vil nu vise følgende sætning

Sætning 1 *Sandsynligheden for, at måle en værdi af y som opfylder*

$$\frac{c}{r}-\frac{1}{2*2^n}\leq\frac{y*r}{2^n}\leq\frac{c}{r}+\frac{1}{2*2^n}\leftrightarrow c*\frac{2^n}{r}-\frac{1}{2}\leq y\leq c*\frac{2^n}{r}+\frac{1}{2} \tag{8}$$

er begrænset nedad af $\frac{4}{\pi^2}$*, hvor c er et heltal fra* 0 *til* $r-1$

Bevis: At y opfylder (8) er ækvivalent med, at $-\frac{r}{2}\leq y*r \mod (2^n)\leq\frac{r}{2}$. Idet vi nu husker på at y løber fra 0 til 2^n-1, ser vi, at der er netop r værdier af y som opfylder denne betingelse: De 2^n værdier af $y*r$ har jo afstanden r med hindanden, hvorfor der, for hvert multiplum af 2^n op til $r*2^n$, netop er én værdi af $y*r$ som er inden for afstanden $\frac{r}{2}$ af hvert af multiplaene af 2^n, og dem er der netop r af. Lad os bemærke, at for disse værdier af y er exponenten i (7) underlagt $-\pi*\frac{r}{2^n}\leq\phi_y\leq\pi*\frac{r}{2^n}$. Vi ønsker nu at evaluere (7) for disse værdier af ϕ_y:

$$\frac{a}{2^n}|\frac{1}{a}\sum_{k=0}^{a-1}e^{\phi_y*i*k}|^2=\frac{a}{2^n}|\frac{1}{a}\frac{e^{i*a*\phi_y}-1}{e^{i*\phi_y}-1}|^2=\frac{a}{2^n}|\frac{1}{a}|^2*|\frac{e^{i*a*\phi_y}-1}{e^{i*\phi_y}-1}|^2 \tag{9}$$

For at evaluere dette udtryk, vil det være nyttigt at observere at hvis $B*|\theta|\leq\pi$ gælder der at

$$|e^{i*B*\theta}-1|\geq\frac{2B|\theta|}{\pi}\leftrightarrow|e^{i*B*\theta}-1|-\frac{2B|\theta|}{\pi}\geq 0 \tag{10}$$

For at bevise dette vil vi først indse, at funktionen $|e^{i*B*\theta}-1|-\frac{2B|\theta|}{\pi}$ har tre nulpunkter i intervallet $-\pi\leq B|\theta|\leq\pi$, nemlig $B|\theta|=-\pi$, $B|\theta|=0$ og $B|\theta|=\pi$: Det ses ved inspektion af disse punkter er nulpunkter, og idet funktionens anden afledede er $-B^2$, ser vi, at på hver side af punktet $B*|\theta|=0$, hvor funktionen ikke er differentiabel, kan funktionen maximalt have 2 nulpunkter. Idet funktionen er kontinuert og har nulpunkt i $B*|\theta|=0$ har den maximalt 3 nulpunkter. Idet den anden afledte af funktionen er negativ er det ønskede vist, idet funktionen så må være ikke negativ mellem dens nulpunkter, dvs. for

værdier af $B|\theta|$ som ligger i intervallerne $[-\pi, 0]$ og $[0, \pi]$, tilsammen $[-\pi, \pi]$. Lad os nu vende tilbage til det udtryk, som vi oprindeligt forsøgte at evaluere:

$$\frac{a}{2^n}|\frac{1}{a}|^2 * |\frac{e^{i*a*\phi_y}-1}{e^{i*\phi_y}-1}|^2 = \frac{a}{2^n}|\frac{1}{a}|^2 * |\frac{e^{i*(a-1)*\phi_y}-1}{e^{i*\phi_y}-1} + e^{i*\phi_y}|^2 \geq$$
$$\frac{a}{2^n}|\frac{1}{a}|^2 * (\frac{|e^{i*(a-1)*\phi_y}-1|^2}{|e^{i*\phi_y}-1|^2} - 1)^2 \quad (11)$$

hvor vi i sidste linje har brugt trekantsuligheden. Idet vi har at $|e^{i*\theta}-1| \leq |\theta|$ (længden $|\theta|$ langs enhedscirklen er længere end den rette linje fra 1 til $e^{i*\theta}$), og idet vi husker fra (8) at $(a+1)*\phi_y \leq \pi$, kan vi nu anvende (10) med $B = a+1$ til at konkludere:

$$\frac{a}{2^n}|\frac{1}{a}(|^2 * \frac{|e^{i*(a-1)*\phi_y}-1|}{|e^{i*\phi_y}-1|} - 1)^2 \geq \frac{a}{2^n}|\frac{1}{a}|^2(\frac{4*(a-1)^2*\|\phi_y|^2}{\pi^2*|\phi_y|^2} - 1) =$$
$$\frac{a}{2^n}\frac{1}{a^2}\frac{4}{\pi^2} * (a^2 - 2a + 1 - \frac{\pi^2}{4}) \quad (12)$$

Lad os igen benytte at har ca. $2^{\frac{n}{2}}$ perioder, dvs. $a \gg 1$, hvorfor vi kan se bort fra alle andre led end det, der har den største potens af a, og samtidigt ved vi fra definitionerne af a og r, at $\frac{1}{r} \simeq \frac{a}{2^n}$, hvorfor vi finder at sandsynligheden for at måle netop en af de værdier af y, og dermed ϕ_y, som opfylder (8) er $\frac{1}{r} * \frac{4}{\pi^2}$. Men da der er netop r forskellige værdier af y som opfylder (8) er det klart at sandsynligheden for at måle én af dem er $\frac{4}{\pi^2}$, hvilket var det, vi postulerede. □

8 gør os istand til at finde $\frac{2^n}{r} * c$, forudsat at perioden (r) er væsentligt mindre end 2^n - Dette krav er ikke så overraskende: Hvis vi skal gøre os håb om, at en fouriertransformation skal give et fornuftigt resultat, må vi kræve, at funktionen faktisk er periodisk inden for det interval vi kan få oplysninger om. Lad os nemlig antage at $r^2 \leq 2^n$. Så er afstanden mellem to forskellige, potentielle værdier af $c * \frac{2^n}{r}$ nedadtil begrænset af $\frac{1}{2^n}$, idet $\frac{\alpha}{r} - \frac{\beta}{r} = \frac{r*(\alpha+\beta)}{r^2} \geq \frac{r*(\alpha+\beta)}{2^n}$. Hvis vi nu antager at vi har målt en værdi af $\frac{y}{2^n}$ som opfylder (8), har vi en entydigt bestemt værdi af $\frac{c}{r}$, idet der netop er en værdi af $\frac{c}{r}$ som ligger i intervallet $\left[\frac{y}{2^n} - \frac{1}{2*2^n}, \frac{y}{2^n} + \frac{1}{2*2^n}\right]$, der jo har længden $\frac{1}{2^n}$. Hvis $SFD(c, r) = 1$ (dvs. hvis c og r er coprime) så har vi nu fundet r - denne værdi kan kontrolleres ved at beregne $f(x+r)$(evt. på kvantenetværket specificeret i afsnit 3.2) og se om den er lig $f(x)$. Hvis den ikke er det kan det skyldes to ting:

1. Den målte værdi af y opfyldte ikke (8). Vi kan her prøve forskellige værdier af y tæt på den målte - det kan være vi var tæt på, uden faktisk at opfylde (8).

2. c og r var ikke coprime. I dette tilfælde finder vi reelt en faktor af r, men det er sandsynligt at $SFD(c, r)$ ikke er særligt stor, hvorfor vi kunne prøve forskellige små multipla af r.

Hvis disse forsøg ikke giver et tilfredsstillende resultat, kan vi køre algoritmen en gang til. Dette vil føre til et resultat, som med rimelig sandsynlighed ($\frac{4}{\pi^2}$) vil opfylde (8). Hvis denne værdi heller ikke umiddelbart giver værdien af r, har vi nu to værdier $\frac{c_1}{r}$ og $\frac{c_2}{r}$, og under antagelse af at de begge kommer fra værdier af

y som opfylder (8), kan vi nu bestemme r som mindste fælles multiplum af de to forslag til en værdi af r som var resultatet af de to kørsler (r_1 og r_2), *forudsat* at c_1 og c_2 er coprime. Da værdierne for c stort set er jævnt fordelt mellem 0 og $r-1$, evalueres sandsynligheden for at to tilfældigt valgte tal er coprime: At c_1 og c_2 er coprime er det samme som at der ikke findes noget primtal, som indgår i begge tals primtalsopløsning, dvs. der er intet primtal p som går op i både c_1 og c_2. Da et primtal p kun går op i alle multipla af p, går det op i $\frac{1}{p}$ af alle tal, og således er sandsynligheden for at et *givet* primtal p går op i *både* c_1 og c_2 $(\frac{1}{p})^2$. Vi kan nu udtrykke sandsynligheden for, at c_1 og c_2 er coprime som produktet af $1-\frac{1}{p^2}$ over alle primtal p: $\prod_{\text{p et primtal}} 1-\frac{1}{p^2} = \frac{6}{\pi^2}$, et resultat som er kendt fra talteori.

3.2 Implementationen af $a^x \mod N$

Resultatet i foregående afsnit (at man kan Fourier transformere en funktion $f : 0,1^n \rightarrow 0,1^n$ i Poly(n) tid) er et af de mest lovende kvantealgoritmer som eksisterer, pga. Fourier transformationens store anvendelighed. Men der er et krav, som vi endnu ikke har redegjort for, nemlig at funktionen $f(x)$ (i vores tilfælde $a^x \mod N$) skal kunne beregnes hurtigt, dvs. i Poly(n) tid, på en kvantecomputer, da vi, for at gennemføre Fourier transformationen, beregner $|f(x)>$, hvor $|x>$ er i en superposistion af alle mulige værdier. Dette er grunden til at algoritmen *skal* udføres på en kvantecomputer. I dette afsnit vil vi konstruere et netværk (først givet i [3]), som gør netop dette. Lad os dog først analysere problemstilligen lidt grundigere:

- Selv om algoritmen som sagt skal implementeres på en kvantecomputer, er den ikke probabilistisk i den forstand, at hvis $|x>$ er en egenfunktion (i modsætning til en superposistion) i beregningsbasen, så er $|f(x)>$ det også.

- Det er *kun* x der kan være i en superposistion af værdier - a og N er almindelige, klassiske tal. Dette kan vi udnytte til at konstruere et mindre og dermed hurtigere kvantenetværk, hvis vi er villige til at bruge en begrænset mængde ressourcer på en *klassisk* computer til at beregne netværkets opbygning, udfra tallene a og N. Idet klassiske computere formentligt i lang tid fremover vil være hurtigere en kvantecomputere (dvs. kunne behandle flere instruktioner pr. tidsenhed), vil dette være en idé som vi her vil benytte os af. Det skal dog selvfølgeligt godtgøres, at enhver klassisk algoritme vi måtte ønske at bruge, kan køre i Poly(n) tid.

- Vi bliver nødt til at have nogle "overskydende" qubits, idet $a^x \mod N$ ikke er invertibel (vi forsøger jo netop at finde dens periode), hvorfor man ikke kan implementere $|x> \rightarrow |f(x)>$, men vi er nødt til (jf. afsnit 2.2) at implementere $|x>_{ind} |0>_{ud} \rightarrow |x>_{ind} |f(x)>_{ud}$. Desuden er det inddataregistret vi arbejder videre med i kvantefouriertransformationen, hvorfor det også derfor er nødvendigt at beholde dette register.

- I afsnit 3.1 blev det forudsat at $a^x \mod N$ var gennemløb tilstrækkeligt mange perioder når x gennemløb værdierne dens værdier. Dette sikrer vi nu ved at lade inddata-registret have $2n$ qubits, således at der i alt fald gennemløbes 2^n perioder af funktionen.

Vi ser nu, jf. det sidste punkt, at vi har brug for i hvert fald $3n$ qubits for at lave faktorere et n-bit tal - $2n$ til $|x>$ og n til $|f(x)>$. Antallet af qubits er vigtigt af to grunde: Dels er antallet af qubits det i øjeblikket en af de begrænsende parametre for implementationen af en fysisk kvantecomputer, og dels indebærer flere qubits en større risiko for fejl pga. støj, hvad vi skal se nærmere på i afsnit 4. Til gengæld vil tidskompleksisteten af algoritmen blive større hvis vi forsøger at reducere antallet af nødvendige qubits, igen med konsekvenser for risikoen for fejl pga. støj, idet netværket skal køre i længere tid, og dermed vil have større risiko for at bliver ramt af uoprettelige fejl pga. støj.
Der eksisterer en række forskellige netværk som implementerer den ønskede funktion, men den algoritme som vi vil anvende udgør et godt kompromis mellem de to grundlæggende parametre: tidskompleksitet og hukommelsesforbrug, idet den med $5n + 1$ qubits implementerer en algoritme som kører i $\Theta(n^3)$ tid.

Del og hersk metoden anvendt på $a^x \mod N$

Vi vil i det følgende bryde problemstillingen ned i mindre dele, indtil vi kan implementere enkeltdelene nemt, mere specifikt vil vi transformere (exponential funktion modulo N) $\rightarrow$ (kontrolleret multiplikator funktion modulo N) $\rightarrow$ (addition modulo N) $\rightarrow$ (addition og sammenligning), hvor "kotrolleret" skal forstås på samme måde som $CNOT$ gaten i 2, dvs. en kontrol qubit's tilstand kontrollerer hvorvidt multiplikationen skal foretages, eller ej.
Grundlæggende anvender alle kendte kvantenetværk til beregning af $a^x \mod N$ at $a^x \mod N$ kan omskrives til en sum af potenser af a: $a^x \mod N = \prod_{i=0}^{2n-1}(a^{(2^i)x_i} \mod N)$ hvor vi har anvendt den binære repræsentation af x, $x = \sum_{i=0}^{2n-1}(2^i x_i), x_i = \{0, 1\}$. Idet a er en almindeligt klassisk bitvektor, er det nemt at bestemme a^{2^i} på forhånd, hvorfor vi nu har transformeret problemet at lave en exponentialfunktion, modulo N, om til en multiplikator funktion, modulo N, som dog skal kontrolleres af x_i'erne.
Den multiplikation vi ønsker at foretage er imellem en klassisk binært tal (a^{2^j}) og en superposition af forskellige tal ($z \mod N \equiv \prod_{i=0}^{j-1}(a^{(2^i)x_i} \mod N)$, idet x_i'erne er i en superposition af $|0>$ og $|1>$. Vi anvender nu samme trick som ovenfor, dvs. vi opskriver z i dens binære repræsentation, $\sum_{k=0}^{n-1}(2^k z_k)$, hvorefter vi kan skrive $a^{2^j} z \mod N = \sum_{k=0}^{n-1}(z_k 2^k a^{2^j} \mod N)$. Dette genkender vi som en kontrolleret addition modulo N, hvor z_k fungerer som kontrol-qubit.
Vi ser, at den kontrollerede addition modulo N kommer til at foregå mellem to tal, som begge er mindre end N, nemlig $b \equiv \sum_{k=0}^{l-1}(z_k 2^k a^{2^j} \mod N$ og $c \equiv 2^l a^{2^j} \mod N$. Dette kan vi udnytte til at implementere "modulo N" delen nemt ved hjælp af en sammenligning ($b + c \geq N$) og, på bagrund af resultatet af denne, addere b og c (hvis $b+c < N$) eller b og $c-N$ (hvis $b+c \geq N$). Her skal vi huske på, at vi på forhånd kan lave beregne værdierne for c og $c - N$ - de afhænger udelukkende af klassiske tal (dvs. ikke af $|x>$). Vi kan derfor implementere

addition modulo N ved at lave en multiplexet addition, som, baseret på resultatet af sammenligningen, adderer enten c eller $c - N$. Denne multiplexede addition viser sig at være relativt nem at implementere (tildels fordi vi adderer et "klassisk tal", som ikke kan være i en superposition), hvilket vi vil gøre i afsnit 3.2.1.
Dermed er den sidste komponent som vi vil analysere sammenligningen imellem tallene $b+c$ og N, eller rettere imellem b og $N-c$ - fordelen ved den sidste sammenligning af tallet b og et tal, som kan beregnes på forhånd, idet det er et klassisk tal som kun har a og N som parametre (c kan bestemmes fra a). Implementationen af dette komponent er en af de ting, som varierer imellem de forskellige implementationer - grundlæggende kan man enten foretage en subtraktion mellem de to tal, og så anvende den sidste carry-qubit som indikator for hvorvidt b var større eller mindre end $N-c$, nøjagtigt som på en klassisk arkitektur. Implementationen af en subtraktion er enkel, idet den er den inverse funktion af additionen, dvs. hvis vi har implementeret $|\alpha> |\beta> \rightarrow |\alpha> |\alpha+\beta>$, så kan vi blot gennemløbe denne transformation baglæns og opnå $|\alpha> |\gamma> \rightarrow |\alpha> |\gamma-\alpha>$. Dette er den metode som bliver anvendt i [4]. En anden metode (som bliver anvendt i [3] og som vi vil benytte) benytter sig af $N - c$'s klassiske natur: Det er klart at en sammenligning kan udføres qubit for qubit, hvis vi starter fra den mest betydende qubit, idet den første qubit som er forskellig helt vil afgøre sammenligningen. Vi kan dermed, udfra vort kendskab til $N - c$, lave et netværk som sammenligner de to tal udelukkende ved at se på b.

3.2.1 De basale komponenter i $a^x \mod N$

I det forgående har vi brudt problemstillingen ned til at implementere en sammenligning og en multiplexet addition. Vi vil nu overveje netværk for disse operationer, som kun anvender de basale operationer som blev specificeret i afsnit 2.2.1. Sammenligningen er den mest enkle: Den kan vi implementere den udelukkende ved brug af C_kNOT gates, da vi kan betragte sammenligning af to tal som sammenligning mellem deres binære repræsentationer, *qubit for qubit*.
Under henvisning til [3] afsnit VC, ser vi, at vi stort set kan opbygge netværket af to forskellige byggeklodser, en for $(c - N)_k = 0$ og en for $(c - N)_k = 1$ (hvor $c - N$ er den klassiske operand i sammenligningen). De to byggeklodser kan ses i produktet i (5.23) i [3], hvorfra det umiddelbart ses, at de kan implementeres vha. k-kontrollerede not-gates. Det eneste udover disse byggeklodser i netværket er den første og den sidste qubit-sammenligning: Den første har ikke nogen kontrol-qubit fra den forrige sammenligning, da der ikke *er* nogen forrig sammenligning, og tilsvarende sætter den sidste sammenligning ikke nogen kontrolqubit til den næste sammenligning, idet en sådan ikke eksisterer. Idet vi husker på, at vi ønsker at vide om $b \geq N - c$ (og ikke $b > N - c$) er det klart at hvis den sidste bit i $N - c$ er 0, så er den sidste sammenligning triviel: Uanset om den sidste qubit i b er 0 eller 1, vil $b \geq N-c$. Efter sammenligningen vil vi have en uheldig tilstand, dels fordi b bliver ændret i netværket (stort set negeret - muligvis (hvis $(N-c)_0 = 0$) er den mindst betydende qubit uændret), og dels fordi vi har en del "overskydende" information i vore kontrolqubits. Hvis vi ønsker at genbruge kontrolqubittene bliver vi nødt til at transformere

dem tilbage til $|0>$ tilstanden: Vi kan ikke (som i det klassiske tilfælde) slette indholdet, idet dette naturligvis ikke er reversibelt. Det viser sig at det samme trick (oprindeligt formuleret af Bennett, her taget fra [3]) kan løse begge problemerne: Vi udfører sammenligningen, kopierer resultatet til et uddata register, og kører sammenligningen baglæns.
Vi har således vist hvordan vi kan sammenligne to tal, og mangler nu kun at lave den kontrollerede, multiplexede addition for at have løst opgaven. Vi vil lave en addition af to n (qu)bit tal ved at lave qubit-vis addition og beregne en mente-qubit i hvert trin. Ligesom ved sammenligningen vil vi vælge mellem forskellige byggeklodser udfra vores kendskab til den klassiske operand. Blot har vi her 4 tilfælde, idet vi har to potentielle operander, nemlig c og $N - c$, og vi vælger imellem disse to værdier baseret på $|x>$, dvs. en kvantemekanisk superposition. Derfor er vi nødt til at have fire byggeklodser, idet hver af de to operander jo kan have værdierne 0 eller 1. For at spare plads vil jeg ikke inkludere en figur for disse 4 komponenter, men blot henvise til figur 7 i [3].

3.3 Tidskompleksitet af det grundlæggende netværk

Vi vil nu i dette afsnit kort argumentere for at Shor's algoritme - som den er præsenteret ovenfor - faktisk er $\Theta(n^3)$. Dette gøres principielt i to skridt:

1. Argumentation for, at de klassiske beregninger, som indgår i algoritmen kan gennemføres i $poly(n)$
2. Argumentation for, at selve kvantenetværket forløber i $\Theta(n^3)$, givet at de basale instruktioner fra afsnit 2.2.1 kan implementeres i $\Theta(1)$

Man kunne forestille sig, at vi også skulle ønske at de klassiske dele af algoritmen skulle forløbe i $\Theta(n^3)$ - strengt taget ville dette også være nødvendigt. Indtil videre er det dog ikke vigtigt, idet hastigheden af klassiske computere er omtrent en million gange større end de kvantecomputere der kan implementeres i en overskuelig fremtid. Dette vil muligvis ændre sig i fremtiden, og så vil man være nødt til at gennemføre analysen af de klassiske algoritmer.
I vores tilfælde ser vi, at det eneste vi anvender klassiske computere til er:

- At gennemløbe Euklids algoritme
- At vælge, baseret på en kvantecomputers måling, hvilket kvantenetværk kvantecomputeren skal gennemløbe (f.eks. hvorvidt vi i QFT skal anvende en faseskiftsgate eller ej).
- At konstruere et kvantenetværk udfra en klassisk bitvektor af længde $\Theta(n)$ (hvilket vi anvender i netværket for $a^x \mod N$)

Idet Euklids algoritme vides at køre i $O(n^2)$ tid (jf. [11], afsnit 31.2), og idet valgene baseret på både målinger fra kvantecomputeren og de klassiske bitvektorer må antages at kunne forløbe i konstant tid, idet de hver især, som inddata aldrig har mere end en enkelt (qu)bit, ser vi kravet om de klassiske algoritmers polynominelle tidskompleksitet er opfyldt.
Idet vi husker, at vi overordnet først skal anvende netværket for at beregne

$a^x \mod N$ én gang, og derefter anvende kvantenetværket som udfører QFT én gang. Dermed er tidskompleksiteten af det samlede netværk, idet vi betegner tidskompleksiteten af QFT og $a^x \mod N$ netværket som hhv. $\Theta(QFT)$ og $\Theta(a^x \mod N)$, lig med $\Theta(QFT) + \Theta(a^x \mod N)$.
QFT ses relativt enkelt at forløbe i $\Theta(n^2)$:
Størrelsen af input er $\Theta(n)$. For hver qubit skal anvendes en hadamard transformation, som er basal og dermed $\Theta(1)$. Desuden anvendes der, for hver qubit, i værste tilfælde et antal faseforkydninger (som hver især er basale, dvs. $\Theta(1)$) som svarer til antallet af qubits minus 1, dvs. $\Theta(n)$ faseforskydningsoperationer pr qubit, og idet vi i værste tilfælde skal gøre det for hver qubit, ser vi at vi ialt skal anvende $\Theta(n)\Theta(n) = \Theta(n^2)$ faseforskydninger. Da de eneste operatorer vi anvender i QFT er hadamard operatorer og faseforskydningsoperatorer, ser vi at $\Theta(QFT) = \Theta(n) + \Theta(n^2) = \Theta(n^2)$.
Dernæst vil vi undersøge netværket som bestemmer $a^x \mod N$:
I afsnit 3.2 argumenterede vi for, at hvis x har en længde som er $\Theta(n)$, så kan at $a^x \mod N$ implementeres af $\Theta(n)$ multiplikationer. Disse multiplikationer kan hver implementeres af $\Theta(n)$ additioner, som igen kan implementeres af $\Theta(n)$ sammenligninger og $\Theta(n)$ enkeltqubit additioner, som jf. afsnit 3.2.1, kan implementeres med et fast antal basale operationer, dvs. i $\Theta(1)$. Så er det ligetil at beregne tidskompleksiteten af $a^x \mod N$: $\Theta(a^x \mod N) = \Theta(n)\Theta(n)\Theta(n)\Theta(1) = \Theta(n^3)$.
Således er tidskompleksiteten af kvantenetværket for Shor's algoritme $\Theta(n^2) + \Theta(n^3) = \Theta(n^3)$.

4 Støj, fejlretning og fejltollerant beregninger

I det foregående afsnit blev der argumenteret for at Shor's algoritme fungerer korrekt i det tilfælde, at tilstanden og operationerne på kvantecomputeren er kan kontrolleres fuldstændigt. Dette er ikke fysisk gennemførligt: I en implementeret kvantecomputer vil der altid være en vis risiko for fejl, dvs. at den fysiske tilstand af kvantecomputerens hukommelse vil være en smule anderledes end den abstrakte teori ville forudsige, pga. uønsket interaktion med omgivelserne. Dette er for så vidt generelt for enhver behandling af information på fysiske systemer, dvs. også klassiske computere. Der er dog kvalitativt større problemer med disse effekter (som kaldes "støj" i det følgende), som vi vil beskrive og undersøge i afsnit 4.1, hvor vi også vil definere hvad vi præcist mener med støj. I afsnit 4.2 vil vi give metoder til at stabilisere hukommelsen i et fysisk system med støj, og vi vil udvikle og demonstrere en speciel kode som, indenfor visse forudsætninger, kan siges at være optimal. Dette er et relativt avanceret emne rent matematisk, men den grundlæggende idé er at sprede informationen på mere end en qubit, og til gengæld opnå en sikring af informationen. At vi dermed ikke længere opererer med enkelte qubit, men med blokke af fysiske qubits, som repræsenterer en enkelt logisk qubit, har som konsekvens, at vi er nødt til at ændre de basale operationer fra at operere på enkelte qubits til at opererer på disse blokke af qubits. Dette er emnet for afsnit 4.3.
De første to afsnit, 4.1 og 4.2 er primært inspirerede af [2] kapitel 7 (specielt afsnit 7.1-7.3 og 7.9 og 7.10), mens afsnit 4.3 repræsenterer en anvendelse af flere forskellige kilder, dog primært [6] og [5]

4.1 Grundlæggende diskussion

De to problemer, som kvalitativt forværrer situationen på en kvantecomputer ift. en klassisk computer er

1. Hukommelsen af kvantecomputeren er essentielt kontinuert, dvs. koefficienterne på hver af egentilstandende er kontinuerte variable, i modsætning til en klassisk computer, hvor hukommelsens værdi kan specificeres som én af et diskret sæt af tilstande. Dette gør at støj giver anledning til en kontinuert mængde af fejltilstande, i modsætning til det klassiske tilfælde.

2. Måling af hukommelsen i kvantecomputeren ændrer i sig selv hukommelsen. Dette betyder, at vi ikke (som i en klassisk computer) kan måle hukommelsen midt i beregningen for at kontrollere om dele af beregningen er gennemført korrekt, og derefter fortsætte beregningen.

Disse to problemer kan virke uoverskuelige, og har været grundlag for tvivl om, hvorvidt en kvantecomputer overhovedet kunne realiseres, men som vi skal se i dette afsnit, kan problemerne løses ved en kombination af indsigt i kvanteinformations natur og snedigt designede kvantenetværk. I de næste to underafsnit vil vi give den teoretiske opskrift på at løse hvert af de to problemer:

4.1.1 1. problem: kontinuerte tilstande

Løsningen af problemet med kontinuiteten af tilstande kan præsenteres i to grundlæggende skridt:

- Ekspander de kontinuerte fejl til en lineær funktion af diskrete fejl.
- Argumentation for, at dette tillader os at rette kontinuerte fejl, ved blot at rette disse diskrete fejl.

Omskrivning til diskrete fejl:

Grundlæggende kan alle fejl i kvantecomputeren beskrives som unitære operatorer som virker på et tensorprodukt af kvantecomputerens tilstand og "omgivelserne". I afsnit 1.3 blev det specificeret at vi vil se bort fra støj, som samtidigt giver anledning til fejl på flere qubits (dvs. korreleret støj), hvorfor vi kan beskrive den mest generelle støj i kvantecomputeren som en kobling mellem en given qubit i en generel tilstand, $\alpha|0> +\beta|1>$, og omgivelserne, $|o>$:

$$\begin{aligned} U((\alpha|0> +\beta|1>) \otimes |o>) = \alpha|0> \otimes|o_{00}> +\alpha|1> \otimes|o_{01}> + \\ \beta|0> \otimes|o_{10}> +\beta|1> \otimes|o_{11}> \end{aligned} \tag{13}$$

,hvor $|o_{xy}>$ er omgivelsernes tilstand i det tilfælde hvor qubitens tilstand er skiftet fra x til y. Det bør her bemærkes at omgivelsernes tilstande i (13) ikke (nødvendigvis) er orthonormale, og vi kan derfor *aldrig* skelne imellem dem, selv ikke hvis vi kunne måle omgivelsernes tilstand. Omskrivning af (13) giver

$$\begin{aligned} &= (\alpha|0> +\beta|1>) \otimes \tfrac{1}{2}(|o_{00}> +|o_{11}) \\ &+(\alpha|0> -\beta|1>) \otimes \tfrac{1}{2}(|o_{00}> -|o_{11}) \\ &+(\alpha|1> +\beta|0>) \otimes \tfrac{1}{2}(|o_{01}> +|o_{10}) \\ &+(\alpha|1> -\beta|0>) \otimes \tfrac{1}{2}(|o_{01}> -|o_{10}) \end{aligned} \tag{14}$$

Hvis vi nu husker på at $\sigma_x(\alpha|0> +\beta|1>) = \alpha|1> +\beta|0>$, $\sigma_y(\alpha|0> +\beta|1>) = i*(\alpha|1> -\beta|0>)$, $\sigma_z(\alpha|0> +\beta|1>) = \alpha|0> -\beta|1>$, og samtidigt definerer $|o_I>= \frac{1}{2}(|o_{00}> +|o_{11})$, $|o_X>= \frac{1}{2}(|o_{01}> +|o_{10})$, $|o_Y>= \frac{1}{2}(|o_{01}> -|o_{10})$ og $|o_Z>= \frac{1}{2}(|o_{00}> -|o_{11})$, ser vi at vi kan lave en ekspansion af den generelle fejloperator U til sættet $\{I, \sigma_x, \sigma_y, \sigma_z\}$, idet (14) nemlig kan nu kan skrives, idet vi tager den oprindelige venstreside med og definerer $|\Psi>= \alpha|0> +\beta|1>$:

$$\begin{aligned} U(|\Psi> \otimes|o_{før}) = I|\Psi> \otimes|o_I> +\sigma_x|\Psi> \otimes|o_X> + \\ i\sigma_y|\Psi> \otimes|o_y> +\sigma_y|\Psi> \otimes|o_y> \end{aligned} \tag{15}$$

For at følge den generelle konvention i artikler om kvantefejlretning, vil vi fra nu af definere $X \equiv \sigma_X, Y \equiv -i\sigma_y$ og $Z \equiv \sigma_z$, således at vi nu har udtrykt en generel fejloperator som en superposition af sættet $\{I, X, Y, Z\}$.

Korrektion af diskrete fejl medfører retning af kontinuerte fejl

Det er vigtigt at holde sig for øje at omskrivningen ovenfor netop kun er en omskrivning - vi har ikke løst problemet idet vi ikke har nogen måde at rette en superposition af fejl på. Dette kan dog let afhjælpes: Hvis vi antager at vi har en operator (Benævnt C), som kan måle hvorvidt en generel tilstand $|\Psi>$ er blevet påvirket af en X-fejl, dvs. C returnerer resultatet e_0 hvis der ikke er forekommet nogen fejl, og e_1 hvis der er forekommet en fejl, kan en måling af denne operator, og en korrektion baseret på denne måling, rette "kontinuerte" X-fejl. Lad os nemlig måle C på en generel tilstand, $aI|\Psi> + bX|\Psi>$: Den vil nu kollapse kollapse til enten $|\Psi>$ og vi vil få resultatet e_0, eller til tilstanden $X|\Psi>$, og vi vil få resultatet e_1. Hvis vi fik e_1 kan vi da blot anvende X på tilstanden, idet $X^2|\Psi> = I|\Psi> = |\Psi>$. Dermed er fejlen korrigeret for den kontinuerte X-fejl. Tilsvarende kan vi forestille os et netværk som kan rette en Z fejl, hvis en sådan skulle dukke op, ved samme teknik som for at rette X fejl, blot med X udskiftet med Z . Idet $Y = XZ$ er dette faktisk nok, en Y fejl kan betragtes (og dermed rettes) som en Z fejl efterfulgt af en X fejl. Vi kan dermed, hvis vi kombinerer de to netværk, som retter hhv. X og Z fejl, rette *superpositioner* af X,Y og Z fejl, og vi kan således rette *alle* kontinuerte fejl ved at rette to diskrete fejl, nemlig X og Z. Vi har endnu ikke fundet nogen måde at rette disse to fejl, men da dette kræver, at vi overvejer problemerne med at måle hukommelsens tilstand, er det derfor henlagt til afsnit 4.1.2.
Det er naturligt her at indføre nogle begreber som vi vil anvende i resten af denne rapport: Det er klart at eftersom vi kan skrive støj på én qubit op som en superposition af operatorerne I, X, Yog Z, så kan vi skrive støj på n qubits op som en superposition af fejloperatorer på formen $E = \{I, X, Y, Z\}^{\otimes n}$, hvor $^{\otimes n}$ skal forstås på den måde, at der til hver af de n qubits tilskrives netop en af operatorerne i sættet $\{I, X, Y, Z\}$, og støjen på alle n qubits repræsenteres da af tensorproduktet af disse n operatorer. Vi vil definere en fejloperator E's *vægt*, som antallet af enkeltqubit-operatorer i tensorproduktet, som er forskellige fra I, dvs. antallet af qubits som operatoren introducerer støj på.

4.1.2 2. problem: tilstandens kollaps

Det andet basale problem som optræder er, at man ikke kan måle tilstanden af hukommelsen uden at ødelægge evt. superpositioner i denne, og det er dermed et problem at finde ud af, hvorvidt der er indtruffet en fejl i hukommelsen. Måske ganske overraskende kan dette løses ved anvendelse af et begreb fra klassisk fejlretningsteori, nemlig paritetsmålinger. Vi vil herunder præsentere dette emne i tre punkter:

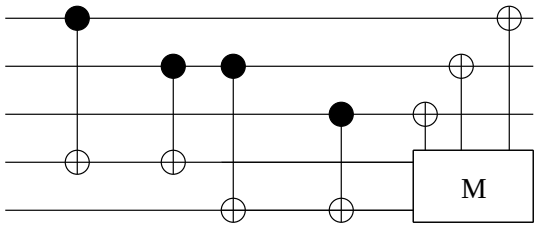
1. En måde at detektere og rette en enkelt X-fejl
2. En måde at detektere og rette en enkelt Z-fejl
3. Kombinationen af disse, nemlig den såkaldte Shor-kode, som var det første eksempel på en fejlretnings kode for kvantecomputere, idet vi jf. afsnit 4.1.1 kan rette enhver enkelt-qubit fejl, hvis vi kan rette både X- og Z-fejl på en enkelt qubit.

Korrektion af X-fejl

Den nemmeste måde at indse, hvorledes vi kan detektere og rette X-fejl uden at opnå information om tilstanden er ved et lille eksempel: Lad os for nemheds skyld antage at den qubit, vi ønsker at beskytte er i en egentilstand af Z, dvs. enten $|0>$ eller $|1>$. Det første vi gør, er at kopiere qubitten til to andre qubits vha. et par $CNOT$ gates - vi har nu altså enten $|000>$ eller $|111>$. Lad os nu antage at der sker en X fejl på den første af vore qubits, dvs. vi har enten tilstanden $|100>$ eller $|011>$. Vi ønsker nu at foretage en måling, som fortæller os at der er sket en X-fejl på qubit 1, men som ikke giver os nogen information om hvorvidt tilstanden er $|100>$ eller $|011>$. Dette kan vi gøre ved at måle operatoren $Z_1Z_2 \equiv Z \otimes Z \otimes I$ og operatoren $Z_2Z_3 \equiv I \otimes Z \otimes Z$. Vi ved jo at $|0>$ har egenværdien 1 for operatoren Z og $|1>$ har egenværdien -1, og dermed vil Z_1Z_2 have egenværdi $1 * 1 = -1 * -1 = 1$ hvis qubit et og to er ens, og egenværdi $-1 * 1 = 1 * -1 = -1$ hvis qubit et og to er forskellige - tilsvarende er Z_2Z_3's egenværdi 1 hvis qubit to og tre er ens, og -1 hvis de er forskellige. Det, som de to operatorer har gjort, er med andre ord at måle pariteten af to par af de tre qubits. Vi kan nu opstille de fire mulige egenværdi-par for de to operatorer, og derudfra bestemme hvor en X-fejl er forekommet:

(Z_1Z_2, Z_2Z_3)	X-fejl på qubit
$(1,1)$	ingen
$(-1,1)$	1
$(-1,-1)$	2
$(1,-1)$	3

Hvordan kan man implementere et netværk, som udfra denne teknik kan rette X fejl? "Tricket" er at anvende en såkaldt ancilla, dvs. nogle ekstra qubits, som kun bliver anvendt ved fejlretningen. Således ser vi at netværket i figur 4 udfører det ønskede.



Figur 4: netværk til fejlretning af X fejl. Husk, at operatoren M er en måling af ancilla qubittens værdi, og de NOT gates, som implementeres med M som kontrolator, tilvælges dermed på en klassisk evaluering af måleresultaterne.

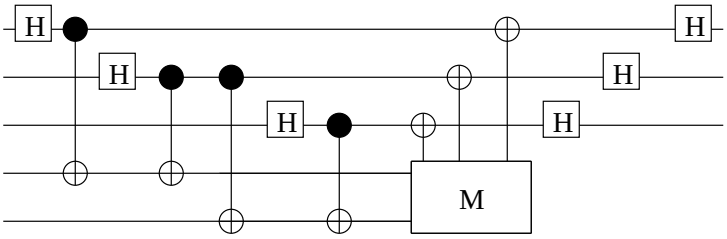
Korrektion af Z-fejl

Rettelse af Z fejl foregår i princippet på samme måde som rettelse af X fejl: Lad os igen for nemheds skyld antage, at den qubit vi ønsker at beskytte er i en egentilstand af X operatoren, dvs. enten $\frac{1}{\sqrt{2}}(|0> + |1>)$ eller $\frac{1}{\sqrt{2}}(|0> - |1>)$. Vi kopierer nu først denne qubit til to andre qubits og anvender, at Z skifter imellem de to egenvektorer til X, dvs. $Z\frac{1}{\sqrt{2}}(|0> + |1>) = \frac{1}{\sqrt{2}}(|0> - |1>)$ og

$Z\frac{1}{\sqrt{2}}(|0>-|1>) = \frac{1}{\sqrt{2}}(|0>+|1>)$. Således kan vi nu - nøjagtigt som ovenfor - måle pariteten af X på tilstanden, dvs. $X_1X_2 \equiv X \otimes X \otimes I$ og $X_2X_3 \equiv I \otimes X \otimes X$, og dermed nå frem til at måle hvor der evt. er indtruffet en Z fejl:

(X_1X_2, X_2X_3)	Z-fejl på qubit
$(1,1)$	ingen
$(-1,1)$	1
$(-1,-1)$	2
$(1,-1)$	3

I implementationen anvender vi teknikken beskrevet nederst i afsnit 2.1 (at skifte beregnings basis fra $\{|0>,|1>\}$ til $\{\frac{1}{\sqrt{2}}(|0>+|1>), \frac{1}{\sqrt{2}}(|0>-|1>)\}$), idet vi så stort set kan anvende ovenstående netværk. Skiftet imellem de to baser sker ved hjælp af Hadamard operatoren (heraf det alternative navn Hadamard rotationen), idet $H|0>= \frac{1}{\sqrt{2}}(|0>+|1>)$ og $H|1>= \frac{1}{\sqrt{2}}(|0>-|1>)$. Bemærk at $H^2 = I$, hvorfor Hadamard rotationen også skifter tilbage til den oprindelige basis igen. Hermed bliver netværket som det, der er angivet i figur 5.



Figur 5: netværk tilsvarende figur 4, blot sørger hadamard transformationerne her for, at vi fejlretter Z fejl istedet for X fejl.

Korrektion af generelle fejl - Shor koden

Ovenfor har vi set, hvorledes man kan rette henholdsvis X eller Z fejl, men som vi ved, skal vi kunne rette *begge* typer af fejl, for at vi kan rette generelle fejl. Dette kan opnås ved en sammenkobling af de ovenstående koder, som historisk er den første kvante fejlkorrektionskode, nemlig Shor-koden (først publiceret i [13]): Vi anvender først X-korrektions koden til at sikre tre blokke, hver med tre fysiske qubits, imod X fejl, dvs. vi anvender 9 fysiske qubits. Disse tre logiske qubits anvender vi så Z fejlkorrektionskoden på, således at tilstanden også sikres imod Z fejl, og dermed er beskyttet imod generel støj, sålænge denne kun rammer en enkelt qubit. Ønsker vi at indkode den generelle tilstand $|\Psi>= \alpha|0>+\beta|1>$ får vi $\alpha(|000>+|111>)^{\otimes 3} + \beta(|000>-|111>)^{\otimes 3}$ - X fejl kan nu korrigeres ved at anvende teknikken ovenfor på hver af de tre tre-qubits blokke, og dermed måle Z_1Z_2, Z_2Z_3, Z_4Z_5, Z_5Z_6, Z_7Z_8 og Z_8Z_9. Hvad angår Z fejl skal vi i princippet måle $X_{\text{blok 1}}X_{\text{blok 2}}$ og $X_{\text{blok 2}}X_{\text{blok 3}}$, men her skal vi huske på at Z fejlen kan opstå på hver fysisk qubit, hvorfor vi er nødt til at måle på alle tre qubits for at måle på en blok - mao. vi skal måle $X_1X_2X_3X_4X_5X_6$ og $X_4X_5X_6X_7X_8X_9$. En fejl i en blok kan korrigeres ved at anvende Z på en tilfældig qubit i blokken, idet den Z fejlbefængte tilstand af blok fra en logisk $|0>$ qubit vil være $Z_x(|000>+|111>) = (|000>-|111>)$,

uanset om x er 1, 2 eller 3, og $Z_1(|000> - |111>) = |000> + |111>$, og en tilsvarende sammenhæng gælder for den logiske $|1>$ tilstand.
Det er her naturligt at indføre en generel notation for kvante fejlkorrektionskoder: Shor koden er en $[[9, 1, 3]]$-kode, hvor tallenes betydning er følgende:
Vi anvender 9 fysiske qubits til at kode 1 logisk qubit, og der skal 3 fejl til at ændre en lovlig tilstand til en anden. Dette udtrykkes ofte på den måde at koden har "hamming afstanden" 3, og det er det, som gør at vi kan rette 1 fejl med koden, idet det jo dermed gælder, at en tilstand som har været udsat for støj på 1 qubit vil være (mindst) 2 fejl fra alle lovlige tilstande, undtagen den tilstand som vi ønsker at komme tilbage til - dette afsnit kan faktisk forstås som en beskrivelse af, hvordan vi finder denne oprindelige tilstand og kommer tilbage til den.

4.2 Stabilisator koder. [[5,1,3]] koden

Med Shor koden kunne vi egentligt stoppe vores analyse af koder som stabiliserer hukommelsen: Den opfylder det krævede, nemlig at beskytte hukommelsen imod støj af enhver art, sålænge denne støj kun rammer en enkelt qubit ad gangen. Vi vil dog forsøge at udvikle en bedre kode, forstået på den måde, at vi ønsker at reducere antallet af fysiske qubits som skal bruges til at indkode en logisk qubit. Grunden til dette ønske er dels, at en begrænsende faktor i fysisk implementerede kvantecomputere netop er antallet af tilrådighedværende qubits, dels at risikoen for støj jo vokser med antallet af fysiske qubits (simpelthen fordi der er flere steder, hvor støj kan give andledning til fejl), og endeligt er andre koder bedre beskrevne i litteraturen, specielt med henblik på fejltollerante operationer, som diskuteres i afsnit 4.3. Den bagvedliggende teori for begrebet "stabilisator koder" er gennemgået i appendiks C - slutresultatet bliver, at en række operatorer siges at være *generatorer* for en ikke-udartet stabilisatorkode hvis de opfylder en bestemt række af krav. I appendiks C argumenteres der ligeledes for, at der er en en-til-en korrespondance mellem disse generatorers egenværdi og hvilken fejl der er indtruffet i hukommelsens tilstand - essentielt på samme måde som operatorerne i Shor-koden ovenfor.

4.2.1 [[5,1,3]] stabilisator koden

De krav, som appendiks C specificerer for at et sæt af operatorer er generatorer for en [[n,k,d]] ikke-udartet stabilisator kode kan kort formuleres således:

1. Der skal være $n - k$ operatorer.
2. Operatorerne skal kommutere med hindanden.
3. Enhver generel fejloperator skal med en vægt mindre end d skal antikommutere med mindst en af operatorerne.
4. Idet koden er ikke-udartet skal hver af disse fejloperatorer antikomutere med et unikt sæt af operatorerne.

Er disse krav opfyldt, så er operatorerne virkeligt, som postuleret, generatorer for [[n,k,d]] koden. Vi vil nu postulere at følgende sæt af operatorer genererer [[5,1,3]] koden:

$$\begin{aligned} M_1 &\equiv XZZXI \\ M_2 &\equiv IXZZX \\ M_3 &\equiv XIXZZ \\ M_4 &\equiv ZXIXZ \end{aligned} \tag{16}$$

Jeg har her (og i det følgende) valgt at udelade tensorprodukterne, da notationen ellers bliver meget hæmmende.
Vi observerer at de tre sidste generatorer er cykliske permutationer af den første, og desuden at den "manglende" $M_5 \equiv ZZXIX$ kan skrives som $M_5 = M_1M_2M_3M_4$, og den er således med i stabilisatoren (hvis generatorerne alle har egenværdien $+1$ for en given tilstand, har M_5 det også).
Det første krav, at der er $n-k = 5-1 = 4$ generatorer ses umiddelbart at være opfyldt.
Det andet krav, at operatorerne kommuterer med hindanden, kan enten ses ved direkte inspektion, eller også som konsekvens af, at der for hvert par af operatorer er der netop to kollisioner af en X og en Z operator (dvs. der er netop to qubits hvor den ene operator har en X operator og den anden en Z operator, eller vice versa). Da disse operatorer jo antikomutere, og da alle andre kombinationer af enkeltqubitoperatorer kommuterer, er det klart at operatorerne kommuterer.
Det tredje krav vil vi argumentere for opfyldelsen af, ved at vise at enhver 1- eller 2-vægts fejloperator altid antikomutere med 1 af operatorene fra sættet $\{M_1, M_2, M_3, M_4, M_5\}$. Vi kan tillade os at medtage M_5, idet hvis en fejloperator antikomutere med M_5 må den også antikomutere med en af operatorerne som M_5 kan skrives som et produkt af. Grunden til at vi medtager M_5 er, at sættet vi så skal checke en given fejloperator imod så er M_1 samt dennes cykliske permutationer, hvilket forsimpler følgende analyse.
Enkeltqubit fejloperatorerne ses nemt at antikomutere med i hvertfald en af operatorerne, idet vi ser at der, for alle qubits eksisterer både en operator, som har X, og en anden som har Z, på den pågældende qubits plads. Da I er den eneste operator af I, X, Y, Z som kommutere med begge disse, kan en fejloperator ikke have en fejl på en enkelt qubit, og samtidigt kommutere med alle operatorerne.
For fejloperatorer med vægten 2 vil vi først observere at der til hver qubitplads tilhører netop én operator, som har I på denne qubits plads.

Dernæst observerer vi, at for et givet par af qubitpladser, f.eks. qubit 2 og qubit 5, har hver operatorer fra sættet $\{M_1, M_2, M_3, M_4, M_5\}$ et unikt par af enkeltqubitoperatorer - eksempelvis har M_2 - og *kun* M_2 - X på både qubitplads 2 og 5. Hvis vi antager at vi har en fejloperator med vægt 2, som har sine to fejl på to givne qubits, kan vi dermed udvælge de to operatorer som har I på en af de to pladser. Disse to operatorer har en enkeltqubitoperator på den anden qubits plads, som hver kun kan være X eller Z. Dette specificerer fejloperatoren til netop den operator, som har disse to enkeltqubitoperatorer

på de to pladser, hvor der optræder fejl.
Eksempel: Lad os antage at vi har en operator som introducerer fejl på qubit to og tre. Her vil vi vælge M_3 og M_4: I M_3 ses det, at den eneste relevante enkeltqubitoperator, som fejloperatoren skal kommutere med, er X på qubit tre's plads - på alle andre qubits er enten M_3 eller fejloperatoren I, der jo kommuterer med alting. Den eneste mulighed er derfor at fejloperatoren på qubit tre har operatoren X. Tilsvarende giver sammenligning med M_4 at fejloperatoren på qubit to har operatoren X, hvorfor fejloperatoren må være $IXXII$.
Når fejloperatoren således er fast bestemt, er der kun to muligheder for at den kommuterer med de resterende 3 operatorer: enten kommuterer fejloperatoren og en given M_i på både qubit to og tre, eller også antikomutere de på både qubit to og tre. Men da M_i'erne kun består af I,X og Z enkeltqubitoperatorer (og ikke Y) giver det kun to muligheder: XX eller ZZ. Men da vi jo har observeret at to M_i operatorer aldrig har de samme enkeltqubitoperatorer på et givent par af qubits, kan fejloperatoren kun komme til at kommutere med to ekstra M_i, dvs. fire ialt. Dermed vil der altid være mindst en operator M_i som den antikomuterer med.
Det sidste krav, at hver enkeltqubit fejloperatorer (de fejloperatorer vi ønsker at kunne rette) svarer til et unikt sæt af operatorer, som denne fejloperator antikomuterer med, kontrollerer vi eksplicit. Det har ydermere den fordel, at vi dermed får et skema, som kan oversætte fra måleresultaterne af operatorerne, til hvilke fejl der er opstået, hvilket umiddelbart giver os en klassisk proces, som udfra målingerne af egenværdierne kan bestemme hvilken operator der vil korrigere effekten fra støjen.

Først X fejlene:

X fejl på qubit nr.	Antikomutationsvektor med $M_1M_2M_3M_4$
1	0001
2	1000
3	1100
4	0110
5	0011

Dernæst Z fejlene:

Z fejl på qubit nr.	Antikomutationsvektor med $M_1M_2M_3M_4$
1	1010
2	0101
3	0010
4	1001
5	0100

Og tilsidst Y fejlene:

Y fejl på qubit nr.	Antikomutationsvektor med $M_1M_2M_3M_4$
1	1011
2	1101
3	1110
4	1111
5	0111

Herudfra kan vi også se, at [[5,1,3]] koden er "perfekt": Enhver givent mønster af antikommutation med operatorerne svarer netop til en fejl, hvorfor der ikke er plads til flere mulige fejl, end de femten vi netop beskytter os imod.
Vi ser nu, at operatorene $\{M_1, M_2, M_3, M_4\}$ faktisk genererer [[5,1,3]] stabilisa-

torkoden, og har dermed vist det postulerede.
Vi kan altså nu korrigere enkeltqubitfejl med [[5,1,3]]-koden ved at måle egenværdien af de fire generatorer som specificeret i (16), og evt. rette de fejl, som måtte være opstået, efter ovenstående skemaer.

4.2.2 tilstande i [[5,1,3]] koden

At have specificeret, hvordan vi kan rette opståede fejl i tilstande som er korrekte kodeord, dvs. tilstande som svarer til logiske tilstande, i [[5,1,3]] koden er langt fra nok: I resten af dette afsnit vil vi udvikle en komplet "værktøjskasse" til at kunne implementere Shors algoritme på en fejlresistent måde. Der er først et par grundlæggende ting, som det vil være formålstjenligt at overveje følgende basale operationer mht. [[5,1,3]]-koden:

- Definition af de logiske tilstande $|\bar{0}>$ og $|\bar{1}>$, samt en operation der kan skelne imellem dem, og en operation der kan skifte imellem dem (løst sagt logiske versioner af σ_z og σ_x).
- Initialisering til et kodeord i [[5,1,3]] koden, eller: hvordan man laver en tilstand af fem qubits som faktisk er et kodeord i [[5,1,3]] koden.

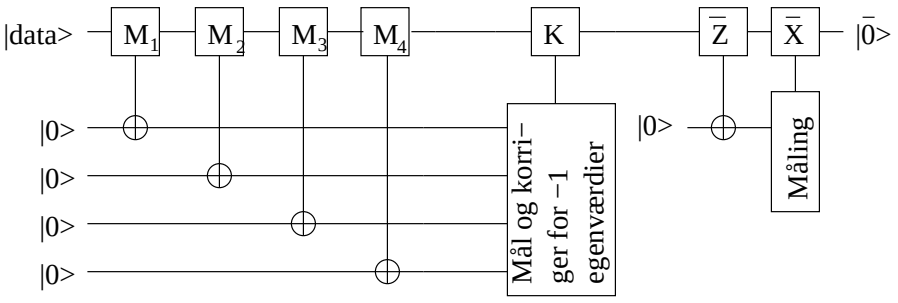
logiske tilstande

Situationen er indtil videre at vi kan holde en given tilstand i rummet $H_{\mathcal{S}}$ (se appendiks C) af tilladte tilstande i [[5,1,3]] koden stabil, dvs. vi kan reparere enkeltqubit fejl på denne tilstand. Indtil videre har vi dog ikke nogen definition af, hvilken del af dette rum vi skal forstå som den logiske $|\bar{0}>$ og den logiske $|\bar{1}>$. Nøglen til at lave en sådan definition er, at finde to operatorer, kaldet $\bar{Z}$ og $\bar{X}$, som begge kommuterer med stabilisatorerne, og samtidigt antikomuterer med hindanden. De vil have de egenskaber som umiddelbart kan forventes af "logisk σ_z" og "logisk σ_x", nemlig at de bevarer tilstanden i det tilladte koderum (da de kommuterer med stabilisatorerne), og hvis den ene, f.eks. $\bar{X}$, bliver anvendt på et kodeord, flippes egenværdien af $\bar{Z}$ imellem $+1$ og -1, idet hvis f.eks $\bar{Z}|\Psi>=|\Psi>$ så er $\bar{Z}\bar{X}|\Psi>=-\bar{X}\bar{Z}|\Psi>=-\bar{X}|\Psi>$, og omvendt. Det kan teoretisk bevises at sådanne operatorer eksisterer, men vi vil her blot præsentere dem: $\bar{Z}=ZZZZZ$ og $\bar{X}=XXXXX$. Det kan let checkes at disse opfylder kravene, altså komuterer med alle stabilisatorerne og antikomuterer med hindanden. Idet vi vedtager at $\bar{Z}$ skal være den operator, som definerer vores logiske basis, siger vi at en tilstand i $|\Psi>\in H_{\mathcal{S}}$ som opfylder at $\bar{Z}|\Psi>=|\Psi>$ er i den logiske tilstand $|\bar{0}>$, mens en tilstand $|\Psi>\in H_{\mathcal{S}}$ som opfylder at $\bar{Z}|\Psi>=-|\Psi>$ er i den logiske tilstand $|\bar{1}>$.

initialisering af tilstande til [[5,1,3]] koden

En af styrkerne ved stabilisatorformalismen er, at der er en simpel måde at initialisere kvantecomputerens hukommelse. Startpositionen er et antal, i vores tilfælde fem, qubits i en ukendt tilstand, som vi ønsker at transformere ind i

rummet $H_{\mathcal{S}}$, således at den har f.eks. egenværdien $+1$ for operatoren $\bar{Z}$. At tilstanden ligger i $H_{\mathcal{S}}$, husker vi fra appendiks C, er ækvivalent med er en egenvektor med egenværdien $+1$ for samtlige generatorer i $\mathcal{S}$. Dette illustrerer at begge krav kan opfyldes ved den samme fremgangsmåde, nemlig ved at kollapse hukommelsens tilstand til et givet egenrum af en række operatorer, nemlig generatorerne og $\bar{Z}$. Dette kan opnås ved forskellige metoder, og jeg vælger her en metode, som inkluderer målinger af operatorerne, idet det på en enkel måde får hukommelsen til at kollapse til egenrum af operatorerne (se afsnit 2). Eneste problem er, at vi kan risikere at få -1 som egenværdier for generatorerne eller $\bar{Z}$, men hvis det sker, skal vi blot anvende en operator, som antikomuterer med den operator hvis egenværdi blev målt til -1, og samtidigt komuterer de operatorer, hvis egenrum hukommelsen allerede er restringeret til, således at hukommelsen kommer i det korrekte del af egenunderrummet for samtlige operatorer, dvs. den del af underrummet, hvor alle operatorerne har egenværdi $+1$.
I vort tilfælde kan vi faktisk let optimere denne proces lidt, idet vi uden videre har operatorer som antikomuterer med alle delmængder af stabilisatorerne: Det er jo netop de enkeltqubit transformationer, som [[5,1,3]] koden har til formål at beskytte imod, jf. afsnit 4.2.1, og da koden er perfekt, har vi en "fejloperator" til enhver kombination af stabilisatorer. Målingerne af generatorerne giver en information, som vi kan anvende til en baglæns opslag i tabellerne nederst i afsnit 4.2.1, og vi udfører så den tilhørende operator. Herefter er hukommelsen i $H_{\mathcal{S}}$, hvorefter vi måler $\bar{Z}$, og alt efter hvilken egenværdi tilstanden kollapser til, er vi enten færdige, eller også anvender vi $\bar{X}$ (der jo antikomuterer med $\bar{Z}$ og komuterer med alle generatorerne), og vi herefter er vi i den ønskede tilstand. Netværket er illustreret på figur 6.



Figur 6: Kvantenetværk, som bringer fem qubits fra en tilfældig startilstand til $|\bar{0}>$. Bemærk dels, at $|data>$ faktisk er en femqubit tilstand, samt at operatoren K bestemmes udfra et omvendt opslag i tabellerne nederst i 4.2.1

4.3 Operationer på den fejltollerante tilstand

I afsnit 4.2 opnåede vi at stabilisere i tilstand i hukommelsen - i den uformelle definition vi indtil videre har anvendt, vil det sige, at så længe der kun opstår støj på en enkelt qubit, så kan vi implementere et kvantenetværk, som kan rette denne fejl. Vi har desuden set hvordan vi kan forberede tilstande, som giver me-

ning i [[5,1,3]] koden, nemlig $|\bar{0}>$ og $|\bar{1}>$, samt hvordan vi kan skifte imellem dem vha. operatoren $\bar{X}$, og måle hvilken tilstand en logisk qubit befinder sig i med $\bar{Z}$.
Dette er dog ikke nok til at gennemføre Shor's algoritme. Hertil skal vi naturligvis også kunne lave operationer med disse logiske qubits, f.eks. skal vi kunne lave en logisk $CNOT$ gate, altså en $CNOT$ gate imellem to *logiske* qubits. Desuden skal vi sørge for, at alle de metoder, som blev specificeret i afsnit 4.2, ikke i sig selv indføjer uoprettelige fejl i tilstanden - f.eks. vil en fejl i en måling i netværket som måler en given generators egenværdi kunne føre til introduktion af nye fejl i den logiske tilstand, udover de fejl, som uønsket interaktion med naturen evt. har introduceret..
Før vi kan gå ind i detaljerne om hvordan vi opbygger netværk, som sørger for at fejltollerante beregninger på tilstandende fra [[5,1,3]] koden, vil det være nødvendigt at gennemgå en række basale observationer og definitioner ifm. fejltollerante kvantenetværk:

- Vi har brug for en klar model for, hvordan vi simulere støj i kvantecomputeren.
- En klar definition af, hvad vi mener med at et kvantenetværk er fejltollerant.
- En diskussion af nogle generelle krav til fejltollerante kvantenetværk, baseret på definitionen af fejltollerant kvantenetværk.

Desuden må det siges, at dette emne i sin fulde udstrækning er for stort til at det kan gennemgåes i nærværende rapport. Jeg vil derfor kun gennemgå de logiske operationer i en hvis detalje, mens jeg vil henvise til litteraturen (specielt [5] afsnit 10.6) for de mere "basale" operationer, dvs. fejltollerant forberedelse af hukommelsen, fejltollerant måling af hukommelsens tilstand og fejltollerant fejlretning af hukommelsen. Dog vil jeg eksplicit nævne de detaljer i forbindelse hermed, som jeg har brug for i resten af rapporten, samt præsentere en kvalitativ diskussion af dem.

Støjmodel

Støjmodellen afhænger naturligvis af den fysiske implementation af kvantecomputeren. Vi har dog i denne rapport valgt at begrænse os fra at diskutere fysiske implementationer af kvantecomputeren, og vi vælger derfor en relativt arbitrær, simpel model: For et givent tidsrum er der en sandsynlighed p for, at der vil optræde en form for fejl, som kan ekspanderes i sættet $\{I, X, Y, Z\}$, på en given fysisk qubit. Bemærk at vi her ikke har sagt noget om, hvor mange operationer kvantecomputeren kan nå at udføre på dette tidsrum - det afhænger af hastigheden på kvantecomputeren.

En definition af et fejltollerant netværk

Som indgang til denne definition, må vi konstatere, at det ikke er muligt at garderer sig *fuldstændigt* imod fejl: Hvis vi er tilstrækkeligt uheldige, kan der

opstå arbitrære fejl på samtlige fysisk qubits samtidigt, og vi vil dermed have en totalt randomiseret tilstand i hukommelsen - hvilket vi naturligvis ikke kan rette op på. Vi vil istedet forlange, at der er en sammenhæng imellem risikoen for at der opstår støj på en given *fysisk* qubit, og risikoen for, at kvantenetværket fejler, dvs. giver et forkert resultat som følge af støj i netværket: Jf. ovenfor definerer vi risikoen for at der opstår en fejl på en fysisk qubit indenfor et givet tidsrum som p, og vi forlanger så at et fejltollerant netværk, at det giver et korrekt resultat med sandsynligheden $1 - O(p^2)$, dvs. fejlrisikoen kvadreres (bortset fra en evt konstant fra O funktionen). Vi ser, at [[5,1,3]] koden opnår netop denne effekt umiddelbart hvis vi blot ønsker at opbevare en logisk qubits tilstand fejltollerant: Den kan vha. et netværk af fast størrelse, og dermed med fast tidsforbrug, bevirke at en fejl i den logiske qubit kræver *to* fejl i de fysiske qubits. Da netværket har et konstant tidsforbrug, er risikoen for fejl på en enkelt qubit $O(p)$. Hermed er risikoen for fejl på to qubits jo $O(p^2)$, og vi har opnået det ønskede. Dette eksempel illustrerer et vigtigt krav: Vi skal med et *konstant* antal fysiske operationer (og dermed konstant tidsforbrug) kunne reducere netværkets fejlrisiko fra $O(p)$ til $O(p^2)$: Hvis tidsforbruget ikke er konstant kan vi ikke være sikre på risikoen for at der opstår fejl på fysiske qubits er konstant (længere tidsforbrug = større fejlrisiko), dvs. p ændres, og der er ikke nogen sikkerhed for at $O(p_{ny}^2) < O(p_{gammel})$.
Det er relativt nemt at indse, at hvis vi kan gøre de nedenfor specificerede blokke i kvantenetværket for Shor's algoritme fejltollerante, så er hele kvantenetværket fejltollerant. Beviset for dette er dog temmeligt trættende og ikke specielt interessant, og jeg henviser derfor til [5] s. 478-480, hvor analysen er gennemført for et "netværk", der består af en enkelt CNOT gate. Det generaliserer til de andre basale operationer, og kan rekursivt anvendes på et arbitrært stort netværk, f.eks. Shor's algoritme. Vi vil dog observere at måden at implementere et større netværk på en fejltollerante måde er, at lave hver fejltollerante operation, og derefter lave fejlretning for *alle* logiske qubits, og så fortsætte med den næste logiske operation - bare det at opbevare qubits udsætter dem for støj, hvorfor vi ikke kan nøjes med de qubits som vi har opereret på.

Generelle krav til fejltollerant netværk

Udfra ovenstående definition, kan vi give visse retningslinjer for, hvordan et givent fejltollerant netværk må se ud. For det første må der ikke være nogen operationer som involverer flere fysiske qubits, som ligger i den samme logiske qubits blok. For at illustrere grunden til dette krav, vil vi her vise, hvilke problemer der kan opstå, hvis vi udfører en CNOT operation mellem to qubits i den samme "blok": Fra starten af netværket, til vi begynder at udføre CNOT gaten, går der et konstant stykke tid, dvs. der er en risiko på $O(p)$ for at der optræder en X fejl på kontrol qubitten. Når vi så udfører CNOT gaten, ser vi,

at følgende transformation effektivt finder sted:

$$CNOT \circ (X \otimes I)|\Psi >= CNOT \circ (X \otimes I) \circ CNOT^{\dagger} \circ CNOT|\Psi >= \quad (17)$$
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \circ CNOT|\Psi >=$$
$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \circ CNOT|\Psi >= (X \otimes X) \circ CNOT|\Psi >$$

Men idet det "forventede" resultat (dvs. resultatet uden støj) er $CNOT|\Psi >$, ødelægger dette fejltollerancen: Resultat i (17) vil opstå med sandsynligheden $O(p)$, og vi kan ikke reparere fejlen, idet den er blevet "spredt" til to qubits, hvorfor der med samme sandsynlighed vil være opstået fejl i den logiske qubit. En interessant effekt i kvantecomputeren er, at $CNOT \circ I \otimes Z \circ CNOT^{\dagger} = Z \otimes Z$, dvs. fejl kan sprede sig *fra* målqubitten *til* kontrolqubitten, ulig det klassiske tilfælde. Dette giver blot nok en grund til ikke at tillade operationer mellem fysiske qubits som indgår i samme logiske qubits blok, hvis vi ønsker at netværket skal være fejltollerant.
Et andet krav, som vi må stille til et fejltollerant netværk må være, at en enkelt fejl på en qubit hvor som helst i netværket ikke forårsager, at *nogen* logisk qubit noget andet sted i netværket bliver så fejlbefængt, at den ikke kan repareres, idet risikoen jo så er $O(p)$ for logiske fejl i netværket. Et simpelt eksempel på en netværk, som opfylder alle andre krav end dette, og som klart ikke er fejltollerante, er et netværk, som bruger en fysisk qubit fra en logisk qubit som kontrol qubit i to CNOT gates, der har målqubit på to fysiske qubits i den samme logiske qubit blok. Hvis der her sker en X fejl på kontrol qubitten, inden CNOT operationerne, så kommer der to fejl i mål qubit blokkene, hvilket vi ikke kan reparere.
Udfordringen for dette afsnit kan præcist formuleres således: vi ønsker at designe netværk som kan udføre alle operationer vi indtil videre har benyttet i denne rapport, og som samtidigt opfylder ovenstående krav.

4.3.1 Basale operationer

Som sagt i indledningen af dette afsnit, vil jeg ikke gå i detaljer med disse operationer, men istedet referere til [5] afsnit 10.6.3 for en fejltolerant metode til at måle værdier af forskellige operatorer, f.eks. generatorerne af stabilisatorkoden og $\bar{Z}$, samt fejlretning og forberedelse af de logiske egentilstande, $|\bar{0}>$ og $|\bar{1}>$. Det bør her nævnes, at mht. målinger er der et yderligere krav for at operationen er fejltollerant: Resultatet skal være korrekt (dvs. det samme som i det teoretiske netværk uden støj) med en sandsynlighed på mindst $1-O(p^2)$. Grunden til dette er dels, at resultatet af Shors algoritme vil blive målt med denne operation (hvorfor netværkets fejltollerance aldrig kan blive bedre en sandsynligheden for en korrekt måling), og dels at vi anvender målinger i mange af de

følgende fejltollerante netværk, hvorfor *disses* fejltollerance beror på at resultatet af en måling er korrekt.
Dette giver os en umiddelbar måde at lave fejlretning fejltollerant: Eftersom vi nu kan måle en generator på en fejltollerant måde, vil vi nu kunne måle alle fire stabilisatorer fejltollerant (da $4 * O(p^2) = O(p^2)$). Udfra disse målinger vil vi nu, på den måde som er beskrevet i afsnit 4.2.1, anvende den korrekte gate (hvis der er opstået en fejl) for at rette hukommelsestilstanden. Grunden til at denne operation er fejltollerant er dels, at vi ved, at målingerne er korrekte med sandsynligheden $1 - O(p^2)$, dels at vi i afsnit 1.3 har forudsat at gaten, som evt. retter fejlen, er perfekt.
Ligeledes er det nu nemt at se, hvorledes vi kan forberede de logiske tilstande $|\bar{0}>$ og $|\bar{1}>$: Vi anvender blot den metode, som blev beskrevet i afsnit 4.2.2, men sørger for at anvende den fejltollerant version af målingerne. Idet vi igen antager at de fysiske gates er perfekte, så er det klart at den resterende del af initialiseringen er fejltollerant.

4.3.2 Logiske operationer på [[5,1,3]] koden

En logisk, fejltollerant gate ændrer på den logiske hukommelse (dvs. tilstanden af de logiske qubits) af kvantecomputeren. Dette betyder, at vi udover de allerede stillede krav, er nødt til at forlange at hukommelsen efter den logiske gate ligger i $H_{\mathcal{S}}$ hvis hukommelsen før den logiske gate gjorde det. I appendiks D er den grundlæggende teori (taget fra [6]), som vi vil anvende til at opbygge fejltollerante versioner af hadamard transformationen, CNOT gaten og faseskiftsoperatorne ($P(d)$ fra 2.2.1).
Appendiks D godtgør, at vi for at implementere en operation fejltollerant, skal konstruere et netværk, som dels opfylder de allerede specificerede krav til fejltollerante netværk, og desuden opfylder at den netværket transformerer $\bar{X}$'erne og $\bar{Z}$'erne på samme måde som den operation vi forsøger at gøre fejltollerant. Mere specifikt kan dette krav formuleres således: hvis $\bar{L}$ betegner en arbitrær $\bar{X}$ eller $\bar{Z}$ operator, og netværket svarer til operationen N, så implementerer N den logiske operator U hvis og kun hvis det for alle $\bar{L}$ gælder at

$$N\bar{L}N^{\dagger} = U\bar{L}U^{\dagger} \tag{18}$$

Desuden skal det vises, at for alle generatorerne, M_i, i [[5,1,3]] gælder, at

$$N\bar{M}_iN^{\dagger} \tag{19}$$

kan skrives som et produkt af stabilisatorer.
Inden vi konstruerer netværk for de ønskede operationer, vil vi gennemgå to grundlæggende fejltollerante operationer, som vi vil bruge til at konstruere de andre - specielt vil vi vise, at (19) er opfyldt for dem, hvorfor det også er opfyldt for alle operationer som er bygget op af dem.

Grundlæggende fejltollerante operationer

Den grundlæggende transformation, som vi vil bygge de andre transformationer op af, er (jf. [6]) transformationen $T : \bar{X} \rightarrow i\bar{Y} \rightarrow \bar{Z} \rightarrow \bar{X}$. Her, og i resten af

afsnittet, definerer vi $\bar{Y} = Y^{\otimes n}$. Lad os vise, at denne transformation opfylder kravet om, at et element i stabilisatoren bliver transformeret til et andet element i stabilisatoren. Først indser vi, at $T^3 = I$ da transformationen er cyklisk. Dernæst, idet vores definition af generatorerne for [[5,1,3]]-koden er cyklisk, så er det nok at vise at TM_1 og T^2M_1 ligger i stabilisatoren: for $2 \leq i \leq 4$ er TM_i og T^2M_i blot cykliske rotationer af TM_1 og T^2M_1, og dermed i stabilisatoren hvis TM_1 og T^2M_1 er det. Vi ser nu at

$$TM_1 = T(X \otimes Z \otimes Z \otimes X \otimes I) = -Y \otimes X \otimes X \otimes Y \otimes I = M_3M_4 \tag{20}$$

og

$$T^2M_1 = T^2(X \otimes Z \otimes Z \otimes X \otimes I) = T(-Y \otimes X \otimes X \otimes Y \otimes I) = (-Z \otimes Y \otimes Y \otimes Z \otimes I) = M_1M_3M_4 \tag{21}$$

Hermed ses det, at T er en fejltollerant operation.
Vi får brug for en slags triple version af T, dvs. en transformation, som virker på 3 logiske qubits istedet for blot 1 som T. Denne transformation kalder vi i lighed med [6] for T_3, og den er defineret ved

$$\begin{aligned}
\bar{X} \otimes I \otimes I &\rightarrow \bar{X} \otimes T\bar{X} \otimes T^2\bar{X} = i\bar{X} \otimes \bar{Y} \otimes \bar{Z} \\
\bar{Z} \otimes I \otimes I &\rightarrow \bar{Z} \otimes T\bar{Z} \otimes T^2\bar{Z} = i\bar{Z} \otimes \bar{X} \otimes \bar{Y} \\
I \otimes \bar{X} \otimes I &\rightarrow T\bar{X} \otimes \bar{X} \otimes T^2\bar{X} = i\bar{Y} \otimes \bar{X} \otimes \bar{Z} \\
I \otimes \bar{Z} \otimes I &\rightarrow T\bar{Z} \otimes \bar{Z} \otimes T^2\bar{Z} = i\bar{X} \otimes \bar{Z} \otimes \bar{Y} \\
I \otimes I \otimes \bar{X} &\rightarrow \bar{X} \otimes \bar{X} \otimes \bar{X} \\
I \otimes I \otimes \bar{Z} &\rightarrow \bar{Z} \otimes \bar{Z} \otimes \bar{Z}
\end{aligned} \tag{22}$$

Vi ser først, at T_3 kan implementeres ved at implementere den tilsvarende transformation bitvist - dvs. imellem de tre første qubits i de tre blokke, dernæst mellem de tre næste qubits i de tre blokke, osv. Idet vi har set, at T er fejltollerant, er de fire første linjer i definitionen af T_3 det også, idet de to logiske qubits, som før transformationen blot har operationen I, efter operationen vil have en cyklisk transformation af operationen på den qubit, som startede med en operation, som vi antager ligger i stabilisatoren, f.eks. M_1 - men dermed har vi, jf. ovenstående bevis for T's fejltollerance, at hver af de logiske qubits bliver udsat for en operation som er i stabilisatoren. Slutteligt ser vi, at de sidste to sidste linjer blot kopierer den operation som virker på den tredje qubit til de to første. Under antagelse af at operationen på den tredje qubit er i stabilisatoren, er operationerne på hver af de logiske qubits efter transformationen det derfor også. Dermed er T_3 bevist at være fejltollerant. Vi vil i det følgende bruge disse to transformationer til at opbygge de transformationer, som svarer til de ønskede operationer, f.eks. $CNOT$ gaten.
I de følgende konstruktioner vil de eneste operatorer, som vil blive anvendt til *målinger* (se appendiks D), være logiske enkeltqubitoperatorer, dvs. operatorer der, opererer på de logiske qubits med enten I eller en af $\bar{X}$ eller $\bar{Z}$. Disse operatorer vil pr. definition altid komuterer med generatorerne af [[5,1,3]]-koden på de logiske qubits, og der er således ikke nogen grund til at arbejde videre med disse generatorer: for konstruktion af fejltollerante logiske operationer er

generatorer kun interessante, for så vidt at de antikomuterer med en operator, som bliver målt. Fra nu af vil vi derfor kun overveje "specielle" generatorer, som eksisterer pga. vort forkendskab til en given tilstand: f.eks. er $\bar{Z}$ generator for $|\bar{0}>$, og $I \otimes \bar{X}$ er generator for $|\Psi> \otimes \frac{1}{\sqrt{2}}(|0> + |1>)$, hvor $|\Psi>$ står for en ukendt tilstand.

Hadamard

Det første skridt for at opbygge en fejltollerant version af hadamard gaten er at finde $H\bar{X}H^\dagger$ og $H\bar{Z}H^\dagger$: Målet vil derefter være at lave en transformation, som gør dette, og samtidigt opfylder kravene til en fejltollerant operation.

$$\begin{aligned} H\bar{X}H^\dagger &= \bar{Z} \\ H\bar{Z}H^\dagger &= \bar{X} \end{aligned} \tag{23}$$

I [6] redegøres der for, at $H = P(1)Q^\dagger P(1)$, hvor $Q = T^\dagger P(1)$ - for at være konsistent med [6] vil jeg herefter definere $P \equiv P(1)$, og samtidigt påpege at Gottesman anvender betegnelsen R for hadamard transformationen. Vi har nu at $H = P(T^\dagger P)^\dagger P = PP^\dagger TP = TP$. Idet T allerede er fejltollerant, er det eneste vi nu skal gøre at vise at P kan implementeres fejltollerant - metoden til at gøre dette står i [6], men vi præsenterer den også her, dels for kompletheden, dels fordi det er en simpel anvendelse af principperne for fejltollerant beregning. Først må vi gøre os klart hvilken transformation vi præcist ønsker:

$$\begin{aligned} P\bar{X}P^\dagger &= i\bar{Y} \\ P\bar{Z}P^\dagger &= \bar{Z} \end{aligned} \tag{24}$$

Denne transformation opnås ved at forberede to (logiske) ancilla qubits i tilstanden $|\bar{0}>$, og betragte indgangs tilstanden til T_3 transformationen som $|\bar{0}\bar{0}\Psi>$, hvor $|\Psi>$ er den tilstand vi ønsker at anvende P på. Vi ser her, at $\bar{X} = I \otimes I \otimes \bar{X}$, $\bar{Z} = I \otimes I \otimes \bar{Z}$, idet disse to hhv. flipper og måler den logiske dataqubit $|\Psi>$, og de interessante generatorer er $\bar{Z} \otimes I \otimes I$ og $I \otimes \bar{Z} \otimes I$. Jf. (22) transformerer T_3 disse til $\bar{X} = \bar{X} \otimes \bar{X} \otimes \bar{X}$, $\bar{Z} = \bar{Z} \otimes \bar{Z} \otimes \bar{Z}$, og stabilisatorerne bliver $i\bar{Z} \otimes \bar{X} \otimes \bar{Y}$ og $i\bar{X} \otimes \bar{Z} \otimes \bar{Y}$. Vi måler nu to operatorer, nemlig $I \otimes \bar{Z} \otimes I$ og $I \otimes I \otimes \bar{Z}$ - først $I \otimes \bar{Z} \otimes I$: Den antikomuterer med den første generator og $\bar{X}$, og $\bar{X}$ erstattes derfor af $(i\bar{Z} \otimes \bar{X} \otimes \bar{Y})(\bar{X} \otimes \bar{X} \otimes \bar{X}) = -i\bar{Y} \otimes I \otimes -Z = i\bar{Y} \otimes I \otimes Z$. Derefter måles $I \otimes I \otimes \bar{Z}$, som antikomuterer med den tilbageværende generator, men ikke andet. Idet vi nu vælger at ignorere de sidste to bits har vi tilbage at $\bar{X} \to i\bar{Y}$ og $\bar{Z} \to \bar{Z}$, hvilket jf. (24) ses præcist at være P.
Vi har altså at hadamard transformationen gennemføres ved følgende skridt:

1. Forbered tilstanden $|\bar{0}\bar{0}\Psi>$ hvor $|\Psi>$ er inddata tilstanden
2. Anvend T_3 på denne tilstand
3. Mål anden og tredje qubits tilstand i beregnings basen (og korriger for måleresultatet -1), og ignorer disse to qubits.
4. Andvend transformationen T på den tilbageværende qubit

CNOT

Ligesåvel som for hadamard transformationen vil vi starte med at opskrive den transformation af $\bar{X}_1 \equiv \bar{X} \otimes I$, $\bar{X}_2 \equiv I \otimes \bar{X}$, $\bar{Z}_1 \equiv \bar{Z} \otimes I$ og $\bar{Z}_2 \equiv I \otimes \bar{Z}$ som vi ønsker at opnå:

$$\begin{array}{c} CNOT(\bar{X} \otimes I)CNOT^\dagger = \bar{X} \otimes \bar{X} \\ CNOT(I \otimes \bar{X})CNOT^\dagger = I \otimes \bar{X} \\ CNOT(\bar{Z} \otimes I)CNOT^\dagger = \bar{Z} \otimes I \\ CNOT(I \otimes \bar{Z})CNOT^\dagger = \bar{Z} \otimes \bar{Z} \end{array} \tag{25}$$

Beskrivelsen af hvorledes man opnår denne transformation udfra T_3 og T er ligesom for hadamard transformationen taget fra [6] afsnit V.
Inden vi laver selve CNOT transformationen, er det smart at lave følgende hjælpetransformation, A:

$$\begin{array}{c} \bar{X}_1 \rightarrow i\bar{Y} \otimes I \\ \bar{X}_2 \rightarrow i\bar{Y} \otimes \bar{Z} \\ \bar{Z}_1 \rightarrow \bar{Z} \otimes i\bar{Y} \\ \bar{Z}_2 \rightarrow i\bar{Y} \otimes \bar{X} \end{array} \tag{26}$$

Det første vi gør er at forberede tilstanden $|\Psi> \otimes |\Phi> \otimes |\bar{0}>$, hvor $|\Psi>$ og $|\Phi>$ er de to inddata-qubits. Derefter anvender vi T_3, og idet vi inden gaten har at $\bar{X}_1 = \bar{X} \otimes I \otimes I$, $\bar{X}_2 = I \otimes \bar{X} \otimes I$,$\bar{Z}_1 = \bar{Z} \otimes I \otimes I$ og $\bar{Z}_2 = I \otimes \bar{Z} \otimes I$ og generatoren $I \otimes I \otimes \bar{Z}$, får vi efter T_3 gaten at: $\bar{X}_1 = i\bar{X} \otimes \bar{Y} \otimes \bar{Z}$, $\bar{X}_2 = i\bar{Y} \otimes \bar{X} \otimes \bar{Z}$, $\bar{Z}_1 = i\bar{Z} \otimes \bar{X} \otimes \bar{Y}$ og $\bar{Z}_2 = i\bar{X} \otimes \bar{Z} \otimes \bar{Y}$ og generatoren $\bar{Z} \otimes \bar{Z} \otimes \bar{Z}$. Derefter måles operatoren $I \otimes \bar{X} \otimes I$, som ses at antikomutere med generatoren, $\bar{X}_1$ og $\bar{Z}_2$, hvorfor vi får

$$\begin{array}{c} \bar{X}_1 = -i\bar{Y} \otimes -\bar{X} \otimes I = i\bar{Y} \otimes \bar{X} \otimes I \\ \bar{X}_2 = i\bar{Y} \otimes \bar{X} \otimes \bar{Z} \\ \bar{Z}_1 = i\bar{Z} \otimes \bar{X} \otimes \bar{Y} \\ \bar{Z}_2 = -i\bar{Y} \otimes I \otimes -\bar{X} = i\bar{Y} \otimes I \otimes \bar{X} \end{array} \tag{27}$$

samt generatoren $I \otimes \bar{X} \otimes I$. Vi smider derefter den anden qubit ud, og har dermed til slut

$$\begin{array}{c} \bar{X}_1 = i\bar{Y} \otimes I \\ \bar{X}_2 = i\bar{Y} \otimes \bar{Z} \\ \bar{Z}_1 = \bar{Z} \otimes i\bar{Y} \\ \bar{Z}_2 = i\bar{Y} \otimes \bar{X} \end{array} \tag{28}$$

hvilket vi genkender som den ønskede transformation A. Dernæst indser hvis vi har implementeret en fejltollerant version af P, så har vi implicit også implementeret $P^\dagger = P^{-1}$ fejltollerant: De netværk som indgår, kan udenvidere anvendes omvendt, og målinger og indsættelser af ancilla qubits i den originale transformation "bytter plads" i den omvendte transformation. For at illustrere dette vil vi oversætte en "opskrift" fra konstruktionen af en given original transformation, til den tilsvarende opskrift for den omvendte transformation: Den originale transformation: "Indføj ancilla qubitten $|\Xi>$ som stabiliseres af operatoren M_1 på plads y, og mål operatoren M_2 for qubitten på plads x og smid den væk", hvilket i den omvendte transformation bliver til: "indføj på plads x en qubit i tilstanden $|\Psi>$, som stabiliseres af operatoren M_2, og mål operatoren M_1 for

qubitten på plads y, og smid den væk". Argumentet for at disse transformationer netop er de omvendte af hindanden er, at de transformerer alle operatorer (generatorer og $\bar{X}$ og $\bar{Z}$ operatorer) på netop den omvendte måde af hindanden. Observer desuden at $T^{-1} = T^2$, hvorfor denne uden videre kan implementeres. Nu er CNOT transformationen nem at udføre, idet vi anvender hjælpetransformationen A ovenfor, samt P og T, og indser, jf. [6], at A ovenfor også kan opnås vha. CNOT transformationen og allerede implementerede, fejltollerante operationer:
Vi starter igen med standart $\bar{X}$'erne og $\bar{Z}$'erne,$\bar{X} \otimes I$, $I \otimes \bar{X}$, $\bar{Z} \otimes I$ og $I \otimes \bar{Z}$, og observerer at:

$$\begin{array}{cccccccc} \bar{X}\otimes I & & \bar{X}\otimes I & & \bar{X}\otimes I & & i\bar{Y}\otimes I \\ I\otimes\bar{X} & I\otimes T^\dagger P & I\otimes\bar{X} & CNOT & \bar{X}\otimes\bar{X} & P\otimes T^2 & i\bar{Y}\otimes\bar{Z} \\ \bar{Z}\otimes I & \rightarrow & \bar{Z}\otimes I & \rightarrow & \bar{Z}\otimes\bar{Z} & \rightarrow & \bar{Z}\otimes i\bar{Y} \\ I\otimes\bar{Z} & & I\otimes i\bar{Y} & & \bar{X}\otimes i\bar{Y} & & i\bar{Y}I\otimes\bar{X} \end{array}$$

, hvor det skal bemærkes at CNOT operationen har første bit som *mål*qubit og anden bit som *kontrol*qubit. Udfra ovenstående kan det således ses, at hjælpetransformationen kan skrives som $A = (I \otimes T^\dagger P)CNOT$(fra qubit 2 til 1)$(P \otimes T^2)$, hvorfor vi kan skrive $CNOT$(fra qubit 2 til 1) $= (I \otimes P^\dagger T)A(P^\dagger \otimes T^{\dagger 2})$, dvs. en sammenkobling af fejltollerante operationer - og en sådan operation er selv fejltollerant, jf. ovenfor.

Fase forskydning

Ovenstående operationener, H, P og CNOT, er nok til at generere alle operatorer U som opfylder at $UGU^\dagger \in G_n, G \in G_n$ - sættet af alle mulige U der opfylder dette kriterium kaldes i litteraturen normalisatoren af G_n. Desværre har Gottesman og Knill bevist, at dette sæt ikke er komplet, og faktisk kan simuleres med en klassisk computer ([6]). Shor og DiVincenzo har bevist (i [8]) at man ved at tilføje toffoligaten (en dobbeltkontroleret NOT gate) kan få et komplet sæt af fejltollerante operatorer, og Gottesman viser i [6] hvordan man implementerer denne gate vha. teknikken i dette afsnit. Den operation, vi ønsker at kunne implementere, for at kunne konstruere Shor's algoritmes netværk, er $P(d)$, og hvis vi ønskede det, kunne anvende toffoligaten til sammen med ovenstående gates til at aproksimere $P(d)$ til en præcision ϵ med $Poly(Log(\epsilon^{-1}))$ gates, dvs. de kan implementeres effektivt - dette er bevist af bl.a. Kitaev i [9]. Den faktiske konstruktion af $P(d)$ udfra toffoli gaten er dog ikke nødvendigvis en simpel opgave, og vi vil derfor ikke bruge toffoligaten til at sættet komplet. Istedet følger vi anvisningerne i [7], hvor det dels bevises at sættet $\{H, CNOT, P(2)\}$, er komplet, dels præsenteres en konstruktionsanvisning, som vi vil lade os inspirere af til at skabe $P(d+1)$ induktivt udfra $P(d)$. Præsentationen her ligner den i afsnit 4 i [7], blot er målet $P(d+1)$ istedet for $P(2)$, og udgangspunktet $P(d)$ istedet for P - vi bemærker her, at [7] anvender betegnelsen $\sigma_z^{\frac{1}{2^{d+1}}}$ istedet for $P(d)$.
For at kunne implementere $P(d+1)$ udfører vi to skridt:

- først konstruerer vi tilstanden $|\phi_0> \equiv P(d+1)H|\bar{0}> = \frac{|\bar{0}> + e^{\frac{i\pi}{2^{d+1}}}|\bar{1}>}{\sqrt{2}}$

- dernæst konstruerer vi $P(d+1)$ baseret på denne tilstand.

Konstruktion af $|\phi_0>$

Nøglen til at lave denne tilstand er at vi kan skabe tilstanden $|cat>=\frac{|00000>+|11111>}{\sqrt{2}}$ (navnet kommer fra, at hvis der er mange qubits, så svarer det til kattens tilstand inden boksen åbnes i tankeeksperimentet "Schrödingers kat") fejltollerant, og vi kan måle den i basen $\{\frac{|00000>+|11111>}{\sqrt{2}}, \frac{|00000>-|11111>}{\sqrt{2}}\}$ fejltollerant - dette blev faktisk også brugt i de afsnit i [5] som vi refererede til i afsnit 4.3.1, og vi vil ligesom i afsnit 4.3.1 ikke bevise det her. Vi observerer at den tilstand, vi ønsker at skabe, er en egentilstand af operatoren $U \equiv P(d+1)\bar{X}P(d+1)^{-1}$, med egenværdien 1:

$$P(d+1)\bar{X}P(d+1)^{-1}P(d+1)H|\bar{0}>=P(d+1)\bar{X}H|\bar{0}>=P(d+1)H\bar{Z}|\bar{0}>=P(d+1)H|\bar{0}> \tag{29}$$

Samtidigt ses, at tilstanden $|\phi_1>\equiv P(d+1)H|\bar{1}>=\frac{|\bar{0}>-e^{\frac{i\pi}{2^{d+1}}}|\bar{1}>}{\sqrt{2}}$ også er en egentilstand til U, men med egenværdien -1, idet

$$P(d+1)\bar{X}P(d+1)^{-1}P(d+1)H|\bar{1}>=P(d+1)H\bar{Z}|\bar{1}>=-P(d+1)H|\bar{1}> \tag{30}$$

At vi kan implementere U fejltollerant ses let, idet $U = P(d+1)\bar{X}P(d+1)^{-1} = e^{\frac{i\pi}{2^{d+1}}}P(d+1)^2\bar{X} = e^{\frac{i\pi}{2^{d+1}}}P(d)\bar{X}$, og induktionsantagelsen var netop at $P(d)$ kunne implementeres fejltollerant (og ovenfor har vi allerede vist at $\bar{X}$ kan implementeres fejltollerant). Vi kan nu skabe $|\phi_0>$ fejltollerant, dvs. vi kan skabe tilstanden under de sædvanlige betingelser, og vi kan vide, med sandsynligheden $1-O(p^2)$, at den tilstand vi ender med, faktisk er $|\phi_0>$: Vi ved fra [1] at hvis vi kan implementere U og $CNOT$ fejltollerant, så kan vi også implementere en fejltollerant, kontrolleret U gate, som i litteraturen ofte benævnes $\vee_1(U)$. Desuden ser vi, at idet $|\phi_0>$ og $|\phi_1>$ er ortogonale kan enhver tilstand i koderummet udspændt af $|\bar{0}>$ og $|\bar{1}>$ skrives som $|\Phi>=\alpha|\phi_0>+\beta|\phi_1>$. Lad os nu anvende $\vee_1(U)$ på tilstanden $|cat>\otimes|\Phi>$, hvor $|\Phi>$ er en tilfældig tilstand i koderummet:

$$\begin{aligned}\vee_1(U)\tfrac{1}{\sqrt{(2)}}(|00000>+|11111>)\otimes\alpha|\phi_0>+\beta|\phi_1>= \\ (\tfrac{1}{\sqrt{(2)}}|00000>\otimes\alpha|\phi_0>+\beta|\phi_1>)+ \\ (\tfrac{1}{\sqrt{2}}|11111>\otimes\alpha|\phi_0>-\beta|\phi_1>)= \\ (\tfrac{1}{\sqrt{2}}(|00000>+|11111>)\otimes\alpha|\phi_0>)+ \\ (\tfrac{1}{\sqrt{2}}(|00000>-|11111>)\otimes\beta|\phi_1>)\end{aligned} \tag{31}$$

Idet vi kan lave en fejltollerant måling af, hvorvidt kontrolqubitten er i tilstanden $\frac{1}{\sqrt{2}}(|00000>+|11111>)$ eller $\frac{1}{\sqrt{2}}(|00000>-|11111>)$, kan vi (fejltollerant) bestemme, hvorvidt vi har skabt tilstanden $|\phi_0>$ eller $|\phi_1>$. Hvis vi har skabt $|\phi_0>$ er vi færdige, hvis vi har skabt $|\phi_1>$ kan vi blot anvende $\bar{Z}$ på tilstanden, idet $\bar{Z}|\phi_1>=|\phi_0>$ - $\bar{Z}$ er allerede bevist fejltollerant.

Konstruktion af $P(d+1)$

Vi har nu skabt $|\phi_0>$, men formålet var jo at implementere $P(d+1)$ udfra $P(d)$. Det viser sig dog at være ret let: Lad os kalde inddata tilstanden $|\Phi>$, og opskrive den vha $|\bar{0}>$ og $|\bar{1}>$. Denne anvender vi som kontrolqubit i en fejltollerant CNOT, hvor $|\phi_0>$ er målqubitten:

$$\begin{aligned} &CNOT(|\Phi> \otimes |\phi_0>) = \\ &(\alpha|\bar{0}> \otimes \frac{|\bar{0}>}{\sqrt{2}}) + (\alpha|\bar{0}> \otimes \frac{e^{\frac{i\pi}{2^{d+1}}}|\bar{1}>}{\sqrt{2}}) + \\ &(e^{\frac{-i\pi}{2^{d+1}}}\beta|\bar{1}> \otimes \frac{e^{\frac{i\pi}{2^{d+1}}}|\bar{1}>}{\sqrt{2}}) + (e^{\frac{i\pi}{2^{d+1}}}\beta|\bar{1}> \otimes \frac{|\bar{0}>}{\sqrt{2}}) = \\ &(P(d+1)|\Phi> \otimes \frac{|\bar{0}>}{\sqrt{2}}) + (P(d+1)^{-1}|\Phi> \otimes e^{\frac{i\pi}{2^{d+1}}}\frac{|\bar{1}>}{\sqrt{2}}) \end{aligned} \tag{32}$$

På samme måde som det blev gjort ovenfor, kan vi nu bestemme hvorvidt kontrolqubitten nu er i tilstanden $P(d+1)|\Phi>$ eller i tilstanden $P(d+1)^{-1}|\Phi>$, idet vi jo, jf. afsnit 4.3.1, fejltollerant kan måle hvorvidt målqubitten er i tilstanden $|\bar{0}>$ eller $|\bar{1}>$. Hvis den er i tilstanden $|\bar{0}>$ er dataqubitten i tilstanden $P(d+1)|\Phi>$ - den ønskede tilstand. Hvis målqubitten derimod er i tilstanden $|\bar{1}>$ er dataqubitten i tilstanden $P(d+1)^{-1}|\Phi>$ - denne tilstand "reparerer" vi så, fejltollerant, ved at anvende $P(d)$ på dataqubitten:

$$P(d)P(d+1)^{-1}|\Phi> = P(d+1)^2P(d+1)^{-1}|\Phi> = P(d+1)|\Phi> \tag{33}$$

Dette ses at være den ønskede tilstand, og vi har dermed konstrueret en fejltollerant version af $P(d+1)$ udfra $P(d)$. Idet vi ser, at $P(1) = P$ som blev implementeret fejltollerant ovenfor, kan vi nu induktivt konstruere $P(d)$ for ethvert $d \in \mathbb{N}$, og den sidste gate som skal anvendes for at konstruere netværket for Shor's algoritme er dermed implementeret fejltollerant.

Tidsforbruget af det fejltollerante faseskift

Ovenstående konstruktion har desværre en alvorlig fejl: I Shor algoritme er $d \in \Theta(n)$, og lad os for nemheds skyld blot sætte $d = n$, dvs. vi skal lave en QFT (kvante Fourier transformation) af en størrelse med d qubits. Hvor mange basale operationer kræves der for at udføre denne operation fejltollerant? Vi viste i afsnit 3 at der er brug for et polynominelt antal $P(x)$ gates, hvor $x \in [1, d]$, men disse skal nu opbygges af et antal gates for at blive fejltollerant. Idet vi udelukkende ser på anvendelsen af $P(x-1)$ gates (og ikke $\bar{X}$ gates) ses det, at vi for at konstruere $P(x)$ i værste tilfælde skal bruge *to* $P(x-1)$ gates: En til at skabe $|\phi_0>$, og en til at rette dataqubitten, hvis denne skulle vise sig at være i tilstanden $P(x)^{-1}|\Phi>$ (se foregående afsnit). Men idet vi kun kan lave $P(1) = P$ uden anvendelse af andre $P(x)$ operationer, får vi da, i værste tilfælde, at $P(x)$ kræver 2^x P operationer. Hvis vi iflg. QFT konstruktionen skal skabe $P(d)$ vil denne altså kræven $2^d = 2^n$ operationer, og dermed have et tidsforbrug på $\Theta(2^n)$ idet P kan implementeres i konstant tid. Idet hele pointen med Shor's algoritme netop var, at den kunne implementeres i $poly(n)$ tid, er

dette umiddelbart fatalt for hele projektet.
Heldigvis kan situationen reddes: Vi kan lave en aproksimeret QFT ved at ignorere $P(x)$ gates for tilstrækkeligt store x, idet faseforskydningen, som $P(x)$ forårsager, bliver eksponentielt mindre i x. Vi kan f.eks. nøjes med at udføre $P(x)$ for $x < m$, og under forudsætning af, at n er tilstrækkeligt stor (hvilket i vort tilfælde jo er den interessante situation) har Barenco et al. vist, i [10], at sandsynligheden for at få det ønskede resultat af QFT (som i afsnit 3 blev vist at være $\frac{4}{\pi^2}$) bliver større end

$$\frac{8}{\pi^2} sin^2(\frac{\pi}{4}\frac{m}{n}) \tag{34}$$

Hvis vi ser på den "ekstreme" approksimation, dvs. $\frac{m}{n} \approx 0$, ser vi at ovenstående grænse kan rækkeudvikles til

$$\frac{1}{2}(\frac{m}{n})^2 + O((\frac{m}{n})^4) \tag{35}$$

Dermed ses det, at sandsynligheden for *ikke* at få det forventede resultat i en enkelt gennemkørsel, idet vi negligerer højereordensled af $\frac{m}{n}$ er

$$1 - \frac{1}{2}(\frac{m}{n})^2 \tag{36}$$

Hvis vi blot i ét gennemløb får det korrekte resultat vil algoritmen, med konstant sandsynlighed, tillade os at faktorere tallet N. Sandsynligheden for, at dette *ikke* sker på x gennemløb er

$$(1 - \frac{1}{2}(\frac{m}{n})^2)^x \tag{37}$$

Vort ønske er nu at kunne evaluere x ift. n, under forudsætning af, at sandsynligheden i (37) er konstant som funktion af n. Idet vi nu udnytter, at $(\frac{m}{n})^2 \approx 0$, og vi således kan negligere alle led, som er højere orden end 2 i $\frac{m}{n}$, ser vi, at (37) kan aproksimeres til

$$1 - x\frac{1}{2}(\frac{m}{n})^2 \tag{38}$$

Således ser vi, at hvis vi kører algoritmen ca. $(\frac{n}{m})^2 \in poly(n)$ gange, så bliver sandsynligheden for, at algoritmen mislykkes *alle* gangene konstant i n, hvorfor sandsynligheden for succes i blot én gennemkørsel også er konstant i n, og vi er tilbage ved den oprindelige situation.
Det er klart, at i denne sammenhæng er sandsynligheden for et korrekt resultat i ét gennemløb en monotom voksende funktion i m, hvorfor situationen $\frac{m}{n} \approx 0$ er den værst tænkelige situation for os. Dermed har vi vist at vi ved at køre den aproksimerede algoritme $\Theta(n^2)$ gange, kan opnå samme resultat som for det ideelle netværk i afsnit 3, med en tidskompleksitet på ialt $\Theta(n^3)\Theta(n^2) = \Theta(n^5)$, dvs. stadigvæk en polynomiel tidskompleksitet i n. Effekten på $P(x)$'s tidsforbrug er tydeligvis, at det bliver begrænset af $\Theta(2^m)$, og idet vi kan *vælge* m, og kun tage en polynomiel straf (af $(\frac{n}{m})$) på antallet af gange, algoritmen skal gennemløbes for at kompensere for dette valg, kan vi nu betragte tidsforbruget af faseforskydningerne som konstante.

5 Minimal hastighed for kvantecomputeren

I afsnittet 4 analyserede vi os frem til en måde at implementere Shor's algoritme på, således at hvis risikoen for at få en fejl på en enkelt fysisk qubit i løbet af et givet tidsrum er p, så er sandsynligheden for, at hele netværket eksekverer korrekt $1-O(p^2)$. Det er i sig selv et ganske godt resultat: Vi har nu et værktøj til at reducerer effekten af den støj, som, inden afsnit 4, truede med at ødelægge hele ideen i at bygge en kvantecomputer.
Vi kan faktisk, under visse omstændigheder, gøre det endnu bedre: Det er i visse tilfælde muligt, at implementere et netværk, som eksekverer korrekt med sandsynligheden $1-O(p^{(2^d)})$ ved anvendelse af $O(5^d)$ ekstra qubits, og $O(c^d)$ ekstra basale instruktioner, hvor c er et konstant tal. Det ses, at vi kan få en forbedring af sandsynligheden for korrekt eksekvering stiger eksponentielt som funktion det ekstra forbrug af qubits og basale instruktioner. Analysen af, hvornår det er muligt at lave sådanne netværk, kaldes i litteraturen for grænseanalyse (eng: "Threshold analysis"), og det er en sådan analyse, som ville lede os frem til den ønskede funktion for den krævede hastigheden af kvantecomputeren - det viser sig, at afhængigt af problemets størrelse, skal kvantecomputeren kunne eksekvere et vist antal operationer pr. tidsenhed, for at ovenstående "trick" kan implementeres.
Desværre må det konstateres, at omfanget af opgaven for denne rapports emne har vist sig noget større end forventet, både mht. tidsforbrug og plads. Vi vil derfor begrænse os til at diskutere funktionen kvalitativt, specielt dens afhængighed af n, og desuden diskutere den fejltollerante faseforskydning fra sidste afsnit, idet der viser sig at være en vis sammenhæng mellem denne og grænseanalyse.
Tabet ved ikke faktisk at lave grænseanalysen er begrænset: For at en sådan skal give mening ville det være nødvendigt at optimere metoderne i afsnit 4, i en forstand som vil blive forklaret nedenunder - som det er, har vi kun udviklet én relativt tilfældig metode til at gøre kvantenetværket fejltollerant.

5.1 Grænseanalyse

Vi vil her kort gennemgå teorien for grænseanalyse. Jeg har hentet inspiration i [5] afsnit 10.6, men gør opmærksom på, at Nielsen anvender en anden, grundlæggende model for støj end os, hvorfor en separat analyse er nødvendig.
Den grundlæggende idé er relativt simpel: Ved at sprede informationen i de logiske qubits til 5 fysiske qubits, opnåede vi at gå fra en fejlrisiko på $O(p)$ til $O(p^2)$ ved anvendelse af 5 gange flere qubits, og et lineær forøgelse af antallet af nødvendige basale operationer (til fejlretning, fejltollerante logiske gates osv. som beskrevet i afsnit 4). Men vi kan jo gentage denne operation: Vi har nu et antal fysiske qubits som beskriver et antal logiske qubits, men disse fysiske qubits kunne vi igen fejlsikre, ved proceduren beskrevet i afsnit 4. Dermed ville vi opnå $O(p^4)$ sandsynlighed for fejl, med $5^2 = 25$ gange flere qubits, og en kvadratisk forøgelse af antallet basale operationer. Vi kan nu fortsætte med at fejlsikre de fysiske qubits i netværket, principielt uendeligt dybt: Dette er et ganske godt resultat, idet vi ser, at vi ved en eksponentiel forøgelse af

resourceforbruget (tid og qubits), opnå en "dobbelt-eksponentiel" forbedring af fejltollerancen - dvs. hvis vi allokerer $\Theta(Exp(d))$ flere qubits og basale operationer, bliver risikoen for, at netværket fejler pga. støj $\Theta(Exp(Exp(d)))$, som vi postulerede i starten af dette afsnit.
Der er en forudsætning, som vi indtil videre ikke har formuleret eksplicit: For hvert lag vi tilføjer, vokser risikoen for at få fejl på fysiske qubits: Dels kommer der 5 gange flere fysiske qubits, hvilket betyder at der også kommer flere qubits som støj kan påvirke. Dels skal vi efter hver logisk operation på det "nederste" lag af logiske qubits (dvs. de logiske qubits, som umiddelbart består af fysiske qubits) udføre fejlkorrektion på samtlige af disse logiske qubits. For hvert lag efter det første stiger antallet af logiske qubits på det nederste lag med en faktor 5, hvorfor der også skal udføres 5 gange flere fejlkorrektionsoperationer for hvert lag, inden vi kommer til en situation, hvor vi med sikkerhed kan udtrykke risikoen for fejl i netværket som $1 - O(p^2)$. Desuden skal hver operation i det nederste lag af logiske qubits oversættes fra en simpel fysisk operation, til en logisk operation, som er en faktor c' længere end den fysiske operation den erstatter (hvilket vi forlangte i afsnit 4.3). Effekten af, at vi får flere fysiske qubits for hvert lag, og dermed større fejlrisiko, kan til første orden i p evalueres som

$$1 - (1-p)^5 \approx 1 - 1 + 5p = 5p \tag{39}$$

dvs. også herfra forøges fejlrisikoen for en enkelt qubit i systemet lineært.
Alle disse "straffe" for at tilføje et enkelt lag af fejlretningskode ser vi dermed, at vi kan udtrykke som en lineært udtryk af den fejlrisiko, vi opererede med, før vi tilføjede laget. Dvs. reelt er risikoen for fejl på netværket gået fra p til $(cp)^2$, hvor c er en konstant. Det er nu klart, at hvis $c < \frac{1}{p}$, så kan det betale sig at tilføje et lag - ellers kan det ikke, og det er præcis denne grænse, som grænseanalyse sigter mod at bestemme c, og dermed opstille et krav til, hvor lille p skal være, før vi kan implementere algoritmer fejltollerant på en given kvantecomputerimplementation.
Idet vi vælger ikke at tillade parallelitet i kvantecomputeren, dvs. hver basal operation skal udføres enkeltvist, forværres situationen faktisk en smule: c bliver nu en lineær funktion af n, idet der på et givent lag vil være $n*5^d$ logiske qubits, som det er nødvendigt at fejlkorrigere. Dette er umiddelbart et uoverkommeligt problem: Hvis vi skal kunne implementere Shor's algoritme for vilkårligt store værdier af n, skal vi så tilsyneladende implementere en kvantecomputer med en fejlrisiko uendeligt tæt på 0, hvilket klart ikke er gennemførligt. Hvis vi har en mulighed for at gøre kvantecomputeren hurtigere, kan vi dog løse problemet: Hvis vi fordobler antallet af operationer pr. tidsenhed, kan vi også klare dobbelt så mange operationer, inden fejlrisikoen stiger til p - det var jo netop definitionen af p at den var risikoen for enkeltqubit fejl over en given tidsperiode. Idet c jo tildels kom fra et antal operationer som skulle udføres for at fejlkorrigere hukommelsen, kan vi udtrykke en hastighedsfordobling ved at erstatte c med $\frac{c}{2}$. Dvs. hvis vi kan forøge hastigheden som en lineær funktion af n, så kan vi opretholde grænsebetingelsen, og dermed opnå en arbitrært lav risiko for fejl pga. støj i netværket, da vi så har at $(cp)^d \to 0$ for $d \to \infty$.

5.2 Grænseanalyse og den fejltollerante faseforskydning

Som lovet i starten af afsnittet, vil vi nu se, at grænseanalysen og valget af m, dvs. approximeringsgraden af den fejltollerante faseforskydning (sidst i afsnit 4.3.2) hænger sammen. Grunden til det er, at for tilstrækkeligt store værdier af m vil blive den længste, og dermed langsomste, operation, idet vi husker at dens længde afhænger eksponentielt af m. Idet den eneste logiske operation som får indflydelse i grænseanalysen er den, som tager længst tid at gennemføre (idet denne operation bestemmer c' ovenfor), vil det være smart at vælge m *lige netop* så høj, at faseforskydningen *ikke* er den længste operation, idet præcisionen af faseforskydning forstørres når m vokser.

6 Konklusion

Målene, som blev opstillet i formålet, er efter min mening stort set blevet nået: afsnit 3 opstiller og forklarer Shor's algoritme i fuld detalje, dvs. jeg har givet et overbevisende argument for, at det på en kvantecomputer er muligt at faktorere en n-bit tal i $\Theta(poly(n))$. Udfra den generelle teori i afsnit 4 har jeg forklaret hvorledes en funktion for hastigheden af kvantecomputeren hænger intimt sammen med problemerne omkring støj i kvantecomputeren i afsnit 5 - dog må det indrømmes at jeg (grundet mangel på tid og plads) ikke formåede rent faktisk at lave grænseanalysen, som det ellers blev opstillet ønske om i formålet.
Det er desuden mit håb, at læseren har fundet rapporten læsbar og instruktiv, på trods af at det må indrømmes, at rapporten er temmeligt formel og matematisk.

6.1 Mulige forbedringer

Jeg vil i dette afsnit give nogle forslag til forbedringer af rapporten:

- Man kunne forsøge at fjerne nogle af afgrænsningerne i afsnit 1.3. Specielt vil det næppe volde de store problemer at medtage støj på de basale gates, men også muligheden for korellerede fejl kunne medtages i en senere version. Derimod vil det formentligt være fornuftigt at holde rapportens emne abstrakt (dvs. ikke diskutere den fysiske implementation af kvantecomputeren), da dette dels fokuserer rapporten, og dels gør det klart, at de centrale resultater i rapporten i meget høj grad er uafhængige valget af implementationsmodel.

- Man kunne undersøge muligheden for mere effektive (evt. optimale) implementationer af de fejltollerante kvantegates i afsnit 4.3.2.

- Implementationen af den fejltollerante faseforskydning må blankt erkendes at være temmeligt omstændelig, omend vi nåede frem til en fejltollerant gate. Det vides at der eksisterer en mere effektiv approksimation, jf. [9], hvilket også er nævnt i afsnit 4.3.2 - man kunne sætte sig for rent faktisk at finde denne.

- Man kunne gennemføre grænseanalysen i afsnit 5. Dette vil være af begrænset interesse før de fejltollerante operationer er optimerede, da deres størrelse indvirker på grænseanalysen.

A Deutsch's algoritme

Deutsch's algoritme er en algoritme, som forsøger at bestemme hvorvidt en funktion $f(x) : \{0,1\} \to \{0,1\}$ er balanceret eller ej, dvs. om $f(0) \neq f(1)$ eller ej, ved kun at evaluere $f(x)$ én gang. Kvantenetværket for algoritmen (eller i alt fald et muligt netværk) kan ses på figur 7, hvor H står for Hadamand transformation, og f gaten er en gate, som for en given tilstand $|i,j>$ (hvor i er tilstanden for den øverste qubit, og j tilstanden for den nederste) returnerer $|i, j + f(i) >$, hvor aditionen er $\mod 2$. Værdierne af de to qubit's er fra start $|0>$ for den øverste qubit og $|1>$ for den nederste. Vi vil nu vise at, efter netværket er gennemløbet er den nederste qubit altid i tilstand $|1>$, og den øverste qubit er i tilstanden $|1>$ hvis f er balanceret og i tilstanden $|0>$ hvis f ikke er balanceret. Lad os først indse, hvad de to første Hadamand transformationer gør ved start tilstanden $|01>$: $H|0> \otimes H|1> = \frac{1}{\sqrt{2}}(|0> + |1>) \otimes \frac{1}{\sqrt{2}}(|0> - |1>) = \frac{1}{2}(|00> + |10> - |01> - |11>)$. I næste skridt, hvor vi anvender funktionen f har vi 4 forskellige tilfælde, én for hver mulig funktion f:

1. $f(0) = f(1) = 0$, hvor tilstanden bliver $\frac{1}{2}(|00> + |10> - |01> - |11>$
2. $f(0) = 0, f(1) = 1$, hvor tilstanden bliver $\frac{1}{2}(|00> + |11> - |01> - |10>$
3. $f(0) = 1, f(1) = 0$, hvor tilstanden bliver $\frac{1}{2}(|01> + |10> - |00> - |11>$
4. $f(0) = f(1) = 1$, hvor tilstanden bliver $\frac{1}{2}(|01> + |11> - |00> - |10>$

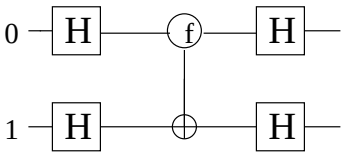
Vi ser at tilfælde 1 og tilfælde 4 (hvor f er konstant) er ens pånær et fortegnsskift, og det samme gælder for tilfælde 2 og 3 (hvor f er konstant). Idet tilstandsfunktioner ikke er bestemt til mere end en total fase (dvs $|x> \equiv e^{i\delta}|x>, \delta \in \mathbb{R}$) har vi nu reelt to tilstande: tilstanden i tilfælde 1 hvis f er konstant og tilstanden i tilfælde 2 hvis f er balanceret. Lad os nu anvende de to sidste Hadamand transformationer på disse tilstande: Først for f konstant:

$$\begin{gathered}\frac{1}{2}((\frac{1}{\sqrt{2}}(|0> + |1>) \otimes \frac{1}{\sqrt{2}}(|0> + |1>)) + \\ (\frac{1}{\sqrt{2}}(|0> - |1>) \otimes \frac{1}{\sqrt{2}}(|0> + |1>)) - \\ (\frac{1}{\sqrt{2}}(|0> + |1>) \otimes \frac{1}{\sqrt{2}}(|0> - |1>)) - \\ (\frac{1}{\sqrt{2}}(|0> - |1>) \otimes \frac{1}{\sqrt{2}}(|0> - |1>))) = \\ \frac{1}{4}(|00> + |01> + |10> + |11> + |00> + |01> - |10> - |11> \\ -|00> + |01> - |10> + |11> - |00> + |01> + |10> - |11> = \\ \frac{1}{4}4|01> = |01>\end{gathered} \tag{40}$$

Derefter for f balanceret:

$$\begin{gathered}\frac{1}{2}((\frac{1}{\sqrt{2}}(|0> + |1>) \otimes \frac{1}{\sqrt{2}}(|0> + |1>)) + \\ (\frac{1}{\sqrt{2}}(|0> - |1>) \otimes \frac{1}{\sqrt{2}}(|0> - |1>)) - \\ (\frac{1}{\sqrt{2}}(|0> + |1>) \otimes \frac{1}{\sqrt{2}}(|0> - |1>)) - \\ (\frac{1}{\sqrt{2}}(|0> - |1>) \otimes \frac{1}{\sqrt{2}}(|0> + |1>))) = \\ \frac{1}{4}(|00> + |01> + |10> + |11> + |00> - |01> - |10> + |11> \\ -|00> + |01> - |10> + |11> - |00> - |01> + |10> + |11> = \\ = \frac{1}{4}4|11> = |11>\end{gathered} \tag{41}$$

Men dette er netop hvad vi ønskede at bevise: Efter netværket er den nederste qubit i tilstand $|1>$ uanset f, mens den øverste netop er afhængig af, hvorvidt f er balanceret - denne kan nu bare måles, og alt efter resultatet har vi udført den ønskede operation.



Figur 7: Kvantenetværk for Deutsch's algoritme

B Fra faktorering til bestemmelse af periode

I afsnit 3 påstod vi, at man ved en klassisk, probabilistisk algoritme, kunne omforme problemet at faktorere et tal N til at finde perioden for funktionen $f(x) = a^x \mod N$, hvor a er et tilfældigt tal, samt at sandsynligheden for at denne omformning faktisk giver en faktorering af N er mindst 0.5. I dette appendix vil vi give et argument for at dette er korrekt:
Problemet for Shor's algoriteme er som skrevet, at faktorere et tal N hurtigt. Det kan betale sig at undersøge dette problem lidt nøjere rent matematisk: Antag, at vi, for et tilfældigt heltal a har fundet det mindste x således, at $a^x \mod N = 1$, eller mao. fundet ordenen af a modolo N (dette kræver at a og N er coprime, men det kan checkes klassisk at $SFD(a, N) = 1$ - er de *ikke* det har vi jo faktoreret N), og antag videre, at x er lige. Vi har da, at $N \mid (a^x - 1) \implies N \mid (a^{\frac{x}{2}} - 1)(a^{\frac{x}{2}} + 1)$. Vi ser, at $N \nmid (a^{\frac{x}{2}} - 1)$, da ordenen af a modolo N i så fald havde været mindre end eller lig med $\frac{x}{2}$, i modstrid med en af vore tidligere antagelser. Antag nu slutteligt, at $a^{\frac{x}{2}} \mod N \neq -1$. Så kan vi sige at $N \nmid a^{\frac{x}{2}} + 1$, og vi kan så sige at der er en ikke-triviel fælles devisor af N og (f.eks.) $a^{\frac{x}{2}} + 1$, idet primtalsfaktoreringen af N er unik (bortset fra ombytning af faktorerne), hvorfor der må eksistere to ikke-tomme delmængder af N's primtalsfaktorer som samtidigt er primtalsfaktorer til hhv. $a^{\frac{x}{2}} - 1$ og $a^{\frac{x}{2}} + 1$, som på en klassisk computer kan findes med $SFD(N, a^{\frac{x}{2}} - 1)$
Vi vil nu bevise følgende sætning:

Sætning 2 *Antag at $N \neq p^\alpha$ for p et primtal og $\alpha \in \mathbb{N}$. Antag desuden, at vi for et naturligt tal a, hvorom det gælder at $SFD(a, N) = 1$, har fundet x så $a^x \mod N = 1$. Da er sandsynligheden for, at x er lige og $a^{\frac{x}{2}} \neq -1$ større end 0.5*

Bevis: Lad $N = p_1^{\alpha_1} p_2^{\alpha_2} \ldots$ repræsentere N's primtalsopløsning, og antag at der er g forskellige indecies så $\alpha_{i,j,\ldots} \neq 0$. Den kinesiske restklassesætning implicerer, at når vi vælger et $a < N$ tilfældigt, så svarer det til at vælge $a_1, a_2, \ldots a_g$ tilfældigt, hvor hvert a_i repræsenterer restklassen af a modulo p_i. Hvis der eksisterer $\alpha_j > 1$ så skal vi vælge α_j tal $(\beta_1, \ldots, \beta_{\alpha_j})$, som opfylder at $\beta_k = (a \mathrm{div} p_j^{k-1}) \mathrm{mod} p_j$. Resten af beviset forløber på samme måde for dette tilfælde, så vi antager herfra for nemhedens skyld at for alle $j \in \mathbb{N}$ gælder at $\alpha_j \in \{0, 1\}$. Idet p_i'erne er primtal, så kan vi til hvert a_i finde ordnen modulo p_i - lad os kalde disse ordner r_i, og pga. den kinesiske restklasse sætning, har vi at $r = MFM(r_1, r_2, \ldots, r_g)$. En følge heraf er, at hvis blot et af r_i'erne er lige, så er r lige, og en af betingelserne er opfyldt, og vi kan begynde at udtale os om, hvorvidt den anden betingelse, $a^{\frac{r}{2}} \neq -1 mod N$ er opfyldt. Idet $N = p_1 p_2 \ldots p_g$ og $a = a_1 a_2 \ldots a_g$ er det klart at

$$\begin{aligned} a^{\frac{r}{2}} \neq -1 \mod N \leftrightarrow \\ a_1^{\frac{r}{2}} \neq -1 \mod p_1 \vee \\ a_2^{\frac{r}{2}} \neq -1 \mod p_2 \vee \\ \vdots \\ a_g^{\frac{r}{2}} \neq -1 \mod p_g \end{aligned} \tag{42}$$

eller mao., hvis blot ét sæt $\{a_i, p_i\}$ opfylder at $a_i^{\frac{r}{2}} = 1 \mod p_1$ så er begge betingelser opfyldt, og det er muligt at bestemme primtalsfaktoreringen af N. For at evaluere sandsynligheden for dette, vil vi observere nogle ting om ordnerne $r_1, \ldots, r_g$: lad c_i være defineret som potensen af 2 i r_i's primtalsopløsning. Antag at der eksisterer to forskellige værdier, c_j og c_k tilhørende to forskellige ordner, r_j og r_k. Eftersom $r = MFM(r_1, \ldots, r_g)$, gælder det da (idet vi lader $c_j > c_k$) at $r = 2 * r_k * l, l \in \mathbb{Z}$, og derfor at $a_k^{\frac{r}{2}} = a_k^{lr_k} = 1 \mod p_k$, idet r_k jo netop er ordnen af a_k modulo p_k, og vi har at $a^{\frac{r}{2}} \neq -1 mod N$. Omvendt, hvis alle værdierne af $c_i, 1 \leq i \leq g$ er ens, medfører det, at der for ethvert $k \in [1, g]$ gælder at $r = r_k * m, m \in [1]_2$ (hvor $[1]_2$ betyder "restklassen 1 modulo 2", dvs. de ulige tal), og idet vi husker at $a_k^{\frac{r_k}{2}} = -1 mod p_k$, giver det os at $a_k^{\frac{r_k}{2} * m} = -1 \mod p_1$ for alle k, dvs. vi kan ikke anvende den fundne værdi af r[4]. Lad os for at evaluere chancen for at dette er opfyldt huske at alle de multiplikative grupper modulo p_1 til p_g er cykliske, og se på tilfældet k, dvs. $a_k^{r_k} \mod p_k$, og her definere o ved $p_k - 1 = 2^o * m, m \in [1]_2$. Da gruppen er cyklisk eksisterer der et element b som har orden $p_k - 1$, og alle elementer i gruppen kan skrives som en potens af dette element. Da gælder der, at hvis et element kan skrives som $b^{2q+r}, r \in [1]_2$ så har det jo ordnen $2^{o-p} * t, t \in [1]_2$, dvs. halvdelen af elementerne har ulige orden ($c_k = 0$), en fjerdedel har orden $2 * u, u \in [1]_2$ ($c_k = 1$), en ottendedel har orden $4 * v, v \in [1]_2$ ($c_k = 2$), osv. (indtil den sidste mulighed, som indeholder ligeså mange elementer som den næstsidste mulighed - og ikke halvt så mange). Således ser vi, at når vi vælger et tilfældigt element, har dette med sandsynligheden 0.5 $c_k = 0$, med sandsynlighed 0.25 $c_k = 1$, osv. Idet vi antager at der eksisterer i alt fald 2 værdier af k så $\alpha_k \neq 0$ i primtalsopløsningen af N så ser vi, at chancen for at der eksisterer forskellige c_k er mindst 0.5: Hvis vi nemlig antager at det første element har $c_k = 0$ er sandsynligheden for, at det andet element *ikke* har $C_k = 0$ $1 - 0.5 = 0.5$. Denne sandsynlighed er større for andre værdier af c_k for det første element. Antager vi at første element har en anden værdi af c_k, så er denne s, idet det er den maximale sandsynlighed for en given værdi af hvert af c_i'erne. Hvis det gælder at $N = p^\alpha$ kan Shor's algoritme ikke anvendes, men dette kan effektivt checkes af klassiske algoritmer, f.eks. kunne man forsøge at beregne alle $\sqrt[\beta_i]{N}$ for alle $\beta_i < n, \beta_i$ et primtal, hvor $2^n > N$, og evaluere om nogen af disse er hele tal - i så fald vil vi have faktoreret N, og vi er færdige. Denne algoritme er klart polynominel i n, idet der er mindre end n forskellige rødder som skal beregnes, og hver af disse kan beregnes i Poly(n) tid.

[4]tilfældet at r er ulige, svarer til at $c_i = 0$ for alle i, og er derfor inkluderet - dermed er kravet at mindst et par, (c_i, c_j) skal være forskellige ækvivalent med begge antagelser

C Baggrundsteori om stabilisator koder

I dette appendiks vil vi i gennemgå den mest grundlæggende teori om stabilisator koder - det er baseret relativt tæt på gennemgangen i [2] afsnit 7.9. Det vil lede frem til de krav som er opstillede i starten af afsnit 4.2.1.

C.1 Stabilisator formalismen

Fremgangsmåden for at forstå denne formalisme vil være opdelt i tre skridt:

- Først er vi nødt til at lave nogle mere præcise bestragninger omkring kravene til en kvantefejlretningskode.
- Derefter vil vi lave nogle formelle udledninger som vil resultere i et ekstremt stærkt værktøj til at definere effektive fejlretnings koder.
- Slutteligt vil vi bevise visse vigtige egenskaber ved de fejlretningskoder, som er lavet vha. dette værktøj, bl.a. at de faktisk opfylder kravene som må stilles til en effektiv fejlretningskode

C.1.1 Gennerelle egenskaber ved fejlretningskoder

Vi har i afsnit 4.1.1 argumenteret for, at enhver fejl kan ekspanderes til en række fejloperatorer på formen $\{I, X, Y, Z\}^{\otimes n}$. Hvilke krav er det tilstrækkeligt at stille til en kvantefejlretningskode, for at vi kan rette alle fejl i en given mængde af fejloperatorer, $\mathcal{E}$? Hvis vi for ethvert par af fejloperatorer, $E_a, E_b \in \mathcal{E}$, har, at en fejlkode bestående af kodeordene $|\alpha>, |\beta>, \ldots$ opfylder at

$$< \alpha|E_b^\dagger E_a|\beta >= \delta_{\alpha,\beta}\delta_{a,b} \tag{43}$$

, så sikrer fejlkoden imod alle fejl i $\mathcal{E}$. Det gælder jo nemlig så, at en måling kan skelne mellem alle forskellige fejloperatorer, idet (43) jo betyder at det underrum (af det 2^n-dimentionale hilbertrum $\mathcal{H}_{2^n}$), som udgør vores fejlretningskode, bliver transformeret af fejloperatoren E_a til et fejlunderrum, som er orthogonalt på alle andre fejlunderrum, dvs. underrum som genereres af andre fejloperatorer end E_a, eller være en given fejloperator andvendt på et andet kodeord. Dette gør det muligt at konstruere en måling, som kan skelne imellem hvilken fejl der er opstået (sålænge vi kun betragter fejlene fra $\mathcal{E}$) og hvilket kodeord vi oprindeligt kom fra - en måling vil kolapse hukommelsen til et af underrummene, og idet underummene er ortogonale kan vi være sikre på, at hukommelsen er i et underrum som *kun* svarer til den målte fejl. Den anden deltafunktion i (43) ($\delta_{\alpha,\beta}$) gør det muligt at konstruere en operator, baseret på målingen af hvilken fejl der opstod, som sender hukommelsen tilbage til fejlretningskodens underrum, nemlig $R_a = \sum_i |i >< i|E_a^\dagger$. At (43) er tilstrækkelig for at denne operator reparerer tilstanden ses umiddelbart idet $R_a E_a|\alpha >= \sum_i |i >< i|E_a^\dagger E_a|\alpha >= \sum_i |i > \delta_{i,\alpha} = |\alpha >$.
Det bør en passent nævnes, at imens (43) er tilstrækkelig for at en fejlretningskode retter alle fejl i $\mathcal{E}$ korrekt, så er den ikke nødvendig - et eksempel på dette er faktisk Shor koden, idet f.eks. $< \bar{1}|Z_1^\dagger Z_2|\bar{1} >= 1 \neq 0$. (hvor $|\bar{1} >$ skal forståes

som den logiske qubit i tilstanden 1). Det kan vises at det er nødvendigt og tilstrækkeligt at $< \alpha|E_b^\dagger E_a|\beta >= \delta_{\alpha,\beta} C_{a,b}$, hvor $C_{a,b}$ er en hermittisk matrice - vi vil dog ikke gennemgå beviset for dette, idet den kode vi ønsker at nå frem til - $[[5,1,3]]$ koden - faktisk opfylder (43), og den siges dermed at være en ikke-udartet kvantefejlretningskode.

C.1.2 Stabilisator koder

Foregående udledninger gav os en bedre forståelse af kravene til en fejlretningskode, men vi mangler stadigvæk en måde, hvorpå vi kan konstruere en specifik kode udfra dette. Det vil vi nu rette op på: Vi vil argumentere for eksistensen af en række uafhængige operatorer (kaldet generatorer) som har den egenskab, at de genererer en fejlretnings kode: Hver af operatorerne (f.eks. M) har den egenskab at hvis $|\Psi >$ er et korrekt kodeord, så er $M|\Psi >= |\Psi >$, mens en fejl (som ligger inden for de begrænsninger som vi specificerer, f.eks. kun er på en fysisk qubit) vil forårsage at en bestemt mængde af generatorerne får egenværdien -1 ($M|\Psi >= -|Psi >$). Ved at måle disse operatorers værdi vil vi kunne bestemme hvilken fejl der er opstået og dermed kunne rette den. Eksempler på sådanne operatorer er for så vidt de operatorer, vi målte værdien af i Shor koden, men vi vil her formulere en måde at finde bedre operatorer, og herved forminske antallet af fysiske qubits. Før vi kan gå igang med det, er vi nødt til at forstå nogle basale egenskaber ved den generelle mængde af fejloperatorer på en samling af n fysiske qubits, som vi nu vil definere som $\mathcal{G}_n \equiv \{A|A \in \{\pm I, \pm X, \pm Y, \pm Z\}^{\otimes n}\}$ Først og fremmest ses følgende

Sætning 3 *Mængden $\mathcal{G}_n$ med almindelig operator anvendelse som komposition udgør en gruppe.*

Bevis: Først husker vi, at enhver anvendelse af en operator, som er et tensorprodukt af enkeltqubitoperatorer, på en en anden sådan operator, blot er tensorproduktet af de respektive enkeltqubitoperatorer anvendt qubitvist på hindanden. Dermed er det nok at vise eksistensen af en invers, og at kompositionen er lukket, på enkeltqubitoperatorerne. Først finder vi den inverse til en operator: Idet vi ved inspektion ser at $(\pm X)^2 = (\pm Z)^2 = I$, $(-I)^2 = I$ og $\pm Y(\mp Y) = I$, ser vi, at alle enkeltqubitoperatorer, og dermed alle elementer i $\mathcal{G}_n$, har en invers.
Dernæst viser vi at mængden er lukket under anvendelse af kompositionen ved at checke alle enkeltqubitoperatorers opførsel under komposition: $X^2 = I$, $XY = XXZ = Z$, $XZ = Y$, $YX = XZX = -XXZ$ (da XZ=-ZX) $= -Z$, $YY = XZXZ = -XXZZ = -I$, $YZ = XZZ = X$, $ZX = -XZ = -Y$, $ZY = ZXZ = -ZZX = -X$, og $ZZ = I$. Komposition med I og $-I$ er indlysende lukket, idet $A \in \mathcal{G}_n \leftrightarrow -A \in \mathcal{G}_n$. Dermed er $\mathcal{G}_n$ lukket under kompositionen □
Vi ser dernæst at alle operatorer i $\mathcal{G}_n$ er unitære, idet enkeltqubitoperatorerne er det, og idet $(\{\pm I, \pm X, \pm Y, \pm Z\}^{\otimes n})^\dagger = \{(\pm I)^\dagger, (\pm X)^\dagger, (\pm Y)^\dagger, (\pm Z)^\dagger\}^{\otimes n}$.
Enhver operator i $\mathcal{G}_n$ er enten hermitisk, hvis der er et lige antal operatorer i tensorproduktet som er lig Y, eller antihermittisk, hvis der er et ulige antal

operatorer i tensorproduktet som er lig Y. Dette ses efter samme model som unitariteten (dvs. vi ser på enkeltqubitoperatorerne), og ved her at indse at $(\pm I)^\dagger = (\pm I)$, $(\pm X)^\dagger = (\pm X)$ og $(\pm Z)^\dagger = (\pm Z)$, men $Y^\dagger = -Y \leftrightarrow (-Y)^\dagger = Y$, hvorfra vi udleder udsagnet om operatorens evt. antihermitisitet (da hver Y i tensorproduktet giver en faktor -1 under hermittisk adjungering).
Som en konsekvens af ovenstående gælder det at for alle $M \in \mathcal{G}_n$ at $M^2 = \pm I \equiv \pm I^{\otimes n}$, idet hvis M er unitær og hermitisk, er $M = M^{-1}$, hvorfor $M^2 = I$, og hvis M er unitær og antihermitisk, er $M = -M^{-1}$, og dermed $M^2 = -I$.
Idet man ved inspektion kan se, at enkeltqubitoperatorerne enten komuterer, eller antikomuterer, (idet alle operatorer kan skrives som et produkt af X og Z, og *de* antikomuterer) er det klart, at to elementer A og B fra $\mathcal{E}$ enten komuterer eller antikomuterer.
Vi er nu klar til at definere vore generatorer som et sæt af basisvektorer for en abelsk undergruppe,$\mathcal{S}$, af $\mathcal{G}_n$, som stabiliserer et underrum ($H_\mathcal{S}$) af vort generelle 2^n dimentionale hilbertrum, dvs.

$$|\Psi >\in H_\mathcal{S} \leftrightarrow M|\Psi >= |\Psi > \text{ for alle } M \in \mathcal{S} \tag{44}$$

At undergruppen er abelsk følger af, at (44) implicerer at alle elementerne i $\mathcal{S}$ kan diagonaliseres samtidigt (da $|\Psi >$ jo er samtidige egenvektorer for alle $M \in \mathcal{S}$). En anden vigtig konsekvens af (44) er at det for alle stabilisatorer gælder at $M^2 = I$, idet M ellers ikke kunne have egenværdien 1 - eneste anden mulighed er $M^2 = -I$, og det er imodstrid med, at nogen egenvektor af M har 1 som egenværdi, idet vi så ville have $M^2|\Psi >= M|\Psi >= |\Psi >$. Dette har den vigtige konsekvens, jf. ovenstånde egenskaber, at M er hermitisk.
Man kan nu argumentere for, at følgende udsagn er ækvivalente:

1. $H_\mathcal{S}$ er en $[[n, k, d]]$ kvantefejlretningskode (defineret efter gennemgangen af Shor koden).

2. Der er $n - k$ stabilisatorer.

Beviset for denne sætning er ikke så besværligt, men vi udelader det af pladshensyn - vi henviser blot til [2] afsnit 7.9.

C.1.3 Egenskaber ved stabilisatorkoder

Vi vil nu bevise en række egenskaber ved ovenstående koder. Observer først at enhver fejloperator, idet den kan ekspanderes i elementer af $\mathcal{G}_n$, ved en måling af generatorenes egenværdier, vil kollapse til en tilstand, som enten komuterer eller antikomuterer med en given stabilisator, simpelthen fordi alle elementer i $\mathcal{G}_n$ gør det - nålingen af generatorenes egenværdier svarer altså til en projektion af fejloperatoren ned på en operator i $\mathcal{G}_n$. Dette leder os frem til en måde at identificere hvilke fejloperatorer der kan rettes vha. en given stabilisator: Hvis der nemlig eksisterer en generator $M_i \in \mathcal{S}$ om hvilken det gælder at M_i og en fejloperator E antikomuterer, så vil en måling af M_i kunne afsløre hvorvidt fejlen E er indtruffet, idet vi så for alle $|\Psi >\in H_\mathcal{S}$ har at $M_iE|\Psi >= -EM_i|\Psi >= -E|\Psi >$. Husk fra (43) at det, vi kræver for at en mængde af fejloperatorer, $\mathscr{E}$, kan

rettes af vores kode $H_{\mathcal{S}}$, er at der for alle $E_a, E_b \in \mathscr{E}$ gælder, at hvis $|\Psi> \in H_{\mathcal{S}}$ (og sættet af kodeord iøvrigt er ortonormaliseret) er $< \Psi|E_a^\dagger E_b|\Psi >= 0$. Vi vil nu vise at dette vil være opfyldt, hvis der eksisterer mindst en $M \in \mathcal{S}$ som antikomuterer med $E_a^\dagger E_b$:

$$< \Psi|E_a^\dagger E_b|\Psi >=< \Psi|E_a^\dagger E_b M|\Psi >= - < \Psi|M E_a^\dagger E_b|\Psi >= \tag{45}$$
$$\text{(da M er hermittisk)} - < \Psi|E_a^\dagger E_b|\Psi > \leftrightarrow < \Psi|E_a^\dagger E_b|\Psi >= 0$$

Lad os nu antage, at der, for hver fejl i den fejlmængde vi ønsker at beskytte os imod, er et sæt af generatorerne som antikomuterer med denne fejl. Da vil forudsætningerne for (45) være opfyldt, hvis der til hver fejl netop er et distinkt sæt af generatorer som antikomuterer med denne fejls operator, dvs. der er ikke to forskellige fejl, som antikomuterer med det samme sæt af generatorer. Da vil der i (45) nemlig altid være en generator som antikomuterer med E_b, men komuterer med E_a (og dermed komuterer med $E_a^\dagger$, da E_a jo er enten hermitisk eller antihermitisk), og dermed vil denne generator antikomutere med $E_a^\dagger E_b$.
Det bør som en sidebemærkning nævnes, at det er muligt at designe koder, som ikke opfylder dette krav, men som alligevel er stabilisator koder. Det er de koder som er udartede (se nederst i underafsnittet "gennerelle egenskaber ved fejlretningskoder"). Vi vil ikke gå videre ind i teorien om disse, idet, som nævnt, [[5,1,3]] koden, som vi vil benytte, ikke er udartet.
Tilsidst må det nævnes at en af de store fordele ved stabilisator koder er, at det er nemt at bestemme dens afstand, d. Vi skal nemlig blot forlange/kontrolere, at alle mulige fejloperatorer med en vægt mindre end d antikomuterer med mindst en generator for koden. Dette betyder jo nemlig, at man for at komme fra et lovligt kodeord til et andet skal have en operator med mindst vægt d, idet et skift fra et kodeord til et andet jo ikke kan antikomutere med nogen generator (da både "start"-kodeordet og "slut"-kodeordet har egenværdien $+1$ for alle generatorer).

D Fejltollerante logiske operationer

I dette appendiks vil vi gennemgå teorien for fejltollerante logiske operationer, samt præsentere de metoder som anvendes i konstruktionen af de fejltollerante operationer. Teorien er hentet fra [6] afsnit II og III.
Matematisk ser vi (jf. [6]) at for alle generatorer i $M_i \in \mathcal{S}$ kan effekten af den logiske gate U formuleres som

$$UM_i|\Psi>=UM_iU^\dagger U|\Psi> \tag{46}$$

Dermed kan overstående krav for U's fejltollerance formuleres i følgende theorem:

Sætning 4 *For alle $M_i \in \mathcal{S}$ skal gælde, at $|\Psi>$ er en egenfunktion med egenværdi $+1$ for M_i (sand for alle tilstande i $H_\mathcal{S}$) hvis og kun hvis $U|\Psi>$ er en egenfunktion for $UM_iU^\dagger$ med egenværdi $+1$.*

Herudfra følger umiddelbart at $\bar{Z}$ og $\bar{X}$ er fejltollerante, idet de begge er hermittiske, komuterer med alle $M_i \in S$ og opfylder $\bar{Z}^2 = I$ hhv. $\bar{X}^2 = I$. Dermed haves at for alle generatorer $UM_iU^\dagger = ZM_iZ^\dagger = ZM_iZ = M_iZ^2 = M_i$ og tilsvarende for $\bar{X}$.
Omvendt ser vi, at den bitvise hadamard transformation, dvs. operationen $\bar{H} = H^{\otimes 5}$ *ikke* er fejltollerant, idet f.eks.

$$\bar{H}M_1\bar{H}^\dagger = HXH^\dagger \otimes HZH^\dagger \otimes HZH^\dagger \otimes HXH^\dagger \otimes HIH^\dagger = Z\otimes X\otimes X\otimes Z\otimes I \tag{47}$$

som ses ikke at komutere med $M_3 \equiv X \otimes I \otimes X \otimes Z \otimes Z$.
Hvad sker der med de logiske qubits tilstand, dvs. den logiske hukommelse, hvis vi anvender en transformation U på den? Lad os betragte en situation, hvor vi har bestemt både $U\bar{X}U^\dagger$ og $U\bar{Z}U^\dagger$ (hvis der er flere logiske qubits involverede i U, så bestemmer vi transformationen af samtlige logiske qubits $\bar{X}$ og $\bar{Z}$ operationer). Specielt vil vi betragte $U|\bar{0}>=U\bar{Z}|\bar{0}>=U\bar{Z}U^\dagger U|\bar{0}>$, eller med andre ord, $U|\bar{0}>$ er en egentilstand af $U\bar{Z}U^\dagger$ med egenværdien $+1$, og vi kan således bestemme $U|\bar{0}>$. Tilsvarende er $U|\bar{1}>=U\bar{X}|\bar{0}>=U\bar{X}U^\dagger U|\bar{0}>$, og eftersom vi kender både $U\bar{X}U^\dagger$ (pr. antagelse), og $U|\bar{0}>$, kan vi nu finde $U|\bar{1}>$. Men så kender vi U's effekt på både $|\bar{0}>$ og $|\bar{1}>$, hvorfor vi kan bestemme U's effekt på en arbitrær tilstand $|\bar{\Psi}>=\alpha|\bar{0}>+\beta|\bar{1}>$, *udelukkende* udfra transformationerne af $\bar{X}$ og $\bar{Z}$.

Målinger i konstruktion af fejltollerante gates

Det viser sig, at man ikke kan konstruere de fejltollerante gates som vi har brug for til Shor's algoritme, udelukkende ved at finde smarte transformationer og anvende teknikkerne givet ovenfor. For at kunne opnå det ønskede er vi nødt til at have et sidste værktøj, nemlig måling af operatorer, som antikomuterer med dele af sættet af generatorer for stabilisatoren. Vi vil vise, at effekten af dette er at vi ændrer en delmængde af de basale logiske operatione ($\bar{X}$'erne og $\bar{Z}$'erne), hvilket er vigtigt, idet vi så jf. ovenstående ændrer den transformation U's effekt på den logiske hukommelse - ved at foretage "smarte" målinger kan

vi ændre den til de transformationer vi ønsker at skabe. Det er klart, at de målinger vi foretager er fejltollerante som beskrevet i 4.3.1.
Lad os antage at vi har fundet en operator A, som kan skrives som et tensorprodukt af $\{I, X, Y, Z\}$, og som antikomuterer med f.eks. M_1. Vi ønsker at den eneste generator som A antikomuterer med netop er M_1, så hvis A antikomuterer med andre generatorer, f.eks. M_2, løses dette ved at erstatte M_2 med M_1M_2, der jo også er en generator: Idet A antikomuterer med både M_1 og M_2 komuterer den med M_1M_2. Et delmål med målingen er at gøre A til en ny generator, dvs. den skal erstatte M_1. Dette er opnået, hvis resultatet af målingen var $+1$, men ikke hvis den var -1 - hvis det sidste skulle være tilfældet anvender vi blot M_1 på tilstanden, idet vi så har (under antagelsen $A|\Psi>=-|\Psi>$):

$$AM_1|\Psi>=-M_1A|\Psi>=-M_1(-1)|\Psi>=M_1|\Psi> \tag{48}$$

hvorfor A nu har egenværdien $+1$, samtidigt med at $\{M_2, M_3, M_4\}$ stadig er generatorer, idet M_i'erne jo komuterer. Ligesom Gottesman i [6] vil jeg for forståelighedens skyld, herunder skrive "måle operatoren A", når jeg præcist mener "måle operatoren A og anvende den operator den erstatter, hvis resultatet af målingen var -1"
Lad os nu undersøge, hvorledes målingen har påvirket $\bar{X}$'erne og $\bar{Z}$'erne, og lad os for nemheds skyld antage at der kun er én af hver. Hvis de begge komuterer med A sker der ingen ændringer: De opfylder stadig kravene (begge komuterer med alle generatorer, og de antikomuterer indbyrdes). Men hvis nogen af dem antikomuterer med A komuterer de selvsagt ikke længere med alle generatorer, og må derfor ændres efter samme princip som de af generatorerne der evt. heller ikke komuterede med A: Vi lader M_1 virke på de antikomuterene $\bar{X}$ og/eller $\bar{Z}$, og efter samme argument som ovenfor komuterer disse nye operatorer, f.eks. $M_1\bar{Z}$, med alle generatorer. Samtidigt antikomuterer de i alle tilfælde, da både $\bar{Z}$ og $\bar{X}$ komuterer med M_1. Vi har nu opnået at kunne ændre $\bar{Z}$ og $\bar{X}$, og således har vi nu det sidste basale værktøj som skal anvendes i fejltollerancen af Shor's algoritme.

Litteratur

[1] Adriano Barenco et al.: *Elementary gates for quantum computation* Phys. rev. A Volume 42, nummer 5 (nov. 1995)

[2] John Preskill: *Lecture notes for Physics 219* (http://www.theory.caltech.edu/%7Epreskill/ph219/index.html#lecture)

[3] Beckman, Chari, Devabhaktuni og Preskill *Efficient networks for quantum factoring* Phys. Rev. A Volume 54, nummer 2 (aug. 1996)

[4] Vedral, Barenco og Ekert: *Quantum networks for elementary arithmetic operations* Phys. rev. A Volume 54 nummer 1 (jul. 1996)

[5] Michael Nielsen og Isaac L. Chuang: *Quantum computation and quantum information* (2002)

[6] Daniel Gottesman: *Theory of fault-tolerant quantum computation* Phys. Rev. A Volume 57, nummer 1 (Jan. 1998)

[7] P. Oscar Boykin et al.: *On Universal and Fault-Tolerant Quantum Computation* quant-ph/9906054v1 (Jun. 1999)

[8] P. W. Shor og D. P. DiVincenzo Phys. Rev Letters 77 (1996)

[9] A. Kitaev: *Quantum Computaions: Algorithms and error correction*, Russian Math. Surveys 52 (1997)

[10] A. Barenco et al: *Approximate quantum Fourier transform and decoherence*, Phys. Rev. A, Volume 54, nummer 1 (Jul. 1996)

[11] T.H. Cormen et al.: *Introduction to Algorithms, 2nd edition* (2001)

[12] P. W. Shor: *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*,quant-ph/9508027

[13] P. W. Shor: *Scheme for reducing decoherence in quantum computer memory* Phys. Rev. A, Volume 52 (1995)