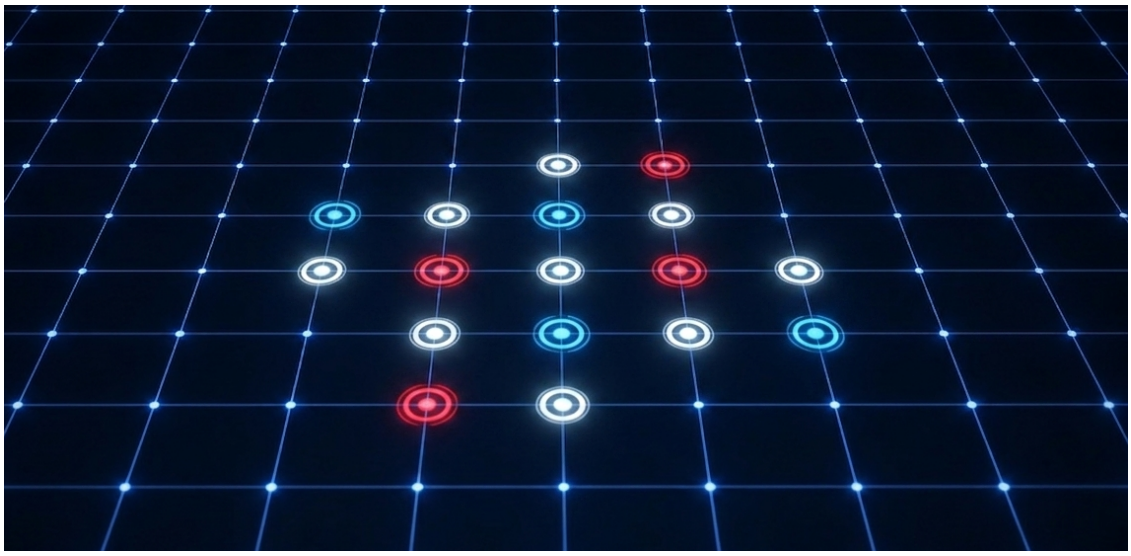
CHALMERS



Lattice Surgery med kvantfelskorrektion på IBM Nighthawk r1

Sammanflätning av logiska felkorrigerade kvantbitar på ett kvantchip med kvadratisk topologi

Kandidatarbete

Max Lilja, Jacob Lorentzon, Ludvig Molinder,
Jonathan Nguyen, Minna Nikolic, Gustav Tollander

INSTITUTIONEN FÖR FYSIK OCH ASTRONOMI

CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2026
www.chalmers.se

Kandidatarbete 2026

Lattice Surgery med kvantfelskorrektion på IBM Nighthawk r1

Sammanflätning av logiska felkorrigerade kvantbitar på ett kvantchip med kvadratisk topologi

Lattice surgery with quantum error correction on IBM Nighthawk r1

Entanglement of error corrected logical qubits on a quantum processor with square topology

Max Lilja, Jacob Lorentzon, Ludvig Molinder
Jonathan Nguyen, Minna Nikolic, Gustav Tollander

CHALMERS

Institutionen för fysik och astronomi
Chalmers Tekniska Högskola
Göteborg, Sverige 2026

Lattice Surgery med kvantfelskorrektion på IBM Nighthawk r1
Sammanflätning av logiska felkorrigerade kvantbitar på ett kvantchip med kvadratisk topologi
Max Lilja, Jacob Lorentzon, Ludvig Molinder
Jonathan Nguyen, Minna Nikolic, Gustav Tollander

Handledare: Mats Granath, Institutionen för fysik vid Göteborgs universitet
Examinator: Jan Swenson, Institutionen för fysik och astronomi vid Chalmers tekniska högskola

Kandidatarbete 2026
Institutionen för fysik och astronomi
Chalmers tekniska högskola
SE-412 96 Göteborg
Telefon +46 31 772 1000

Omslagsbild: Visualisering av en 3×3-ytkod på ett kvadratiskt gitter, genererad med AI-bildverktyget Nano Banana 2 via Gemini den 7 maj 2026.

Skriven i LaTeX
Göteborg, Sverige 2026

Lattice Surgery med kvantfelskorrektion på IBM Nighthawk r1
Sammanflätning av logiska felkorrigerade kvantbitar på ett kvantchip med kvadratisk topologi
MAX LILJA, JACOB LORENTZON, LUDVIG MOLINDER
JONATHAN NGUYEN, MINNA NIKOLIC, GUSTAV TOLLANDER
Institutionen för fysik och astronomi
Chalmers tekniska högskola

Sammandrag

Kvantdatorer har stor potential att utföra särskilt kraftfulla beräkningar för vissa typer av problem, men är i sin nuvarande form bruskänsliga, särskilt när problemstorleken ökar. Därför används kvantfelskorrektion, där kvanttillstånd lagras med extra redundans som logiska tillstånd. Genom upprepade rundor av partiella mätningar kan utfallen avkodas och fel identifieras, utan att det logiska tillståndet kollapsar. I det här arbetet implementerades kvantfelskorrektion med roterade ytkoder, där avkodningen gjordes med Minimum Weight Perfect Matching. Därefter tillämpades *lattice surgery*, för att flera ytkoder skulle kunna interagera. Ytkoden testades både genom simulering med slumpmässiga injicerade fel och genom att köras på den fysiska kvantprocessorn *IBM Nighthawk r1*. På kvantprocessorn testades även hur lattice surgery kan användas för att bilda sammanflätade logiska Belltillstånd, och den logiska CNOT-grinden testades genom simulering. Enskilda ytkoder och avkodning på kvantprocessorn resulterade i 5–20% logiska fel för 2–14 rundor, och när lattice surgery användes för att skapa ett Belltillstånd erhölls de teoretiskt förväntade resultaten i över 85% av fallen vid 12 totala rundor. Resultaten tyder på att roterade ytkoder och lattice surgery har god potential att leda till effektiv, feltolerant kvantberäkning i takt med att större kvantprocessorer blir tillgängliga.

Abstract

Quantum computers have great potential to perform particularly powerful calculations for certain types of problems, but are in their current form sensitive to noise, especially as the problem size increases. Therefore, quantum error correction is used, encoding states with extra redundancy into logical states. Through repeated rounds of partial measurements, outcomes are decoded and errors identified without collapsing the logical state. In this work, quantum error correction was implemented using rotated surface codes, decoded using the Minimum Weight Perfect Matching algorithm. Subsequently, *lattice surgery* was applied to enable interaction between multiple surface codes. The surface code was evaluated both through simulation with random injected errors and by being executed on the physical quantum processor *IBM Nighthawk r1*. The quantum processor was also used to test how lattice surgery can be used to form entangled logical Bell states, and the logical CNOT gate was tested in simulation. Individual surface codes and decoding on the quantum processor resulted in 5–20% logical errors for 2–14 rounds, and when lattice surgery was used to create a Bell state, the theoretically expected results were obtained in over 85% of the cases for 12 total rounds. The results suggest that rotated surface codes and lattice surgery have good potential to lead to efficient, fault-tolerant quantum computing as larger quantum processors become available.

Nyckelord: kvantdatorer, felkorrigering, lattice surgery, IBM Nighthawk r1, ytkoder.
Keywords: quantum computing, error correction, lattice surgery, IBM Nighthawk r1, surface codes.

Förord

Vi vill ge ett stort tack till vår handledare Mats Granath samt Moritz Lange för deras vägledning och stöd under arbetets gång. Deras engagemang och svar på våra frågor har varit ovärderliga för genomförandet av detta projekt.

Användningen av IBM:s fysiska kvantdator, med tillhörande molntjänster, möjliggjordes av stöd från WACQT Quantum Technology Testbed som drivs av Chalmers Next Labs och finansieras av Knut och Alice Wallenbergs stiftelse.

Max, Jacob, Ludvig, Jonathan, Minna, Gustav
Göteborg, Maj 2026

Ordlista

Nedan följer en lista över de facktermer och begrepp som används i rapporten.

Ancilla-kvantbit	En hjälpkvantbit som används vid syndrommätningar.
Kodavstånd	Minsta antalet fysiska fel som krävs för ett logiskt fel.
Ytkod	Topologisk kvantfelskorrigerande kod definierad på ett tvådimensionellt gitter.
Roterad ytkod	En variant av ytkoden där kvantbitar och stabilisatorer arrangeras i ett roterat gitter för att minska antalet fysiska kvantbitar utan att ändra kodavståndet.
Paulioperator	En av operatorerna X, Y eller Z som används för att beskriva kvantoperationer och fel.
Bit-flip-fel	Ett kvantfel motsvarande en oavsiktlig Pauli-X-operation.
Fas-fel	Ett kvantfel motsvarande en oavsiktlig Pauli-Z-operation.
Mätfel	Ett felaktigt resultat av en kvantmätning, utan att det underliggande tillståndet nödvändigtvis är fel.
Unitär operator	Linjär operator U som uppfyller $U^\dagger U = I$.
Hermitesk operator	Linjär operator U som uppfyller $U^\dagger = U$.
Blochsfär	Geometrisk representation av en kvantbits kvanttillstånd.
Kvantbit	Ett tvånivå-kvantsystem, representeras som en superposition av 0 och 1.
Kvantgrindar	Unitära operatorer som agerar på en eller flera kvantbitar.
Kvantkrets	En sekvens av kvantgrindar och mätningar.
Clifford-grindar	Operatorer som transformerar Paulioperatorer till andra Paulioperatorer via konjugering.
Belltillstånd	Kvanttillstånd för två kvantbitar som utgör de enklaste exemplen på kvantsammanflätning.
MWPM	*Minimum-Weight Perfect Matching*, en avkodningsalgoritm som kan användas för ytkoder, och bygger på grafteori.
Logisk kvantbit	En stabil, felkorrigerad enhet av kvantinformation som skapas med flera fysiska kvantbitar.
Logisk operator	En operator som verkar på en logisk kvantbit genom att stanna i kodrummet.
Stabilisatorer	Operator vars egenrum definierar kodrummet i en stabilisatorkod.
X-stabilisatorer	Stabilisator som består av Pauli-X-operatorer

Z-stabilisatorer	Stabilisator som består av Pauli-Z-operatorer
Stabilisatorrunda	Samtidig mätning av alla stabilisatorer hos en ytkod, vilket upprepas i tid.
Lattice surgery	En feltolerant metod för att utföra logiska operationer genom att sammanfoga och dela upp ytkoder.
Merge	En operation i lattice surgery där två ytkoder sammanfogas för att utföra en gemensam mätning mellan logiska kvantbitar.
Split	En operation i lattice surgery där en ytkod delas upp i flera delar för att skapa separata logiska kvantbitar.
Konventionell form	Form av lattice surgery där en rad mellanliggande kvantbitar förekommer mellan ytkoderna.
Kompakt form	Alternativ form av lattice surgery där vissa fysiska ancillakvantbitar delas mellan ytkoderna.
Koherenta fel	Fel som uppstår på grund av unitära operatorer eller kvantgrindar
Inkoherent fel	Fel som uppstår på grund av systemets interaktion med omgivningen (dekoherens)
Beräkningsbas	Den standardbas som består av kvanttillstånden $\|0\rangle$ och $\|1\rangle$ och används för mätningar i Z-basen.
Hadamardbas	Alternativ bas som består av tillstånden $\|+\rangle$ och $\|-\rangle$. Erhålls genom att applicera Hadamardgrinden på beräkningsbasen.
Detektor	Ändring av stabilisatorvärde mellan stabilisatorrundor, vilket utgör kanter i MWPM-grafen.
Detektordensitet	Genomsnittliga antalet detektorer som uppkommer per stabilisatorrunda.
Observabel	Mätbar storhet hos ett kvanttillstånd vars ändringar spåras i MWPM-grafen.
Spatial kant	Kant i MWPM-grafen som representerar fel på datakvantbitar.
Temporal kant	Kant i MWPM-grafen som representerar mätfel eller ett Pauli-fel på en ancilla.
Universell mängd	Tillräcklig uppsättning kvantgrindar för att implementera alla möjliga kvantgrindar.
Basgrind	Kvantgrind som en kvantprocessor direkt stödjer, som andra grindar måste transpileras till.
Hadamardgrinden	Kvantgrind som avbildar beräkningsbasen på Hadamardbasen.
CNOT	Kvantgrind och reversibel 2-bitoperation som inverterar ena biten ifall andra är 1.

Innehåll

Ordlista **viii**

1 Inledning **1**
1.1 Syfte . . . 2
1.2 Avgränsningar . . . 2

2 Teori **3**
2.1 Kvantbitar . . . 3
2.2 Pauligruppen $\mathcal{P}_n$ . . . 4
2.3 Kvantkretsar och kvantberäkning . . . 5
2.3.1 Unitära grindar . . . 5
2.3.2 Clifford-gruppen och Gottesman-Knill-teoremet . . . 6
2.4 Stabilisatorformalismen . . . 6
2.4.1 Stabilisatortillstånd och underrum . . . 6
2.4.2 Projektiva mätningar . . . 8
2.5 Detektion och korrigering av fel med stabilisatorkoder . . . 8
2.5.1 Enkel repetitionskod . . . 9
2.5.2 Ytkoder . . . 10
2.6 Förberedelse för stabilisatormätningar . . . 12
2.7 Avkodning . . . 13
2.7.1 Matchningsgraf . . . 13
2.7.2 Syndromgrafen . . . 14
2.8 Lattice surgery . . . 15
2.8.1 Merging och splittning . . . 16
2.8.2 Logisk CNOT . . . 18
2.9 Kvantprocessor . . . 18

3 Metod **20**
3.1 Ytkoder . . . 20
3.2 Lattice surgery . . . 20
3.3 Körning på kvantdator . . . 21
3.4 Logisk CNOT . . . 21
3.5 Avkodning . . . 22

4 Resultat **23**
4.1 Ideal lattice surgery . . . 23
4.2 Simuleringar med brusmodell . . . 24
4.2.1 Lattice surgery . . . 24

4.3 Körning på IBM Nighthawk r1 25

5 Diskussion **29**
5.1 Simuleringar 29
5.1.1 Lattice surgery 29
5.2 Fysisk körning 29
5.3 Felkällor och potentiella förbättringar 30
5.4 Samhälleliga och etiska aspekter 31
5.5 Slutsats 31

Litteratur **32**

A Stabilisatorer **I**

B Kod **III**

C MWPM-grafer **IV**

D AI-verktyg **VI**

1
Inledning

Kvantdatorer har visats vara betydligt mer effektiva på vissa sorters problem än klassiska datorer, såsom Shors algoritm [1] för primtalsfaktorisering. Kvantdatorer använder kvantbitar (eng. qubits) som kan anta kontinuerliga superpositioner av noll och ett. Eftersom kvantbitar, till skillnad från klassiska bitar, kan sammanflätas, ökar tillståndsrummets dimension exponentiellt snarare än linjärt med antalet kvantbitar. Sammanflätningen måste dock utnyttjas för att mäta det önskade utfallet med tillräckligt hög sannolikhet [2]. Med Grovers algoritm [3] erhålls kvadratisk förbättring för generella diskreta optimeringsproblem, medan algoritmer som Shors, HHL [4] och VQE [5] kan ge exponentiell förbättring för faktorisering, delvis linjär ekvationslösning, respektive kvantsimulationer.

Jämfört med klassiska datorer är kvantdatorer betydligt känsligare för brus, vilket medför en högre felsannolikhet och begränsar den möjliga komplexiteten i kretsar [6]. En lösning är kvantfelskorrektion där ytterligare redundans läggs till i systemet. Genom att utföra vissa partiella mätningar är det möjligt att avgöra huruvida kvantfel har uppstått utan att kollapsa vågfunktionen och därmed förstöra det lagrade kvanttillståndet, samt åtgärda felen [7]. Kvanttillstånd kan dock, enligt icke-kloningsteoremet, inte dupliceras [8]. Felkorrektionsmetoder behöver därför vara särskilt anpassade för kvantdatorer.

Shor utvecklade 1995 en metod [9] för att korrigera fel på kvantkretsar genom att använda kodning (eng. encoding) som betraktar flera kvantbitar som en enda enhet, kallad en logisk kvantbit (eng. logical qubit). Genom att sammanfläta datakvantbitarna som den logiska kvantbiten utgörs av med andra kvantbitar som kallas ancillor (eng. ancillas), kan felen på den logiska kvantbiten identifieras med hjälp av en avkodningsalgoritm som utförs av en klassisk dator. De logiska kvantbitarna kan minska felen markant i en kvantkrets med mycket brus, givet att de fysiska kvantbitarnas felsannolikhet ligger under ett tröskelvärde [10].

Shors kod utvecklades vidare till ytkoder (eng. surface codes) som betraktar en logisk kvantbit som ett tvådimensionellt rutnät med kvantbitar och ancillabitar mellan dem [11], men det finns andra sorters koder såsom repetitionskoder [12] och honeycomb codes [13], med olika för- och nackdelar. En ytkod definieras av dess kodavstånd (eng. distance) d som är lika med den kortaste sidlängden av ytkoden och det minsta antalet samtidiga fysiska fel som en ytkod inte kan detektera [14]. En $d \times d$ ytkod kan korrigera $\lfloor \frac{d-1}{2} \rfloor$ fel [11].

För att ytkoderna ska kunna användas i större kretsar abstraheras ytkoderna till logiska kvantbitar, och det är tillräckligt att implementera en så kallad universell mängd grindar (eng. universal gate set) av logiska operationer, för att alla möjliga kvantkretsar ska kunna köras med ytkoder. Det finns olika uppsättningar grindar som kan användas, men vanligtvis ingår CNOT-grinden [7], som för ytkoder kan implementeras med så kallad lattice surgery [15].

Avkodningen kan uppnås med olika algoritmer, exempelvis *Minimum Weight Perfect Matching*, hädanefter MWPM, där en graf skapas som relaterar olika kvantfel i både rum och tid. Från grafen beräknas en perfekt matchning som approximativt hittar den mest sannolika uppsättningen fel som uppstod [16]. Algoritmen kan effektivt köras på kort tid, men för att uppnå ännu högre avkodningsprestanda kan också grafneurala nätverk användas, på bekostnad av högre

komplexitet och körtid i avkodningen [17].

Kvantfelskorrektion har varit ett aktuellt ämne inom teknikindustrin, med flera företag som har utvecklat mjuk- och hårdvara. Ett av företagen som har drivit framåt utvecklingen av kvantdatorer är IBM med deras olika kvantchip. Den senaste kvantprocessorn IBM Nighthawk r1, som användes i det här arbetet, består av ett kvadratiskt gitter med 120 kvantbitar [18]. Att implementera ytkoder och lattice surgery på ett sådant kvantchip vore betydelsefullt för att vidareutveckla kvantdatorers felkorrigerings- och beräkningsförmåga i praktiken.

1.1 Syfte

Syftet med arbetet är att implementera och undersöka prestandan hos ytkoder som metod för kvantfelskorrektion, och framförallt hur de kan interagera med hjälp av lattice surgery. Metoden tillåter mer komplexa operationer som på sikt förväntas vara möjliga att komponera till feltoleranta kvantkretsar. En central del är därför att skriva logik för felkorrektion och testa felkorrektionen empiriskt på en fysisk kvantdator, samtidigt som arbetet fokuserar till större del på att demonstrera lattice surgery i praktiken, än på att nödvändigtvis uppnå den högsta avkodningsprestandan.

1.2 Avgränsningar

Eftersom huvudsyftet med arbetet var att implementera lattice surgery, begränsades avkodningsalgoritmen till MWPM, istället för att använda exempelvis grafneurala nätverk. För att minska komplexiteten i avkodningen användes inte heller en hel modell av kvantkretsen för att välja sannolikheterna för olika fel i grafen, utan vikterna valdes mestadels konstant. Vid lattice surgery på fysiska kvantdatorn behandlades dessutom endast Z-fel, eftersom detta var tillräckligt för testerna som kördes.

En ytterligare praktisk avgränsning som gjordes i det här arbetet var att begränsa storleken på ytkoden till $d = 3$. Kvantprocessorn *IBM Nighthawk r1* har nominellt plats för en enstaka 5×5 ytkod, men vissa fysiska anslutningar är defekta, och eftersom arbetet skulle innefatta hur flera ytkoder interagerar med varandra behövde flera ytkoder få plats på chippet samtidigt.

2
Teori

I det här avsnittet presenteras den matematiska formalismen som används för att beskriva kvantfelskorrektion och mer specifikt ytkoder. Det beskrivs även överskådligt hur avkodningen av mätresultatet fungerar. I resten av rapporten kommer för kompakthetens skull inte alla tillstånd att normaliseras, vilket är tillåtet så länge normerna divideras med vid uträkningar av sannolikheter.

2.1 Kvantbitar

Klassiska bitar, som utgör grunden i all klassisk datorlogik, kan anta två diskreta tillstånd, $\{0, 1\}$. Kvantbitar kan, till skillnad från klassiska bitar, anta en superposition av de två tillstånden [19], vilket är vad en kvantdator utnyttjar. En kvantbit är ett tvånivå-kvantsystem som beskrivs av ett tvådimensionellt komplext Hilbertrum. Det finns alltid möjlighet att hitta ett par ortonormala kvanttillstånd, exempelvis

$$|0\rangle \equiv \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad |1\rangle \equiv \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tag{2.1}$$

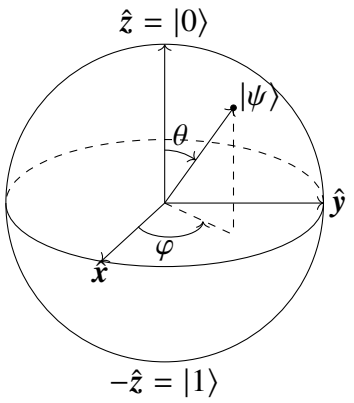
som spänner upp det tvådimensionella Hilbertrummet. Enligt superpositionsprincipen kan ett godtyckligt tillstånd hos en kvantbit skrivas som

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{2.2}$$

där sannolikhetsamplituderna $\alpha, \beta \in \mathbb{C}$ och normaliseringsvillkoret $|\alpha|^2 + |\beta|^2 = 1$ är uppfyllt. Eftersom tillståndsvektorer är definierade upp till en global fas utan fysikalisk betydelse kan ett generellt tillstånd för en kvantbit via en parametrisering skrivas som

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}\,|1\rangle\,, \tag{2.3}$$

för vinklarna θ och φ. Detta gör att kvanttillståndet kan representeras som en punkt på den så kallade Bloch-sfären, en enhetssfär i $\mathbb{R}^3$, se figur 2.1 [20, s. 100–103].



Figur 2.1: En illustration av Bloch-sfären i $\mathbb{R}^3$ som ger den geometriska representationen av en kvantbits tillstånd, $|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle$. Uppsättningen unitära operationer på enskilda Bloch-sfärer kallas för SU(2).

2.2 Pauligruppen $\mathcal{P}_n$

Paulioperatorerna används för att beskriva olika operationer, och fel som uppstår i ett kvantsystem. Operatorerna utgör en bas för rummet av 2×2 hermiteska matriser och varje unitär operator på en kvantbit kan uttryckas som en linjärkombination av dem. De fyra paulioperatorerna med avseende på basen $\{|0\rangle, |1\rangle\}$ definieras som:

$$\sigma_0 = I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \sigma_1 = X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_2 = Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_3 = Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2.4}$$

Operatorerna är både unitära ($\sigma_i^\dagger \sigma_i = I$) och hermiteska ($\sigma_i^\dagger = \sigma_i$), för $i \in \{0, 1, 2, 3\}$. Detta innebär att operatorernas egenvärden är reella; X, Y och Z har egenvärdena ± 1. Följande samband för egentillstånden för X och Z gäller: $X|+\rangle = |+\rangle$, $X|-\rangle = -|-\rangle$, $Z|0\rangle = |0\rangle$, $Z|1\rangle = -|1\rangle$. Operatorn X växlar tillståndet $|0\rangle$ mot $|1\rangle$ och tvärtom, medan Z växlar tillståndet $|+\rangle$ mot $|-\rangle$ och tvärtom [21, s. 65, 81, 174–175]. Vidare gäller att två godtyckliga Paulioperatorer antingen kommuterar eller anti-kommuterar enligt relationerna

$$[\sigma_i, \sigma_j] = 2i\epsilon_{ijk}\sigma_k, \quad \{\sigma_i, \sigma_j\} = 2\delta_{ij}I, \tag{2.5}$$

där ϵ_{ijk} betecknar Levi-Civita-symbolen, δ_{ij} är Kroneckers delta och $[\sigma_i, \sigma_j]$ respektive $\{\sigma_i, \sigma_j\}$ är kommutatorn respektive antikommutatorn av σ_i och σ_j [22]. För att skapa en grupp under multiplikation krävs det att de globala faserna $\{\pm 1, \pm i\}$ inkluderas. Pauligruppen för en enskild kvantbit definieras då som

$$\mathcal{P}_1 = \{\pm I, \pm iI, \pm X, \pm iX, \pm Y, \pm iY, \pm Z, \pm iZ\}. \tag{2.6}$$

För ett system med n kvantbitar expanderas gruppen via tensorprodukter. Pauligruppen $\mathcal{P}_n$ definieras som mängden av alla n-faldiga tensorprodukter av Paulimatriser:

$$\mathcal{P}_n = \left\{ i^k \sigma_{j_1} \otimes \sigma_{j_2} \otimes \cdots \otimes \sigma_{j_n} \mid k \in \{0, 1, 2, 3\}, j_m \in \{0, 1, 2, 3\}, \forall m \in \{1, \ldots, n\} \right\}. \tag{2.7}$$

Storleken på gruppen $|\mathcal{P}_n|$ är 4^{n+1}. Ett element ur $\mathcal{P}_n$ är en specifik kombination av Paulioperatorer som appliceras samtidigt på de n kvantbitarna i systemet. Elementen är fundamentala inom stabilisatorformalismen och ytkoder, då de används för att beskriva både stabilisatorer, logiska operatorer och de fel (X-flips och Z-flips) som systemet ska korrigera [21, s. 453–459].

2.3 Kvantkretsar och kvantberäkning

Ett klassiskt register med n bitar kan beskrivas som en samling av n binära värden, där enkla logiska operationer som NOT och AND kan appliceras på en eller två bitar för att konstruera logiska funktioner. Motsvarande koncept gäller för kvantdatorer. Ett kvantregister med n kvantbitar representerar tillståndet hos kvantdatorn. Medan ett n-bitars klassiskt tillstånd kan skrivas som ett heltal

$$i = i_{n-1}2^{n-1} + \cdots + i_1 2 + i_0, \quad i_k \in \{0, 1\}, \tag{2.8}$$

kan ett n-kvantbit-tillstånd skrivas som

$$\begin{aligned} |\psi\rangle &= \sum_{i=0}^{2^n-1} C_i \; |i_{n-1} \ldots i_0\rangle \\ &= \sum_{i_{n-1}=0}^{1} \cdots \sum_{i_1=0}^{1} \sum_{i_0=0}^{1} C_{i_{n-1}\ldots i_0} \; |i_{n-1}\rangle \otimes \cdots \otimes |i_1\rangle \otimes |i_0\rangle \,, \end{aligned} \tag{2.9}$$

där de komplexa koefficienterna C_i uppfyller $\sum_{i=0}^{2^n-1} |C_i|^2 = 1$. Tillståndet för ett n-kvantbitsregister tillhör ett 2^n-dimensionellt Hilbertrum, konstruerat som tensorprodukten av n tvådimensionella Hilbertrum (ett för varje kvantbit). Eftersom tillståndet endast definieras upp till en global fas, bestäms det av $2^n - 1$ oberoende komplexa parametrar. Mer allmänt kan ett kvanttillstånd skrivas som:

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i \, |v_i\rangle \,, \tag{2.10}$$

där tillstånden $|v_i\rangle$ utgör en ortonormerad bas. Uttryckt i *beräkningsbasen*, $\{|0\rangle, |1\rangle\}$, överensstämmer koefficienterna α_i med C_i [20, s. 105–107].

Kvantberäkningar kan beskrivas som att ett tillstånd $|\psi\rangle$ övergår till $U|\psi\rangle$ där U är en linjär operator. Eftersom tillståndsvektorer uppfyller $\langle\psi|\psi\rangle = 1$ måste sannolikheten $1 = \langle U\psi|U\psi\rangle = \langle\psi, U^\dagger U\psi\rangle$ bevaras oavsett $|\psi\rangle$, vilket kräver att U är unitär [21, s. 81-82, 94].

2.3.1 Unitära grindar

För att realisera komplexa beräkningar på fysiska kvantchip används olika så kallade *kvantgrindar*, där varje kvantgrind är en unitär operator som agerar på ett visst antal kvantbitar. De enklaste grindarna agerar på endast en kvantbit, såsom Pauli-matriserna och Hadamardgrinden som definieras enligt [21, s. 81-82]

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{2.11}$$

Hadamardgrinden är ett basbyte mellan beräkningsbasen och *Hadamardbasen*, $\{|+\rangle = |0\rangle + |1\rangle, |-\rangle = |0\rangle - |1\rangle\}$. En ytterligare grind är CNOT som verkar på ett 2-kvantbitstillstånd. CNOT uttryckt i basen $(|00\rangle, |01\rangle, |10\rangle, |11\rangle)$ motsvarar:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{2.12}$$

Den första kvantbiten kallas för *kontrollkvantbiten* och den andra för *target* [23]. Med andra ord växlar CNOT target-biten om och endast om kontrollbiten är 1, och likt andra operatorer är CNOT linjär för superpositioner.

Mängden av alla en-kvantbitsgrindar, U(2), tillsammans med CNOT-grinden utgör en universell mängd grindar, vilket innebär att varje unitär operation kan implementeras exakt med en krets från dessa grindar [21, s. 191]. Inom feltolerant kvantberäkning används ofta den diskreta grindmängden $\{H, S, \text{CNOT}, T\}$, kallad Clifford + T, eftersom den utgör en universell mängd grindar och kan approximera en kvantkrets till en godtycklig precision [24]. Grindarna S och T definieras som $S = \text{diag}(1, i)$ och $T = \text{diag}(1, e^{i\pi/4})$.

2.3.2 Clifford-gruppen och Gottesman-Knill-teoremet

Clifford-gruppen C_n definieras matematiskt som normalisatorn till Pauligruppen $\mathcal{P}_n$ i den unitära gruppen $U(2^n)$. Det innebär att en unitär operator U tillhör Clifford-gruppen om och endast om den avbildar Pauli-operatorer till andra Pauli-operatorer under konjugering, det vill säga

$$U \in C_n \iff UPU^\dagger \in \mathcal{P}_n, \quad \forall P \in \mathcal{P}_n. \tag{2.13}$$

Clifford-gruppen för n kvantbitar genereras av en diskret mängd grindar, kallade Clifford-grindar som består av Hadamardgrinden (H), fasgrinden (S) och den kontrollerade NOT-grinden (CNOT). Varje unitär operator som kan konstrueras genom en godtycklig komposition av dessa tre grindar tillhör Clifford-gruppen [25]. Ett exempel på en unitär grind som är en icke-Clifford-grind är T.

Gottesman–Knill-teoremet fastställer gränsen för när en kvantberäkning kan simuleras effektivt på en klassisk dator. Kriteriet uppfylls av alla kretsar som används här för felkorrektion, och är centralt eftersom komplexiteten annars växer exponentiellt med antalet kvantbitar, för vanliga tillståndsvektor-simuleringar. Detta formaliseras [21, s. 464] av följande teorem:

Sats 2.3.1 (Gottesman–Knill)**.** *Varje kvantberäkning som uteslutande involverar följande element kan simuleras effektivt (i polynomtid) på en klassisk dator:*

1. *Tillståndspreparering i beräkningsbasen ($|0\rangle$, $|1\rangle$).*
2. *Clifford-grindar (H, S, CNOT och Pauli-grindar).*
3. *Mätningar av observabler i Pauligruppen $\mathcal{P}_n$.*
4. *Klassisk kontroll villkorad av utfallet från dessa mätningar.*

2.4 Stabilisatorformalismen

Stabilisatorformalismen är ett kraftfullt matematiskt ramverk som förenklar beskrivningen av vissa kvanttillstånd och operationer. Istället för att beskriva ett tillstånd med en tillståndsvektor $|\psi\rangle$, som kräver 2^n komplexa amplituder, beskrivs tillståndet av de operatorer som lämnar tillståndet invariant.

2.4.1 Stabilisatortillstånd och underrum

Ett tillstånd $|\psi\rangle$ sägs stabiliseras av en operator S om $S|\psi\rangle = |\psi\rangle$. Mängden av alla sådana Pauli-operatorer $S \in \mathcal{P}_n$ som stabiliserar $|\psi\rangle$ bildar en kommutativ grupp $\mathcal{S} \subset \mathcal{P}_n$, kallad *stabilisatorgruppen*. Gruppen kan beskrivas fullständigt av en minimal uppsättning oberoende och kommuterande operatorer, som kallas generatorer. Om det finns $n - k$ generatorer

$\{g_1, g_2, \ldots, g_{n-k}\}$, definieras $\mathcal{S} = \langle g_1, \ldots, g_{n-k} \rangle$. Det logiska kodrummet, C, definieras som det underrum där alla tillstånd stabiliseras av samtliga operatorer i $\mathcal{S}$:

$$C = \{|\psi\rangle : S\,|\psi\rangle = |\psi\rangle\,, \forall S \in \mathcal{S}\}. \tag{2.14}$$

Om gruppen har $n-k$ generatorer, definierar stabilisatorerna ett 2^k-dimensionellt underrum (med k frihetsgrader) där logisk information kan lagras. Varje ny generator fungerar som ett villkor som halverar dimensionen på underrummet. En logisk kvantbit har en frihetsgrad ($k = 1$), vilket alltså kräver $n-1$ generatorer [21, s. 454–459]. Även om en stabilisator är en godtycklig produkt av generatorer, kommer generatorerna, och mätningen av dem, framöver i texten att refereras till som *stabilisatorer*. Detta är standard inom kvantfelskorrektion.

Eftersom stabilisatorerna kommuterar med varandra finns det en gemensam egenbas för hela Hilbert-rummet. Med $n-1$ generatorer kan tillstånden i systemet karakteriseras utifrån deras unika kombination av egenvärden för varje stabilisator. Definiera en vektor $\mathbf{s} = (s_1, s_2, \ldots, s_{n-1})$, där $s_j \in \{-1, 1\}$ svarar mot utfallet av stabilisator S_j. Det totala Hilbert-rummet $\mathcal{H}_{2^n}$ kan då delas in i 2^{n-1} ömsesidigt ortogonala underrum $\mathcal{U}_\mathbf{s}$ som beskrivs av:

$$\mathcal{U}_\mathbf{s} = \{|\psi\rangle : S_j\,|\psi\rangle = s_j\,|\psi\rangle\,, \forall j \in \{1, \ldots, n-1\}\}. \tag{2.15}$$

Eftersom det totala rummet har dimensionen 2^n och antalet sådana underrum är 2^{n-1}, måste varje $\mathcal{U}_\mathbf{s}$ ha exakt dimension 2. För varje unik kombination av egenvärden $\mathbf{s}$ spänns därmed underrummet upp av exakt två ortogonala *simultana egentillstånd*, vilka kan betecknas $|\nu_{\mathbf{s},1}\rangle$ och $|\nu_{\mathbf{s},2}\rangle$ [21, s. 458–459]. Ett godtyckligt tillstånd $|\psi_\mathbf{s}\rangle$ inom ett specifikt underrum $\mathcal{U}_\mathbf{s}$ är därmed en linjärkombination av dessa två simultana egentillstånd ([26] kallar simultana egentillstånd för *quiescent states*). Av definitionen följer att det ursprungliga kodrummet C exakt sammanfaller med underrummet $\mathcal{U}_{(1,1,\ldots,1)}$. Resterande $2^{n-1}-1$ underrum, där minst ett egenvärde $s_j = -1$, benämns som specifika *felrum* $\mathcal{F}_\mathbf{s}$, och unionen av dessa utgör $\mathcal{F} = \mathcal{H}_{2^n} \setminus C$ [7].

För att manipulera information i kodrummet introduceras logiska operatorer, primärt den logiska Pauli-X-operatorn X_L och Pauli-Z-operatorn Z_L. Dessa utgörs av oberoende Pauli-strängar i $\mathcal{P}_n$ som antikommuterar med varandra ($\{X_L, Z_L\} = 0$) och kommuterar med samtliga stabilisatorer i $\mathcal{S}$, utan att själva tillhöra stabilisatorgruppen. Det är genom valet av dessa operatorer som de logiska bastillstånden definieras. Konventionellt definieras den logiska 0:an, $|0\rangle_L$, och den logiska 1:an, $|1\rangle_L$, som de två egentillstånden till Z_L inom kodrummet C:

$$\begin{aligned} Z_L\,|0\rangle_L &= |0\rangle_L\,, \qquad Z_L\,|1\rangle_L = -\,|1\rangle_L \\ X_L\,|0\rangle_L &= |1\rangle_L\,, \qquad X_L\,|1\rangle_L = |0\rangle_L\,. \end{aligned} \tag{2.16}$$

Om ett fel $E \in \mathcal{P}_n$ inträffar projiceras systemet in i ett felrum $\mathcal{U}_\mathbf{s} \neq C$ [7]. Inom den teoretiska litteraturen förutsätts ofta att fysisk rättning utförs, där felets invers ($E^\dagger = E$) appliceras via fysiska kvantgrindar för att aktivt återföra systemet till C [7]. Vid implementering på fysisk hårdvara undviks dock detta om möjligt, eftersom varje extra kvantgrind introducerar ytterligare brus. Istället tillåts tillståndet förbli i det aktuella felrummet, vars basvektorer utgör en förskjuten logisk 0:a ($E\,|0\rangle_L$) och 1:a ($E\,|1\rangle_L$) [15]. Mätutfallen $\mathbf{s}$ antas som systemets nya normaltillstånd, en ”ny baslinje”. Under avkodningen jämförs framtida stabilisatormätningar med denna baslinje, vilket matematiskt motsvarar en bitvis XOR-operation mellan framtida och nuvarande mätvärde, för att endast identifiera *nya* fel som uppstår [16]. Detta beskrivs mer i avsnitt 2.7.

När avkodaren väl har identifierat felet E utifrån stabilisatormätningarna, kontrolleras huruvida E kommuterar eller antikommuterar med systemets logiska operatorer X_L och Z_L. Om felet E till exempel antikommuterar med Z_L innebär det att det agerar som en bit-flip på det logiska tillståndet (en logisk 0:a transformeras till en logisk 1:a, och vice versa). Istället för att rätta

detta fysiskt, uppdaterar mjukvaran sin referensram och inverterar tolkningen av det slutgiltiga mätutfallet för just den logiska kvantbiten [16]. Detta kallas för *Pauli frame tracking*.

2.4.2 Projektiva mätningar

En stabilisatormätning projicerar tillståndet på ett +1 eller -1-egenrum av stabilisatorn. Betrakta en stabilisator S_a och ett kvanttillstånd $|\psi\rangle$ som endast består av en linjärkombination av simultana +1-egentillstånd $|\nu_i\rangle$, som alltså alla uppfyller $S_a |\nu_i\rangle = (+1) |\nu_i\rangle$. Mätutfallet av en hermitesk operator i kvantmekanik blir alltid ett av operatorns egenvärden. En mätning av S_a kommer därmed garanterat att ge utfallet +1. Om $|\psi\rangle$ däremot är en superposition av både +1- och −1-egentillstånd till S_a, är $|\psi\rangle$ inte längre själv ett egentillstånd till S_a, utan innehåller komponenter i två distinkta egenrum. Därmed är mätningen inte längre deterministisk; tillståndet kommer att kollapsa till ett av egenrummen med en sannolikhet som ges av amplitudernas belopp i kvadrat [21, s. 84–85]. Om systemet till exempel består av en datakvantbit i tillståndet $|\psi\rangle = |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, och en mätning av Z (med egentillstånden $|0\rangle$ och $|1\rangle$) genomförs, kommer $|\psi\rangle$ att kollapsa till antingen $|0\rangle$ eller $|1\rangle$ med 50% sannolikhet vardera. Tillståndet omedelbart efter mätningen kan uttryckas med hjälp av projektionsoperatorer [21, s. 458–459]

$$P_a^{\pm} = \frac{1}{2}(\mathbb{I} \pm S_a), \tag{2.17}$$

där tecknet för projektionsoperatorn bestäms probabilistiskt av mätutfallet av S_a. För tillståndet $|\psi\rangle = \frac{1}{\sqrt{2}}(|\nu_1\rangle + |\nu_2\rangle)$, där $|\nu_1\rangle$ respektive $|\nu_2\rangle$ är ett +1- respektive −1-egentillstånd till S_a, blir effekten av en projektion till +1-rummet:

$$\begin{aligned} P_a^{+} |\psi\rangle &= \frac{1}{2\sqrt{2}} \left(\mathbb{I}(|\nu_1\rangle + |\nu_2\rangle) + S_a(|\nu_1\rangle + |\nu_2\rangle)\right) \\ &= \frac{1}{2\sqrt{2}} \left(|\nu_1\rangle + |\nu_2\rangle + |\nu_1\rangle - |\nu_2\rangle\right) = \frac{1}{\sqrt{2}} |\nu_1\rangle \,. \end{aligned} \tag{2.18}$$

Kollapsen filtrerar alltså bort −1-egentillstånden om utfallet blir positivt (och vice versa om P_a^{-} väljs).

2.5 Detektion och korrigering av fel med stabilisatorkoder

I klassisk information är bit-flips som tar $0 \rightarrow 1$ och vice versa den enda typen av fel som behöver beaktas. För en kvantbit, definierad enligt (2.2) eller (2.3), kan sannolikhetsamplituderna anta ett kontinuerligt spektrum av värden, vilket innebär att kvantbiten kan utsättas för ett oändligt antal möjliga fel. Kvantfel kan uppstå på grund av olika fysiska processer. Ett exempel är koherenta rotationer på Bloch-sfären, vilket kan bero på systematiska kontrollfel i hårdvaran. Matematiskt beskrivs sådana koherenta fel av en unitär operator $U(\delta\theta, \delta\phi)$:

$$U(\delta\theta, \delta\phi) |\psi\rangle = \cos\frac{\theta + \delta\theta}{2} |0\rangle + e^{i(\phi+\delta\phi)} \sin\frac{\theta + \delta\theta}{2} |1\rangle \,, \tag{2.19}$$

där $\theta + \delta\theta$ och $\phi + \delta\phi$ är de nya koordinaterna på Bloch-sfären. Genom att variera $\delta\theta$ och $\delta\phi$ kan kvantbiten utsättas för ett kontinuerligt spektrum av fel. Kvantfel kan dock diskretiseras, vilket gör att korrigering för ett ändligt antal fel räcker för korrigering av alla fel [7].

De koherenta felen kan skrivas som en superposition av Pauli-matriser:

$$U(\delta\theta, \delta\phi) |\psi\rangle = \alpha_I I |\psi\rangle + \alpha_X X |\psi\rangle + \alpha_Z Z |\psi\rangle + \alpha_{XZ} XZ |\psi\rangle \,, \tag{2.20}$$

där $\alpha_{I,X,Z,XZ}$ är expansionskoefficienter. Genom projektiva mätningar kollapsar superpositionen till en delmängd av termerna. Detta innebär att en kvantfelkorrigeringskod som kan korrigera fel representerade av X- och Z-Pauli-matriser kan korrigera alla koherenta fel. Processen benämns *diskretisering av kvantfel* och är avgörande för *stabilisatorkoders* funktion [7]. Andra fel som kan förekomma är inkoherenta fel, och precis som koherenta fel kan dessa diskretiseras och beskrivas av Pauli-operatorer [27]. Dessa fel beror bland annat på interaktion med omgivningen.

2.5.1 Enkel repetitionskod

I detta avsnitt behandlas ett enklare exempel av en så kallad *repetitionskod* för att illustrera koncept som har presenterats i tidigare avsnitt. Det enklaste exemplet på en repetitionskod är att använda två kvantbitar (datakvantbitar) för att skapa ett logiskt tillstånd:

$$|\psi\rangle_L = \alpha\,|0\rangle_L + \beta\,|1\rangle_L = \alpha\,|00\rangle + \beta\,|11\rangle\,. \tag{2.21}$$

Enligt beskrivningen i avsnitt 2.4.1 tillhör detta logiska tillstånd ett Hilbertrum $\mathcal{H}_4$, med basen $|00\rangle$, $|10\rangle$, $|01\rangle$ och $|11\rangle$, jämfört med ett system med endast en kvantbit som tillhör $\mathcal{H}_2$. De logiska tillstånden, $|\psi\rangle_L$ begränsas till kodrummet (se ekvation (2.14)), som är ett delrum av $\mathcal{H}_4$ [7]:

$$|\psi\rangle_L \in C = \text{span}\{|00\rangle\,, |11\rangle\} \subset \mathcal{H}_4, \tag{2.22}$$

Underrummet $|\psi\rangle_L \in \mathcal{F}_{(-1)} = \text{span}\{|01\rangle\,, |10\rangle\}$ är det enda felrummet och därmed lika med $\mathcal{F}$.

Om den logiska kvantbiten utsätts för en bit-flip hos en av de fysiska kvantbitarna fås till exempel:

$$X_1\,|\psi\rangle_L = \alpha\,|10\rangle + \beta\,|01\rangle\,, \tag{2.23}$$

och det framgår att den logiska kvantbiten nu tillhör delrummet $\mathcal{F}$. Då C och $\mathcal{F}$ är ortogonala delrum, är det alltså möjligt att urskilja vilket delrum den logiska kvantbiten tillhör genom en projicerande mätning. Sådana mätningar kallas för *stabilisatormätningar* och använder stabilisatorer S [7](se avsnitt 2.4.1). För att inte kollapsa kvanttillståndet vid stabilisatormätningarna introduceras hjälp-kvantbitar kallade *ancillor*. Ancillorna sammanflätas med datakvantbitarna sådant att mätningar av dem effektivt mäter stabilisatorerna samtidigt som det logiska tillståndet förblir oförändrat (se avsnitt 2.6 för en mer detaljerad redogörelse).

Som nämnt i avsnitt 2.4.1 krävs det $n - 1 = 1$ stabilisator S i det här exemplet där $n = 2$, för att definiera en logisk kvantbit. Stabilisatorn i detta fall är $S = Z_1Z_2$. Egenvärdet av en Z_1Z_2-mätning representerar kvantbitarnas gemensamma Z-paritet, där ett +1-utfall indikerar ett jämnt antal ettor och ett -1-utfall indikerar ett udda antal ettor. Eftersom båda bastillstånden i det logiska kodrummet ($|00\rangle$ och $|11\rangle$) redan har en bestämd jämn total Z-paritet, kommer en mätning av stabilisatorn Z_1Z_2 att ge utfallet +1 med 100% säkerhet, och lämnar därmed superpositionen intakt:

$$Z_1Z_2\,(\alpha\,|00\rangle + \beta\,|11\rangle) = (+1)\,|\psi\rangle_L\,. \tag{2.24}$$

Omvänt projicerar Z_1Z_2 operatorn tillstånden $X_1\,|\psi\rangle_L$ och $X_2\,|\psi\rangle_L$ på egenrummet (-1):

$$Z_1Z_2\left(E\,|\psi\rangle_L\right) = (-1)EZ_1Z_2\,|\psi\rangle_L = (-1)\,E\,|\psi\rangle_L\,, \tag{2.25}$$

där E kan vara antingen X_1 eller X_2. Notera att $Z_1Z_2E = (-1)EZ_1Z_2$ eftersom E antikommuterar med Z_1Z_2.

Exemplet ovan med $|0\rangle_L = |00\rangle$ och $|1\rangle_L = |11\rangle$ är en väldigt enkel repetitionskod. Trots att den introducerar redundans är den kraftigt begränsad i sin förmåga att skydda kvantinformation. För att förstå varför måste kodens logiska operatorer X_L och Z_L identifieras.

Ett logiskt fel inträffar när fysiska felkomponenter i systemet kombineras på ett sådant sätt att de bildar en av de logiska operatorerna, se (2.16) för definitioner. Eftersom de logiska operatorerna per definition kommuterar med alla stabilisatorer i $\mathcal{S}$, kommer ett sådant fel att passera obemärkt förbi alla stabilisatormätningar. I den enkla repetitionskoden utgörs stabilisatorn av Z_1Z_2. Operatorn Z_L måste kommutera med denna och ge upphov till ett fasfel. Ett enda Z-fel på den första kvantbiten uppfyller

$$Z_1 \left(\alpha \left|00\right\rangle + \beta \left|11\right\rangle\right) = \alpha \left|00\right\rangle - \beta \left|11\right\rangle . \tag{2.26}$$

Detta tillstånd ligger i C, vilket bekräftas av att stabilisatormätningen Z_1Z_2 fortfarande ger egenvärdet +1. Eftersom den logiska operatorn Z_L består av en Z-operator, behöver endast ett fel inträffa för att det logiska tillståndet ska ändra fas utan att bryta mot några stabilisatorvillkor. Koden saknar därmed förmåga att detektera fasfel.

För X-fel (bit-flips) är situationen något annorlunda. Den logiska bit-flip-operatorn, X_L, som antikommuterar med Z_L och kommuterar med alla stabilisatorer, måste flippa bastillstånden mellan $|00\rangle$ och $|11\rangle$. För att åstadkomma detta krävs fysiska X-fel på båda kvantbitarna samtidigt: $X_L = X_1X_2$. Om endast ett X_1-fel inträffar, kastas systemet ut ur kodrummet, vilket stabilisatormätningen detekterar som egenvärdet -1. Koden kan dock inte korrigera detta fel. Ett X_1-fel och ett X_2-fel resulterar båda i samma mätutfall, vilket gör det omöjligt för avkodaren att avgöra vilken av de två kvantbitarna som växlats och därmed behöver rättas.

För att kvantifiera en stabilisatorkods förmåga att hantera fel används begreppet kodavstånd, d. Kodavstånd definieras som det minsta antalet fysiska fel (eller den kortaste Pauli-strängen) som krävs för att bilda en logisk operator (X_L, Z_L eller Y_L) och därmed orsaka ett oavsiktligt logiskt fel. Det minsta antalet fel en kod teoretiskt kan detektera är $d-1$, medan den kan korrigera upp till $\lfloor \frac{d-1}{2} \rfloor$ fel [7].

För koden i vårt exempel krävs det två fysiska bit-flips för att skapa ett logiskt X_L-fel, men det krävs endast *ett* fysiskt fasfel för att skapa ett logiskt Z_L-fel. Eftersom kodavståndet definieras av den mest sårbara logiska operationen är kodens totala distans därmed $d = 1$. En kod med $d = 1$ saknar helt förmåga att garantera felkorrigering, vilket belyser nödvändigheten av att konstruera mer avancerade stabilisatorkoder för robusta logiska kvanttillstånd.

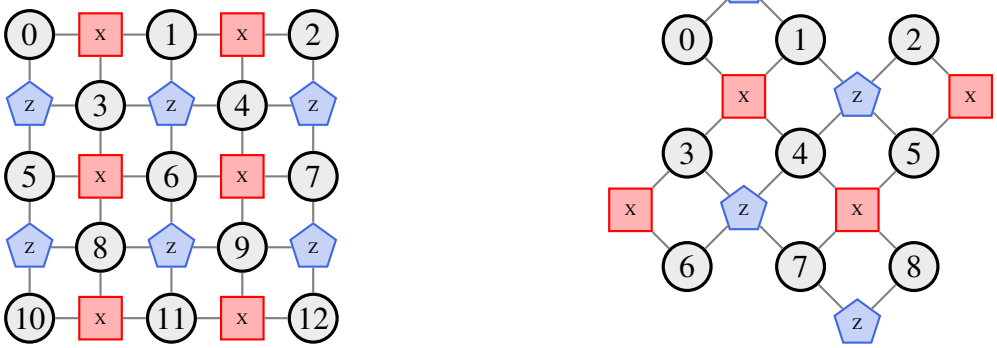
2.5.2 Ytkoder

En ytkod (eng. surface code) är ett exempel på en arkitektur för stabilisatorkoder för att skapa en feltolerant logisk kvantbit [7][15]. Den gör detta genom att, precis som i ekvation (2.21), lagra kvantinformation i flera kvantbitar. I figur 2.2b visualiseras en 3×3 roterad ytkod där den logiska informationen lagras i 9 sammanflätade datakvantbitar. Ytkoderna i figuren 2.2 har kodavståndet $d = 3$, och kan alltså korrigera både bit-fel och fas-fel [7]. Exempelvis har en 7×7 ytkod kodavståndet $d = 7$. Allmänt är kodavståndet för en ytkod lika med den minsta sidlängden. Anledningen att den kallas roterad är att gittret för kvantbitarna är roterat 45° jämfört med gittret för icke-roterade ytkoder [15], se figur 2.2.

De färgade noderna i figur 2.2 representerar de fysiska ancilla-kvantbitar som används som stabilisatorer. Innan stabilisatorerna mäts, sammanflätas respektive ancilla med sina grannar, och mätningen av ancillorna blir då ekvivalent med att mäta stabilisatorerna. Om mätningen av en stabilisator ger egenvärde +1, kommer den korresponderande ancillan att mätas till $|0\rangle$, medan egenvärdet -1 gör att ancillan mäts till $|1\rangle$, förutsatt att ancillan nollställs till $|0\rangle$ efter varje mätning.

Ytkoden i figur 2.2b innehåller åtta stabilisatorer $\mathcal{S} = \{X_{0134}, X_{4578}, X_{36}, X_{25}, Z_{1245}, Z_{3467}, Z_{01}, Z_{78}\}$ där till exempel $X_{0134} = X_0X_1X_3X_4$. Fyra av dessa är vikt-2 och fyra är vikt-4, där vikten mot-

svarar antalet X eller Z operationer i stabilisatorn (eller antalet tal i subskriptet). För ytkoden i figur 2.2b kan de logiska operatorerna Z_L och X_L exempelvis väljas som $Z_L = Z_0 Z_3 Z_6$ och $X_L = X_0 X_1 X_2$ (Z- respektive X-operatorer på en godtycklig vertikal respektive horisontell rad kan väljas). Notera att både Z_L och X_L består av tre datakvantbitsoperatorer och därav kodavståndet $d = 3$.



(a) Oroterad ytkod med $d = 3$.

(b) Roterad ytkod med $d = 3$.

Figur 2.2: Visualisering av ytkoder på ett kvantchipp med kvadratisk topologi. Den 45°-roterade koden använder 9 data- och 8 ancillabitar, till skillnad från 13 data- och 12 ancillabitar hos den oroterade. Datakvantbitarna utgör det sammanflätade kodade tillståndet, medan ancillabitarnas tillstånd är tillfälliga. Ancillabitarna sammanflätas och mäts för att mäta stabilisatorerna, och kollapsas därför till Z-egentillstånd.

Notera att varje X-stabilisator S_x överlappar med en Z-stabilisator S_z på exakt två datakvantbitar i figur 2.2b, vilket säkerställer kommuteringsförhållandet för stabilisatorerna.

En ytkod, oavsett kodavstånd d, innehåller alltid en mindre stabilisator än antalet datakvantbitar. Anledningen till detta, som nämndes kort i avsnitt 2.4.1, är att varje stabilisatormätning effektivt halverar antalet möjliga simultana egentillstånd som superpositionen efter stabilisatormätningarna kan bestå av. För fallet med en ytkod med kodavstånd $d = 3$ är antalet möjliga simultana egentillstånd 2^9 i början. Efter de 8 stabilistatormätningarna, som beskrivs med vektorn $\mathbf{s}$ (se avsnitt 2.4.1), består därmed kvanttillståndet av maximalt $2^9 \cdot 2^{-8} = 2$ simultana egentillstånd $|\nu_{\mathbf{s},1}\rangle$ och $|\nu_{\mathbf{s},2}\rangle$. Om stabilisatormätningarna exempelvis blir endast nollor, kommer dessa två egentillstånd att uppfylla $S\,|\nu_{\mathbf{s},i}\rangle = |\nu_{\mathbf{s},i}\rangle$ för alla stabilisatorer $S \in \mathcal{S}$, vilket är definitionen av kodrummet.

För att konstruera $|0\rangle_L$ initialiseras först alla datakvantbitar i $|0\rangle$ så att $|\psi\rangle = |0\rangle^{\otimes 9}$. Därefter mäts samtliga stabilisatorer. Tillståndet är initialt i ett +1-egentillstånd till alla Z-stabilisatorer, vilket innebär att mätresultatet för dessa (i ett brusfritt system) deterministiskt blir 0. Däremot är inte $|\psi\rangle$ ett egentillstånd till X-stabilisatorerna. Mätningen av dessa kommer successivt att projicera tillståndet ner, se (2.4.2), till ett simultant egentillstånd som definieras som det nya $|0\rangle_L$ [7]:

$$|0\rangle_L = P^{\pm}_{X_{0134}} P^{\pm}_{X_{4578}} P^{\pm}_{X_{12}} P^{\pm}_{X_{67}} |0\rangle^{\otimes 9} = \left(\prod_{i=1}^{4} \frac{1}{2} \left(\mathbb{I} \pm S_i \right) \right) |0\rangle^{\otimes 9}, \tag{2.27}$$

där S_i betecknar en av de fyra X-stabilisatorerna. I beräkningsbasen består detta tillstånd av 16 basvektorer, då $\left(\prod_{i=1}^{4} \frac{1}{2} \left(\mathbb{I} \pm S_i \right) \right) |0\rangle^{\otimes 9}$ motsvarar en summa av 2^4 distinkta operationer på

$|0\rangle^{\otimes 9}$. Eftersom mätresultaten för S_i är slumpmässiga (ancillan mäts till 0 eller 1), hamnar systemet i ett av underrummen $\mathcal{U}_{\mathbf{s}}$, se avsnitt 2.4.1 och ekvation (2.15) för definition av $\mathcal{U}_{\mathbf{s}}$. Tillståndet projiceras med hög sannolikhet till ett av felrummen $\mathcal{U}_{\mathbf{s}} = \mathcal{F}_{\mathbf{s}}$ i samband med första stabilisatorrundan, men det är inte nödvändigtvis ett tecken på att ett fel har uppstått, eftersom kollapsen är slumpmässig även utan brus. Som nämndes i avsnitt 2.4.1 betraktas detta underrum som systemets nya logiska kodrum C, och tillståndet (2.27) tolkas därför som en logisk nolla.

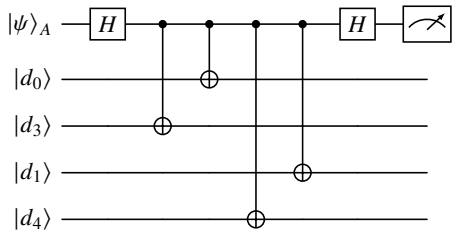
2.6 Förberedelse för stabilisatormätningar

För att mätning av en ancilla ska vara ekvivalent med en stabilisatormätning måste ancillan (i antingen $|0\rangle_A$ eller $|1\rangle_A$) sammanflätas med datakvantbitarna via CNOT. Sammanflätningen för Z-stabilisatorer och X-stabilisatorer görs på olika sätt. För Z-stabilisatorerna följs kretsschemat i figur 2.3b, som beskriver fallet med en av vikt-4 stabilisatorerna från figur 2.2b. Datakvantbitarna agerar som kontroll och ancillan som target i respektive CNOT-grind. Om ancillan var i tillstånd $|i\rangle_A$, med $i = 0, 1$, kommer tillståndet efter de fyra CNOT-grindarna att vara [7]:

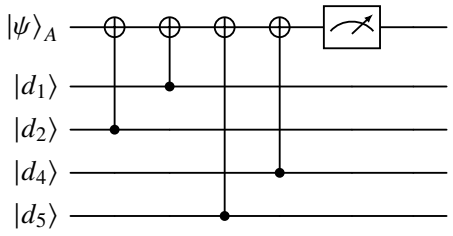
$$\begin{aligned} |\psi_{\text{efter}}\rangle &= P^+ |\psi\rangle |i\rangle_A + P^- |\psi\rangle |i_A \oplus 1\rangle \\ &= \frac{1}{2}\left(\mathbb{I} + Z_1Z_2Z_4Z_5\right)|\psi\rangle |i\rangle_A + \frac{1}{2}\left(\mathbb{I} - Z_1Z_2Z_4Z_5\right)|\psi\rangle |i_A \oplus 1\rangle , \end{aligned} \tag{2.28}$$

där $\oplus$ är XOR. Givet ett udda antal X-fel på datakvantbitar 1, 2, 3 och 4 kommer ancillan byta från $|0\rangle_A$ till $|1\rangle_A$ (eller vice versa). Proceduren att förbereda en X-stabilisator framgår av figur 2.3a, som beskriver fallet med en av vikt-4 stabilisatorerna från figur 2.2b. Skillnaden är att ancillan agerar som kontroll och datakvantbitarna som target, samt att en Hadamard-grind appliceras på ancillan innan och efter CNOT-grindarna. Tillståndet efteråt blir som i ekvation (2.28), med skillnaden att X-stabilisatorn i fråga används istället [26].

Ordningen som CNOT-grindarna appliceras bör vara annorlunda för Z-stabilisatorer och X-stabilisatorer. För fallet med en Z-stabilisator bör ancillan sammanflätas med datakvantbitarna i ordningen: NE (north east), NW (north west), SW (south west), SE (south east), om ytkoden har stabilisatorer enligt figur 2.2b, och denna ordningen följs i figur 2.3b. För X-stabilisatorerna ska ordningen istället vara: NE, SE, NW, SW. Genom att följa den ordningen säkerställs det att en CNOT-grind aldrig appliceras på en datakvantbit samtidigt från två olika ancillor. Ordningen säkerställer också att hook-fel minimeras [28], vilket kortfattat beskrivs i avsnitt 2.7.1. Att utföra en mätning av varje stabilisator kallas för en *stabilisatorrunda*. I regel behövs d stabilisatorrundor för en ytkod, där d är kodavståndet. Detta för att avkodningsalgoritmen ska kunna ta hänsyn till mätfel, och Pauli-fel på ancillor. Detta beskrivs mer i detalj i avsnitt 2.7.1 och 2.7.2.



(a) X-stabilisator, alltså mätning av $X_0X_3X_1X_4$



(b) Z-stabilisator, alltså mätning av $Z_1Z_2Z_4Z_5$

Figur 2.3: Kvantkretsar som implementerar X- och Z-stabilisatorer i ytkoden enligt figur 2.2b. Tillståndet $|\psi\rangle_A$ tillhör en ancillakvantbit medan de andra tillhör datakvantbitar. Ancillatillståndet börjar som rent $|0\rangle$ eller $|1\rangle$, och efter sammanflätning via CNOT-grindarna mäts det till XOR av initialtillståndet med mätresultatet av stabilisatorn. Kretsen här är förenklad, då en parallell variant används i praktiken där alla stabilisatorer kan mätas samtidigt [28].

2.7 Avkodning

På ytkoder kan Pauli-fel uppstå på både datakvantbitar och ancillor. Ett X-fel på en datakvantbit inverterar pariteten hos angränsande Z-stabilisatorer (och ett Z-fel ändrar pariteten på X-stabilisatorer). Mätningarna av de påverkade stabilisatorerna kommer då att ge 1 (eller 0) om de i förra stabilisatorrundan mättes till 0 (eller 1). De flesta datakvantbitar i en ytkod (speciellt en stor ytkod), angränsar till två stabilisatorer av samma typ (X eller Z).

I litteraturen är det vanligt att ancillorna nollställs till $|0\rangle$ efter varje mätning, men i det här arbetet var det nödvändigt att implementera *virtuella nollställningar*. Kvantprocessorn *IBM Nighthawk r1* stödjer inte dynamiska grindar [29], och möjligen av det skälet tillät kompilatorn inte nollställningar. Istället utnyttjas det att ancillabitarna lika gärna initialt kan vara 1, så länge det bokförs, och av det skälet införs konceptet *detektorer*.

En detektor definieras som en specifik uppsättning mätningar vars XOR-summa alltid är deterministiskt lika med 0 i en perfekt, brusfri kvantdator [16]. Om en detektor utvärderas till 1 (ett utslag) är det därmed ett bevis på att ett eller flera fel har inträffat bland de operationer som detektorn övervakar. För varje stabilisatorrunda $t = 0, 1, \ldots$, definieras en ny detektor för alla ancillor, vilket ger en uppsättning över både rum och tid.

Detektorernas konstruktion beror på kretsens arkitektur. I en standardytkod med kontinuerlig nollställning av ancillorna mellan stabilisatorrundor, förväntas mätvärdet för en stabilisator vara identiskt mellan två på varandra följande rundor om inget nytt fel uppstått, varvid detektorn för en ancilla definieras som $D(t) = M(t) \oplus M(t-1)$ för mätresultaten $M(t)$. Med virtuell nollställning jämförs istället mätningen med den förrförra mätningen av exakt samma ancilla för att pariteten ska förbli deterministisk, ekvivalent att beräkna XOR med förra mätningen upprepat två gånger. Den primära detektorn i ytkoden uttrycks därmed som $D(t) = M(t) \oplus M(t-2)$, om $t \geq 2$.

2.7.1 Matchningsgraf

För att avkodningsalgoritmen ska kunna bearbeta utfallen, måste det fysiska systemet först översättas till en matchningsgraf [30]. Det är viktigt att betona att grafen konstrueras helt oberoende av vilka detektorer som faktiskt larmar under körningen; grafen är snarare en teoretisk karta över

kvantdatorns sårbarheter, skapad innan kvantdatorn kör. I denna graf utgör detektorerna noder. Kanterna som förbinder noderna representerar specifika händelser som motsvarar fel i kretsen, till exempel ett Pauli-fel på en datakvantbit, ett Pauli-fel på en ancilla, eller att en mätning ger en felaktig avläsning [30] [16]. En kant dras mellan två noder om en sådan enskild felhändelse förväntas trigga just dessa två detektorer. Vikten (w) på kanten definieras logaritmiskt som $w = -\ln\left(\frac{p}{1-p}\right)$, där p uteslutande baseras på sannolikheten att det specifika felet uppstår i hårdvaran. Kanten representerar alltså inte den allmänna sannolikheten att de två detektorerna larmar av godtyckliga skäl, utan är en direkt avspegling av en specifik felkälla. Kanterna i grafen kan kategoriseras utifrån hur felet propagerar i rummet och tiden:

Temporala kanter (tid): En temporal kant förbinder detektorer som tillhör samma ancilla, men i olika tidsrundor [30]. Dessa representerar fel som uppstår i tidsdomänen, och det finns främst två sådana typer av fel:

- **Mätfel**. Om mätinstrumentet ger ett felaktigt utslag i runda t, kommer detta felaktiga värde att ge utslag på både detektor $D(t)$ och $D(t+2)$. Detta mätfel skapar därmed en temporal kant direkt mellan just dessa två noder, vägd mot utläsningsfelets sannolikhet ($p_{\text{mät}}$). Skulle larm istället uppstå i runda t och $t+4$, saknas en direkt kant; avkodaren måste då kombinera två kanter för att förklara larmen, vilket korrekt leder algoritmen till slutsatsen att två oberoende mätfel har inträffat. En sådan slutsats görs av avkodningsalgoritmen i samband med att syndromgrafen skapas och att MWPM körs, som beskrivs i avsnitt 2.7.2.
- **Ancilla-fel**. Om en ancilla drabbas av ett fysiskt Pauli-fel kan det skapas en annan typ av tidsförskjutning i larmen. I en arkitektur utan aktiv nollställning av ancillor modelleras detta med en kant mellan temporalt närliggande rundor ($D(t)$ och $D(t+1)$), vägd mot sannolikheten för fel på ancillan (p_{anc}).

Spatiala kanter (rum): En spatial kant förbinder detektorer från olika ancillor av samma typ (exempelvis två X-ancillor), men inom samma stabilisatorrunda (t). Dessa kanter uppstår primärt på grund av lokala Pauli-fel på datakvantbitar. Om ett Pauli-fel uppstår på en datakvantbit som inte ligger på ytkodens kant, kommer garanterat två närliggande detektorer att ge utslag. Ett Z-fel på en sådan gemensam datakvantbit inverterar därmed pariteten hos båda X-ancillorna samtidigt, vilket resulterar i att de två motsvarande detektorerna i runda t ger utslag. För att kartlägga denna sårbarhet dras en spatial kant direkt mellan dessa två noder i grafen. Kantens vikt sätts utifrån sannolikheten att just den inblandade datakvantbiten drabbades av ett fel (p_{data}) under stabilisatorrundan. Vissa datakvantbitar på ytkodens kant kan utsättas för ett Pauli-fel så att endast en detektor larmar. För att avkodningsalgoritmen ska kunna ta hänsyn till denna sårbarhet måste en *rand-kant* dras mellan detektorn och en virtuell nod.

Hook-fel uppstår i samband med sammanflätningen mellan ancilla och närliggande datakvantbitar, som beskrivs i figur 2.3a och 2.3b. Ett Z-fel som uppstår på en X-ancilla efter den andra CNOT-operationen men innan den tredje i figur 2.3a, kommer ge utslag på två detektorer som skiljer sig både i tid och ancilla-position. En kant mellan sådana detektorer bör också dras för att kunna upptäcka hook-fel [28].

2.7.2 Syndromgrafen

När matchningsgrafen är konstruerad och kvantdatorn har körts, inleds den faktiska avkodningen. Detta sker i två huvudsakliga steg där matchningsgrafen reduceras och analyseras för att hitta det mest sannolika felet.

Steg 1: Konstruktion av syndromgrafen via Dijkstras algoritm

Under körningen kommer endast en liten delmängd av alla detektorer att ge utslag (larma). För avkodningsalgoritmen är det enbart dessa noder som måste paras ihop. För att göra detta tillämpas en kortaste-vägen-algoritm, vanligtvis Dijkstras algoritm, över den underliggande matchningsgrafen. Dijkstras algoritm beräknar den väg (kombination av fel) som har den lägsta totala vikten mellan alla par av larmande detektorer (samt till rand-noderna). [30] [16].

Resultatet av denna sökning används för att bygga en ny graf med färre noder: syndromgrafen. I syndromgrafen utgörs noderna enbart av de detektorer som larmade, och varje kant mellan dem representerar nu den minsta totala vikten (den mest sannolika felkedjan) som förbinder dem genom den ursprungliga matchningsgrafen.

Steg 2: Minimum Weight Perfect Matching (MWPM)

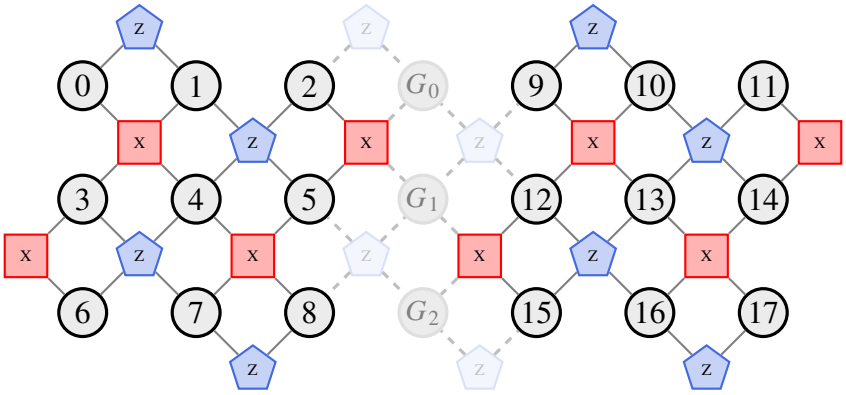
Den färdiga syndromgrafen är en komplett graf där alla larm är sammankopplade med varandra. Ytkodens utformning dikterar att fysiska fel uppstår i ändarna av felkedjor, vilket innebär att larm alltid skapas i par (eller i par med en rand-nod). Problemet reduceras därmed till att para ihop alla larmande noder så att den totala vikten av alla parningar minimeras.

Detta är ett klassiskt graf-teoretiskt problem som löses med en Minimum Weight Perfect Matching-algoritm (historiskt Edmonds Blossom-algoritm). Genom att algoritmen hittar den perfekta matchningen som har lägst sammanlagd vikt maximerar den, till följd av den logaritmiska vikt-definitionen, sannolikheten för det sammantagna händelseförloppet. Den resulterande parningen översätts slutligen tillbaka till de underliggande felmekanismerna i matchningsgrafen, och avkodaren ger tillbaka vilka logiska observabler som mest sannolikt har blivit flippade [16].

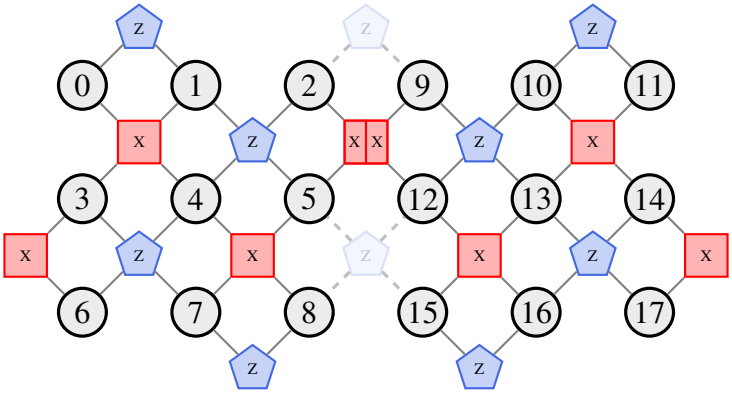
2.8 Lattice surgery

Den logiska CNOT-operationen kan implementeras direkt mellan varje par av datakvantbitar hos två ytkoder, vilket kallas *transversell CNOT*, men på supraledande kvantdatorer kan endast närliggande kvantbitar interagera fysiskt. Därför implementeras logisk CNOT istället med lattice surgery. Lattice surgery är en teknik inom kvantberäkning med två ytkoder där logiska kvanttillstånd manipuleras genom att mäta olika stabilisatorer vid olika stabilisatorrundor, i syfte att utföra logiska operationer [15].

Lattice surgery består av två moment: *merge* och *split*. Det finns olika metoder, där ytkodernas relativa position varierar, för att implementera lattice surgery, se figur 2.4 och 2.5 för två exempel. Dessa metoder har valt att kallas den ”konventionella formen”, som beskrivs i [15], och den ”kompakta formen”, som beskrivs i [31]. Den konventionella formen kräver fler datakvantbitar, men alla stabilisatorer kan mätas samtidigt, medan den kompakta formen inte kräver extra datakvantbitar, utan måste istället dela på vissa fysiska kvantbitar och därmed dela upp vissa stabilisatormätningar i tid. För båda formerna kallas den vänstra ytkoden för A och den högra för B.



Figur 2.4: Konventionell lattice surgery mellan två ytkoder till vänster och höger, för en ZZ-kant. Metoden bygger på att de nya datakvantbitarna $G_0, ..., G_2$ initialiseras till $|+\rangle$, varefter de svagt färgade Z-stabilisatorerna i mitten läggs till i uppsättningen stabilisatorer som mäts efter merge. Produkten av de nya Z-stabilisatorerna är samma som logiska $Z_L^A Z_L^B$, varför en ZZ-merge följt av ZZ-split är ekvivalent med en ZZ-mätning.



Figur 2.5: Kompakt lattice surgery mellan två ytkoder, för ZZ-merge och/eller ZZ-split. Metoden liknar den för konventionell lattice surgery enligt figur 2.4, med skillnaden att det inte finns mellanliggande datakvantbitar, och istället att ytkoderna behöver dela på samma fysiska X-ancillabit i mitten. Produkten av de två svagt markerade Z-stabilisatorerna motsvarar på samma sätt $Z_L^A Z_L^B$.

2.8.1 Merging och splittning

Vardera ytkod i figur 2.4 och 2.5, som beskriver den konventionella respektive kompakta formen, kodar en logisk kvantbit och stabiliseras separat (med minst d stabilisatorrundor), vilket innebär att alla stabilisatorer i figur 2.4 och 2.5 förutom de svagt markerade, mäts innan merge. En så kallad ZZ-merge av två ytkoder, för den konventionella formen, sker med data- och ancilla-kvantbitar placerade enligt figur 2.4. En XX-merge beskrivs på samma sätt, endast att X-ancillorna byter plats med Z-ancillorna, eller om ytkoderna placeras vertikalt i förhållande till varandra, vilket är relevant för CNOT, se avsnitt 2.8.2.

I samband med ZZ-merge för den konventionella formen förbereds de mellanliggande datakvantbitarna i tillståndet $|+\rangle$, medan de i XX-merge förbereds i $|0\rangle$. Sedan påbörjas en ny stabilisatorrunda där alla stabilisatorer (även de svagt färgade Z-stabilisatorerna) mäts. De två vikt-2 X-stabilisatorerna som ligger på A:s högra kant och B:s vänstra kant, görs om till vikt-4

genom att även inkludera de mellanliggande datakvantbitarna. Detta är nödvändigt för att stabilisatorerna för den sammanfogade ytan ska kommutera [15]. Alla stabilisatorer innan och efter merge för den konventionella formen finns beskrivna i tabell A.2.

Produkten av de nya Z-stabilisatorerna visar sig bli $Z_L^{\mathrm{A}} Z_L^{\mathrm{B}}$. Mätningen av stabilisatorerna är därmed också en mätning av $Z_L^{\mathrm{A}} Z_L^{\mathrm{B}}$. Eftersom operatorn $Z_L^{\mathrm{A}} Z_L^{\mathrm{B}}$ är lika med en produkt av stabilisator kommer det sammanfogade tillståndet att förbli i ett egentillstånd till $Z_L^{\mathrm{A}} Z_L^{\mathrm{B}}$ (om inga fel inträffar); $Z_L^{\mathrm{A}} Z_L^{\mathrm{B}} |\psi\rangle = (-1)^{M_{ZZ}} |\psi\rangle$, där M_{ZZ} är 0 om $|\psi\rangle$ kollapsar till ett +1-egentillstånd (kodrummet) vid mätning av $Z_L^{\mathrm{A}} Z_L^{\mathrm{B}}$, och $M_{ZZ} = 1$ annars (ett underrum som inte är kodrummet).

För att kunna utföra logiska operationer på båda ytkoderna oberoende av varandra krävs det att den sammanfogade ytkoden delas till två oberoende ytkoder. Detta åstadkoms genom en operation som kallas för splittning. Liksom vid sammanfogning finns två varianter: XX-split och ZZ-split). Splittning för den konventionella formen genomförs genom att mäta G_1, G_2 och G_3 i Hadamard-basen för ZZ-split eller beräkningsbasen för XX-split, och återgå till att mäta stabilisatorerna för de individuella ytkoderna ($S_1 \cup S_2$ i tabell A.2 för ytkoderna i figur 2.4).

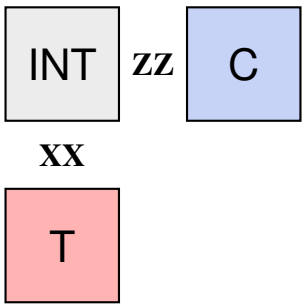
En ZZ-merge följt av ZZ-split på ett allmänt tillstånd med två logiska kvantbitar har följande utfall [31]:

$$\begin{aligned} a|00\rangle_L + b|01\rangle_L + c|10\rangle_L + d|11\rangle_L &\xrightarrow[\text{kollaps}]{M_{ZZ}=0} a|00\rangle_L + d|11\rangle_L\,, \\ a|00\rangle_L + b|01\rangle_L + c|10\rangle_L + d|11\rangle_L &\xrightarrow[\text{kollaps}]{M_{ZZ}=1} b|01\rangle_L + c|10\rangle_L\,, \end{aligned} \tag{2.29}$$

där exempelvis $|00\rangle_L$ betyder att både ytkod A och B är i logiskt tillstånd $|0\rangle_L$. Om $M_{ZZ} = 0$ kommer alltså endast de termer i superpositionen som innehåller jämnt antal logiska logiska ettor, bevaras och på liknande sätt bevaras superpositionerna med udda antal logiska ettor då $M_{ZZ} = 1$.

I samband med ZZ-merge för den kompakta formen blir de nedtonade stabilisatorerna i figur 2.5 stabilisatorer, men de två vikt-2 X-stabilisatorerna på den gemensamma gränsen slås samman till $X_2X_9X_5X_{12}$, vilket säkerställer att alla stabilisatorer för den sammanfogade ytan kommuterar. Stabilisatorerna innan och efter ZZ-merge för den kompakta formen finns i tabell A.1. Produkten av de nya Z-stabilisatorerna är $Z_L^{\mathrm{A}} Z_L^{\mathrm{B}}$, precis som för den konventionella formen [31]. Tillståndet efter ZZ-merge och ZZ-split för den kompakta formen följer också ekvation (2.29).

2.8.2 Logisk CNOT



Figur 2.6: Abstrakt representation av en logisk CNOT där varje kvadrat motsvarar en komplett ytkod av godtycklig storlek. Ytkoderna C och INT är orienterade för att kunna genomgå en ZZ-merge och split, medan ytkoderna T och INT kan genomgå en XX-merge och split. Lattice surgery mellan ytkoderna kan både vara av den konventionella och den kompakta varianten.

Felkorrigerad logisk CNOT med lattice surgery av den kompakta formen utförs genom att göra följande:

1. INT förbereds till $|+\rangle_L$.
2. ZZ-merge följt av ZZ-split mellan yta C och INT utförs och värdet på M_{ZZ} noteras.
3. XX-merge följt av XX-split mellan yta INT och T utförs och värdet på M_{XX} noteras.
4. Logiska Z_L^{INT} mäts med mätufallet M_Z, 0 eller 1.
5. Avkodning utförs och de felkorrigerade värdena för $M_{ZZ} = a$, $M_{XX} = b$ och $M_Z = c$ erhålls.
6. Resultaten tolkas som att Z^b har applicerats $|\text{C}\rangle$ och X^{a+c} på $|\text{T}\rangle$ [31].

Såvida avkodaren inte misslyckas på grund av ett logiskt fel, agerar C och T som kontroll-bit respektive target-bit. För logisk CNOT med lattice surgery av den konventionella formen behövs även ett mätvärde från XX-split (mätvärdet av den mellanliggande datakvantbiten längst till höger vid gränsen mellan INT och T i figur 2.6) användas för att avgöra om ytterligare en justerande X-operation behöver appliceras på $|\text{T}\rangle$.

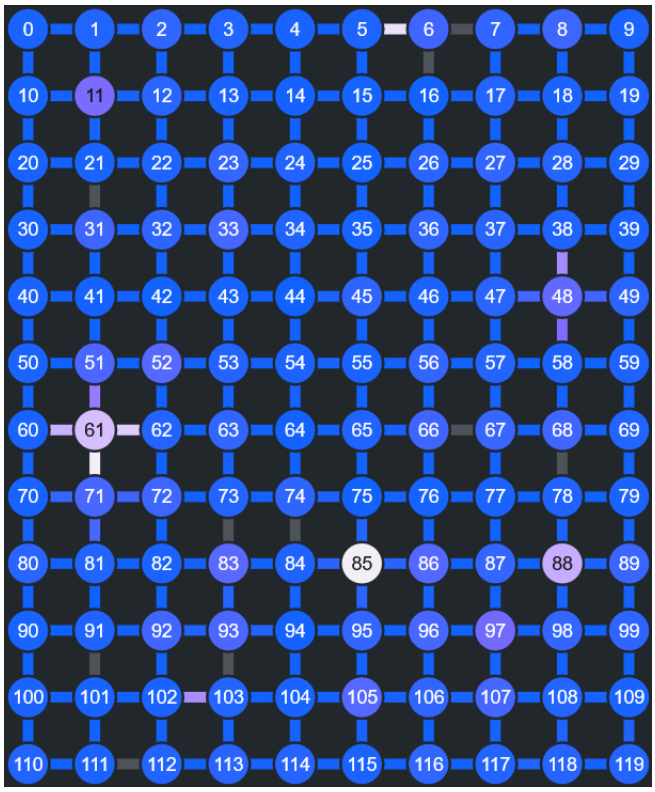
2.9 Kvantprocessor

En kvantprocessor (eng. quantum processing unit, hädanefter QPU), är en hårdvaruenhet som möjliggör beräkningar med en mängd kvantbitar. En QPU kan vara uppbyggd på olika sätt, bland annat med supraledande kvantbitar (eng. superconducting qubits) vilket tillämpas bland annat hos IBM i lager av halvledare [32].

Processorn *Nighthawk r1* är en av IBM:s senaste QPU:er, lanserad i november 2025. Processorn har en kvadratisk topologi med 120 kvantbitar, där varje kvantbit kan interagera med sina fyra närmaste grannar utan SWAP-grindar [18], se figur 2.7. Den fysiska instansen av Nighthawk r1 som användes har namnet *IBM Miami*.

Kvantdatorn styrs genom att programmatiskt ge chippet instruktioner på vilka grindar som ska utföras när och på vilka kvantbitar. Kvantdatorn har inbyggt stöd för vissa *basgrindar* som alla andra grindar måste transpileras till, och även icke-unitära grindar såsom mätningar av kvantbitar. Basgrindarna är bland annat RZ, CZ, $\sqrt{X}$ [33]. När kvantkretsarna i det här arbetet kördes fysiskt, transpilerades exempelvis Hadamardgrinden till en kombination av RZ och $\sqrt{X}$,

och CNOT till CZ med Hadamardgrindar för target-kvantbiten. Kretsen före transpilering bestod endast av Hadamard, CNOT och mätningar.



Figur 2.7: En kartvy av kvantprocessorn *IBM Nighthawk r1* på instansen *IBM Miami* under 2026-05-05 klockan 08:40. Vyn visar varje fysisk kvantbit som en cirkel med ett unikt index, och med anslutningar mellan närliggande kvantbitar där CZ-grindar kan genomföras fysiskt. Kvantbitarna är färgkodade efter sannolikheten för mätfel och anslutningarna efter CZ-fel, där blått är lägst och vitt högst. De gråa anslutningarna är defekta, så en ytkod behöver placeras där den inte skär en sådan anslutning, eftersom ytkoderna placerades 1:1 fysiskt och inte förflyttades under körningen.

3
Metod

För att implementera kvantkretsarna för ytkoder och lattice surgery, har ramverket *Qiskit* använts, både för simulering och exekvering på den fysiska kvantprocessorn *IBM Nighthawk r1*. För simuleringarna användes en brusmodell som konfigurerades för att simulera fel för enkvantsoperationer såsom Hadamard-grindar (H) och tvåkvantbitsoperationer som CNOT. Utöver grindfel simulerades även mätfel. Sannolikheten för fel kunde justeras. I detta avsnitt beskrivs hur simuleringarna och exekveringen på kvantprocessorn har implementerats och testats.

3.1 Ytkoder

Kod för kvantkretsen av en ytkod skrevs, se Appendix B. För att kunna testa koden delades kvantkretsen in i tre moment.

Först initialiserades alla fysiska kvantbitar på ytkoden till ett bestämt tillstånd, till exempel $|1\rangle^{\otimes 9}$. Detta gjordes genom att återställa alla datakvantbitar, vilket innebär att de sattes till tillståndet $|0\rangle$. Därefter applicerades eventuella X-, Z- och Hadamard-grindar för att transformera alla datakvantbitar till ett och samma tillstånd av: $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$.

Stabilisatorrundor implementerades genom att sammanfläta data- och ancillakvantbitar med CNOT-grindar och mäta ancillorna, se avsnitt 2.6. Efter första stabilisatorrundan kollapsade initialtillståndet till ett bestämt logiskt tillstånd (exempelvis $|1\rangle_L$ om det initiala tillståndet var $|1\rangle^{\otimes 9}$). Stabilisatormätningarna för Z-ancillor respektive X-ancillor sparades i olika register. Alla kvantbitar (inklusive datakvantbitarna) mättes slutligen i antingen Z- eller X-basen för att kunna verifiera kretsen och felkorrigeringsförmågan.

Utifrån mätningarna i det sista momentet beräknades mätutfallen av Z_L eller X_L genom att multiplicera ihop mätvärdena av korresponderande datakvantbitar. För att verifiera kretsen utfördes ett antal tester. Med brusfria simuleringar ($p_{\text{mät}} = p_{\text{grind}} = 0$) kontrollerades att mätutfallet av Z_L eller X_L överensstämde med det initiala tillståndet. Med injicerade fel på datakvantbitar kontrollerades att avkodningen korrekt kunde korrigera mätutfallet av Z_L eller X_L. Med brusiga simuleringar uppskattades hur den logiska felsannolikheten berodde på mätfel, grindfel och antalet stabilisatorrundor. Det testades även om avkodningen signifikant kunde minska den logiska felsannolikheten.

3.2 Lattice surgery

En krets för lattice surgery med den konventionella formen respektive den kompakta formen (se avsnitt 2.8 för definitioner) skapades med koden i Appendix B. Den kompakta formen användes på kvantprocessorn, medan den konventionella formen användes i de kvantitativa simuleringarna. Koden tog hänsyn till de faser som en lattice surgery-operation består av och ytterligare moment för att kunna testa koden.

- Två separata ytkoder initialiserades såsom beskrivits i föregående sektion.
- Stabilisatorrundor genomfördes på respektive ytkod följt av en merge- och en split-operation som utfördes såsom beskrivet i 2.8.1. Värdet för M_{ZZ} eller M_{XX} vid merge-operationen beräknades och sparades. Mätutfallen av mellanliggande kvantbitarna vid split-operationen sparades också (vid fallet av den konventionella formen).
- Alla fysiska kvantbitar mättes i antingen Z- eller X-basen beroende på merge- och split-typen.

För varje moment utfördes minst tre stabilisatorrundor. Mätningarna av Z_L respektive X_L beräknades som den totala pariteten från mätutfallen av kvantbitarna närmast merge-gränsen (till exempel kvantbitarna 2, 5, 8 för den vänstra och 3, 6, 9 för den högra ytkoden i figur 2.4). Den totala pariteten beräknades utifrån de logiska mätningarna, $Z_L^A Z_L^B$ respektive $X_L^A X_L^B$, där A och B betecknar de två ytkoderna,

För att verifiera att koden var korrekt genomfördes ett antal tester med simuleringar. Simuleringar med och utan brus kördes, där värdena på $Z_L^A Z_L^B$ respektive $X_L^A X_L^B$ jämfördes med M_{ZZ} respektive M_{XX}. Mätutfallen av Z_L och X_L jämfördes dessutom med ytkodernas initiala logiska tillstånd för de ideala simuleringarna.

3.3 Körning på kvantdator

Rutnätet med 10x12 kvantbitar för kvantprocessorn *Nighthawk r1* tillåter i teorin CNOT att implementeras fysiskt med den kompakta formen, eller enklare lattice surgery mellan endast två ytor för den konventionella formen. Det visade sig dock för det specifika chippet som användes via *IBM Miami*, att många anslutningar mellan kvantbitar var defekta, med rapporterad 100% felsannolikhet för CZ-grindar mellan. Därmed användes den kompakta formen, med endast ett möjligt val av positionering av kvantbitarna på chippet.

På kvantdatorn testades först ytkoderna separat, och därefter testades ZZ-merge mellan två ytor som initialiserades i olika tillstånd. I det förstnämnda fallet användes en enstaka ytkod, som alltid förbereddes i $|0\rangle_L$. Koden genomgick ett varierande antal stabilisatorrundor mellan 2 och 15, och logiska Z_L mättes på slutet, som efter korrekt avkodning förväntades ge $|0\rangle_L$. För ZZ-merge gjordes först triviala tester där ytorna förbereddes i dels $|00\rangle_L$, $|01\rangle_L$, $|10\rangle_L$, $|11\rangle_L$, också med varierande antal rundor före och efter merge. Till sist testades det också ifall ett Bell-tillstånd $|00\rangle_L + |11\rangle_L$ eller $|01\rangle_L + |10\rangle_L$, kunde skapas genom en ZZ-merge av initialtillstånden $|{+}{+}\rangle_L = |00\rangle_L + |01\rangle_L + |10\rangle_L + |11\rangle_L$, beroende på $Z_L^A Z_L^B$-mätutfallet M_{ZZ}. De logiska tillstånden mättes på samma sätt, alltid genom att mäta datakvantbitarna i beräkningsbasen på slutet.

För samtliga tester kördes samma program med 1000 upprepningar på instansen *IBM Miami*. Resultatdatan, i form av mätutfall hos ancillakvantbitarna per stabilisatorrunda och datakvantbitarna vid den slutliga mätningen, användes för att beräkna det nominella mätutfallet av Z_L^A, Z_L^B och $Z_L^A Z_L^B$, och omvandlades till detektorer, enligt avsnitt 2.7. Detektorerna användes med MWPM-grafen för att avkoda felen och återskapa de korrekta kvanttillstånden. Dessa sparades till sist i ett histogram, parametriserat över de relevanta logiska mätutfallen.

3.4 Logisk CNOT

Kod för en logisk CNOT-krets med lattice surgery, både för den konventionella och kompakta formen, skrevs och finns i Appendix B. Kretsen kan delas in i tre moment:

- Tre ytkoder initialiserades enligt metodiken beskriven i sektion 3.1. De betecknades C, INT och T och var placerade enligt figur 2.6.

- Stegen som presenteras i 2.8.2 utfördes för att implementera den logiska CNOT-operationen.
- Alla datakvantbitar mättes i Z-basen för att testa kretsen.

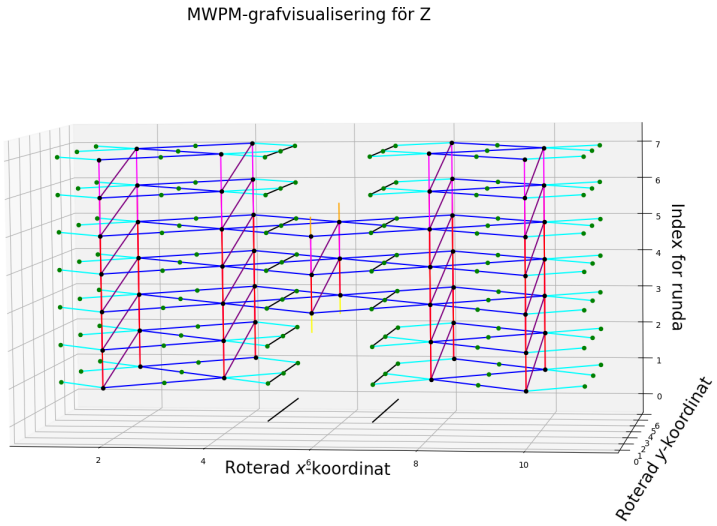
Tre stabilisatorrundor genomfördes för varje lattice surgery moment; innan ZZ-merge, efter ZZ-merge, efter ZZ-split, efter XX-merge och till sist efter XX-split. Datakvantbitarna som användes för de logiska Z-operationerna var de som låg närmast gränsen för ZZ-merge och för ytkod T användes datakvantbitarna längst till höger.

För att testa CNOT-kretsen utfördes ideala simuleringar utan fel. Ytkoderna C och T förbereddes i alla möjliga kombinationer av logiska tillstånd i den logiska beräkningsbasen, $|0\rangle_L$ och $|1\rangle_L$. Flera simuleringar utfördes för varje kombination. Resultatet från simuleringarna jämfördes med värdena för en teoretisk CNOT.

3.5 Avkodning

Avkodningen hanterades av en *Minimum Weight Perfect Matching* (MWPM)-algoritm, implementerad via biblioteket *PyMatching*. Med biblioteket kunde en matchingsgraf skapas som sedan användes för avkodningen. För att skapa matchningsgrafen gjordes varje detektor till en nod och spatiella samt temporala kanter drogs mellan dem som beskrivs i avsnitt 2.7.1.

Kanterna kunde korrespondera mot logiska observabler. Detta innebar att om särskilda kanter ingick i syndromgrafen, skulle ett värde kopplat till observabeln returneras från PyMatchings avkodningsfunktion. Detta värde motsvarade pariteten av antalet kanter som ingick i syndromgrafen och korresponderade mot den logiska observabeln. Exempel på logiska observabler som användes var logisk-Z och -X för varje ytkod. Om till exempel ett X-fel (bit-flip) skulle inträffa på en av datakvantbitarna som användes för att beräkna mätutfallet av Z_L, skulle en kant som korresponderar mot observabeln logisk-Z ingå i syndromgrafen. Koden för felkorrigeringen finns i Appendix B.



Figur 3.1: MWPM-graf som används i det specifika fallet av ZZ-merge mellan två ytor. I grafen markeras spatiala kanter (cyan och blå), temporala kanter (röd, orange gul och rosa) och diagonala hookfel (lila). Datakvantbitar är markerade som gröna prickar, ancillabitar som svarta prickar, och logiska Z-observabler är utritade som svarta linjer.

4

Resultat

Resultaten kan delas upp i dels ideala simuleringar, simuleringar med en brusmodell som propagerar genom kvantkretsen, och när den fysiska kvantprocessorn användes. Det verifierades med simulering att en enskild ytkod bevarade de logiska Z- och X-tillstånden, där alla kombinationer av ett fel injicerades på olika datakvantbitar. Det verifierades också hur deterministiska Pauli-fel på ancillor kunde leda till hook-fel. För brusfri lattice surgery överensstämde utfallet M_{ZZ} vid ZZ-merge, med pariteten av de logiska tillstånden vid initialisering i Z-basen. Slutligen verifierades också en sanningstabell för CNOT, för rena Z-egentillstånd, både med kompakta och konventionella formen. Resultatet av detta överensstämde med teorin.

4.1 Ideal lattice surgery

Vid ideala simuleringar av kvantkretsen för ZZ-merge och split erhölls resultatet som presenteras i tabell 4.1. Varje kombination av initiala logiska tillstånd testades 100 gånger per kombination. Kvanttillstånden längst till vänster motsvarar de logiska tillstånden som ytkoderna initialiserades till. I kolumnerna till höger visas mätutfallen av Z_L för båda ytkoderna, givet att $M_{ZZ} = 0$ respektive $M_{ZZ} = 1$.

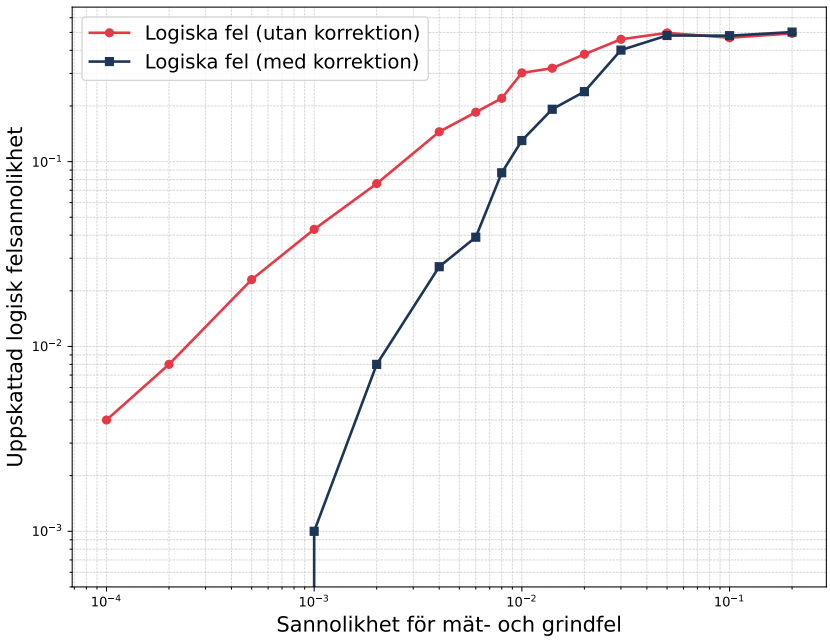
Tabell 4.1: Detaljerad fördelning av logiska utfall uppdelat på paritetsmätningen M_{ZZ}. Resultaten erhölls genom att simulera varje kombination av initiala logiska tillstånd för A och B 100 gånger utan brus.

		$M_{ZZ} = 0$				$M_{ZZ} = 1$			
A	B	$\|00\rangle_L$	$\|01\rangle_L$	$\|10\rangle_L$	$\|11\rangle_L$	$\|00\rangle_L$	$\|01\rangle_L$	$\|10\rangle_L$	$\|11\rangle_L$
$\|0\rangle_L$	$\|0\rangle_L$	100	0	0	0	0	0	0	0
$\|0\rangle_L$	$\|1\rangle_L$	0	0	0	0	0	100	0	0
$\|0\rangle_L$	$\|+\rangle_L$	47	0	0	0	0	53	0	0
$\|0\rangle_L$	$\|-\rangle_L$	49	0	0	0	0	51	0	0
$\|1\rangle_L$	$\|1\rangle_L$	0	0	0	100	0	0	0	0
$\|1\rangle_L$	$\|+\rangle_L$	0	0	0	52	0	0	48	0
$\|1\rangle_L$	$\|-\rangle_L$	0	0	0	55	0	0	45	0
$\|+\rangle_L$	$\|+\rangle_L$	19	0	0	30	0	24	27	0
$\|+\rangle_L$	$\|-\rangle_L$	31	0	0	30	0	19	20	0
$\|-\rangle_L$	$\|-\rangle_L$	24	0	0	22	0	29	25	0

Tabellen exkluderar spegelvända kombinationer (till exempel $|0\rangle_L, |1\rangle_L$ jämfört med $|1\rangle_L, |0\rangle_L$) och mätutfall från XX-merge och split. Dessa fall testades också, och det kunde konstateras att de stämde överens med resultaten i tabellen, med undantag för statistiska variationer och bytet av 0 med + och 1 med − för XX-merge och split.

4.2 Simuleringar med brusmodell

När en brusig modell användes för att simulera stabilisatorrundor på en ytkod över tid inträffade ibland logiska fel (mätningen av Z_L eller X_L var i konflikt med det förberedda tillståndet). Figur 4.1 visar på y-axeln de uppskattade logiska felsannolikheterna, som en funktion av den gemensamma sannolikheten för mät- och grindfel som användes. Totalt utfördes 1000 simuleringar per sannolikhet med 5 stabilisatorrundor per körning.

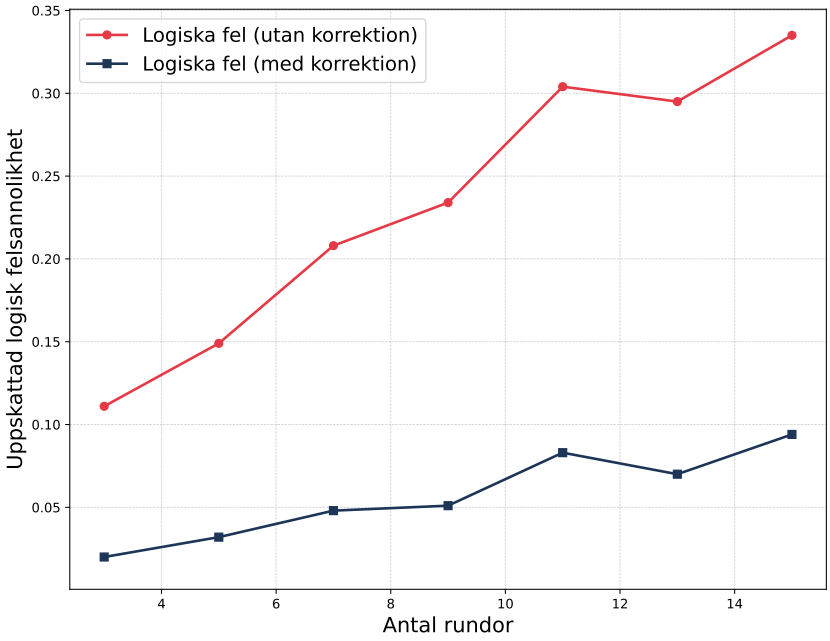


Figur 4.1: Jämförelse av uppskattade logiska felsannolikheter för fem stabilisatorrundor på en ytkod ($d = 3$) vid olika sannolikheter för mät- och grindfel. 1000 simuleringar användes för att uppskatta den logiska felsannolikheten per datapunkt. Grafen visas på en log-log-skala.

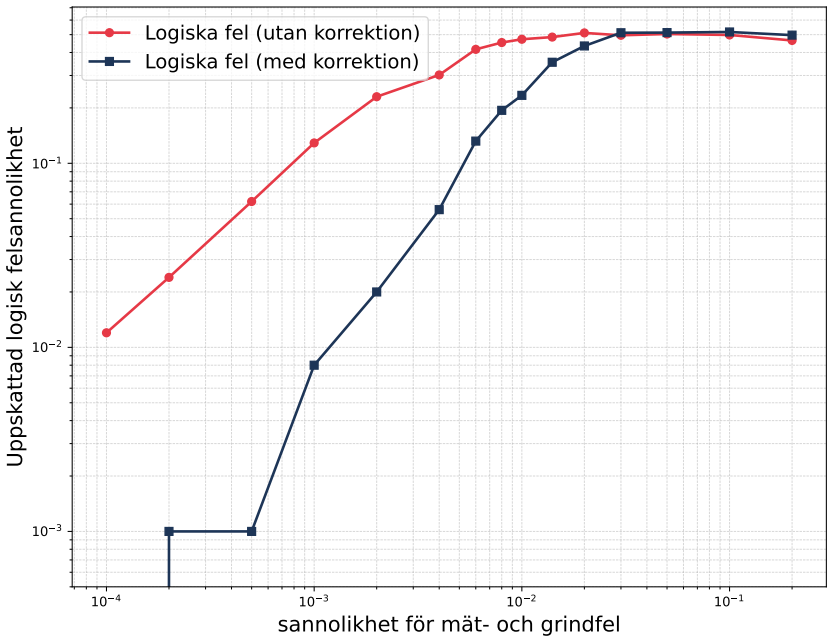
När sannolikheten för mät- och grindfel hölls konstant, $p_{\text{mät}} = p_{\text{grind}} = 0{,}5\%$, men antalet stabilisatorrundor varierades, erhölls resultatet som presenteras i figur 4.2.

4.2.1 Lattice surgery

Vid simuleringar av ZZ-merge följt av split (lattice surgery) där en modell med brus applicerades, blev felkorrigering relevant. De statistiska utfallen när avkodade värdet på M_{ZZ} inte överensstämde med det förväntade mätvärdet från operatorn $Z_L^A Z_L^B$, presenteras i figur 4.3. I figuren motsvarar x-axeln sannolikheten för mät- och grindfel, $p_{\text{mät}} = p_{\text{grind}}$ och y-axeln den uppskattade logiska felsannolikheten baserat på 1000 simuleringar. Totalt utfördes nio stabilisatorrundor. Ett logiskt fel innebar att M_{ZZ} och mätvärdet av $Z_L^A Z_L^B$ hade olika värden. Datapunkterna bestående av cirklar motsvarar felsannolikheten utan felkorrektion och datapunkterna bestående av kvadrater motsvarar felsannolikheten med felkorrektion. Båda ytkoderna förbereddes i $|0\rangle_L$ och antalet syndromrundor var nio uppdelade i tre rundor innan merge, tre efter merge och tre efter split. Liknande grafer erhölls även vid XX-merge och split och när olika initiala tillstånd testades.



Figur 4.2: Jämförelse av uppskattade logiska felsannolikheter för en ytkod ($d = 3$) som genomgår olika antal stabilisatorrundor. 1000 simuleringar användes för att uppskatta den logiska felsannolikheten per datapunkt.

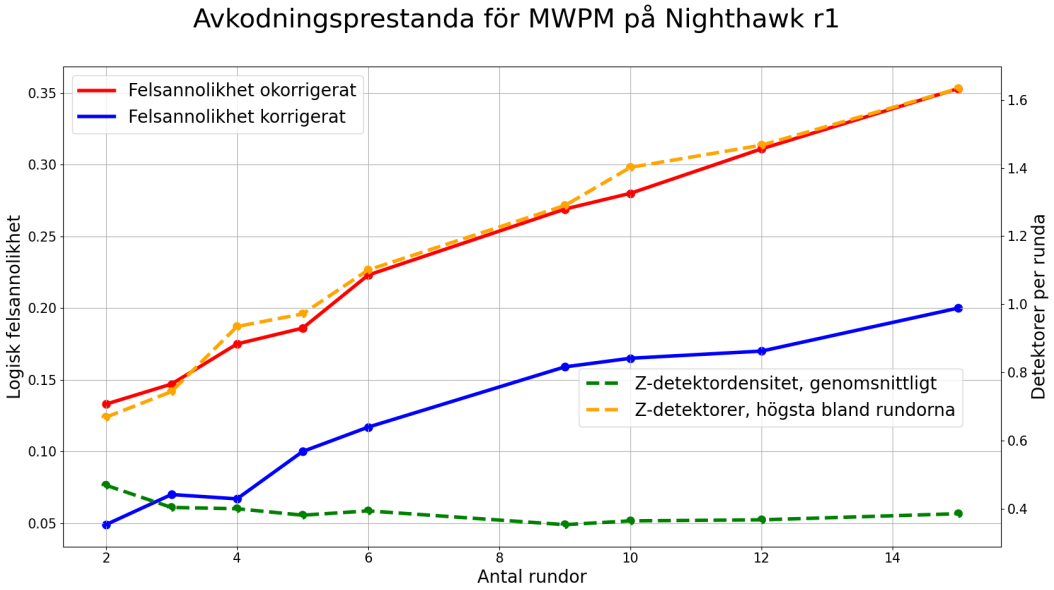


Figur 4.3: Jämförelse av logiska felsannolikheter för lattice surgery-operationer ($d = 3$) vid olika sannolikheter för mät- och grindfel. Båda graferna visas på en log-log-skala.

4.3 Körning på IBM Nighthawk r1

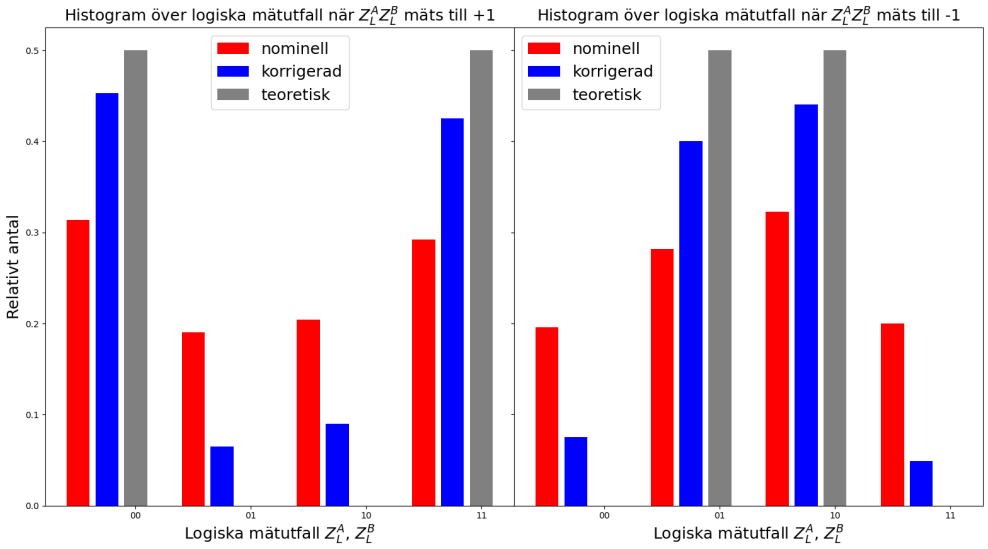
Resultaten för en enskild ytkod körd på *IBM Nighthawk r1* visas i figur 4.4, där ytorna initialiserades till $|0\rangle_L$, ett visst antal stabilisatorrundor utfördes, och där logiska Z_L till sist mättes.

Figuren visar hur antalet logiska fel, både före och efter korrektion, successivt ökar i takt med att antalet rundor ökar. Detektordensiteten hos rundan med flest detektorer ökar successivt, och genomsnittliga detektordensiteten är mestadels oförändrad.



Figur 4.4: Logisk felsannolikhet och detektordensitet hos en enkel ytkod när antalet rundor varieras, vid körning på kvantprocessorn *Nighthawk r1* med 1000 iterationer per körning. Den logiska ytkoden förbereds i $|0\rangle_L$, där stabilisatorer mäts i n rundor (x-axeln) följt av en slutlig Z_L-mätning. Likt simuleringen i figur 4.2 ökar sannolikheten för logiska fel med antalet rundor, relativt likformigt mellan korrigerade och okorrigerade, men långsammare med korrigering. Den genomsnittliga detektordensiteten per runda (och iteration) är relativt oförändrad när antalet rundor varieras. Antalet detektorer hos rundan där flest detektorer förekom, ökar däremot stadigt (medelvärderat över iterationerna).

Figur 4.5 visar ett histogram över Z-mätutfall för respektive yta efter ZZ-merge, när ett Belltillstånd skapas. Histogrammet är uppdelat baserat på ZZ-utfallet, och antalen är relativt det totala antalet vid respektive ZZ-utfall. Tillståndet förbereds genom att först initialisera det till $|{+}{+}\rangle_L = |00\rangle_L + |01\rangle_L + |10\rangle_L + |11\rangle_L$, utföra ZZ-merge, och till sist mäta ut ytorna i logiska Z-basen. När $Z_L^A Z_L^B$ mäts vid merge kollapsar tillståndet antingen till $|00\rangle_L + |11\rangle_L$ (ZZ-egenvärde +1) eller $|01\rangle_L + |10\rangle_L$ (egenvärde -1), som i båda fallen är Belltillstånd. De teoretiska tillstånden som beror på ZZ-egenvärdet överensstämmer relativt väl med de felkorrigerade mätutfallen i figuren, medan de okorrigerade är fördelade mer likformigt. Även med avkodning mäts dock de avkodade tillstånden som icke-Bell-tillstånd vid runt 15% av fallen.



Figur 4.5: Z-mäthistogram vid förberedelse av Bell-tillstånd, exekverat på *IBM Nighthawk r1*, där den kompakta formen används. Tillståndet förbereds genom att initialisera ytorna till $|{+}{+}\rangle_L$, utföra ZZ-merge, och därefter mäta Z_L. En ZZ-merge är ekvivalent med en ZZ-mätning, så teoretiskt kommer ZZ-utfallen för ett initialtillstånd $|00\rangle_L + |01\rangle_L + |10\rangle_L + |11\rangle_L$ antingen bli +1, där tillståndet kollapsar till Bell-tillståndet $|00\rangle_L + |11\rangle_L$, eller -1, där tillståndet istället kollapsar till $|01\rangle_L + |10\rangle_L$. I vardera histogram är de nominella mätutfallen svagt urskiljbara, medan de avkodade mätutfallen är betydligt närmre det som teoretiskt förväntas. De kombinerade nominella utfallen placeras i det vänstra respektive högra histogrammet baserat på det nominella ZZ-värdet, och på samma sätt för de felkorrigera utfallen. Resultaten samplas från 1000 iterationer.

Mätresultat för ytterligare konfigurationer för $|{+}{+}\rangle_L$ visas i tabell 4.2, både när ZZ mäts till 0 eller till 1. Tabellen visar hur fördelningen mellan utfall är nästintill likformig utan felkorrektion, men är betydligt närmre den teoretiska när felkorrektion används. Rundor anges som antal rundor före merge, under merge, och efter split. Prestandan fås genom att summera antalet $|00\rangle_L$- och $|11\rangle_L$-utfall när $M_{ZZ} = 0$, med antalet $|01\rangle_L$ och $|10\rangle_L$ när $M_{ZZ} = 1$, och är exempelvis 86% för (6,6,0) rundor korrigerat. Ifall ytterligare rundor lades till när ytkoderna delades upp (split), såsom (3,3,3) rundor, blev mätutfallen mer jämnt fördelade i kontrast till teorin. Som referens testades också triviala ZZ-merge med initialtillståndet $|00\rangle_L$, där dock endast 68% av fallen mättes till $|00\rangle_L$ med (3,3,0) rundor, och enbart 56% när 3 split-rundor lades till. När fördelningarna marginaliseras efter ytorna separat, avkodas dock vardera ytkod korrekt över 85% av gångerna vid (3,3,0) rundor.

Tabell 4.2: Histogram över mätresultat för ZZ-merge, följt av logiska Z för ytorna, efter att ha förberetts i $|{+}{+}\rangle_L$. Datan presenteras både med och utan felkorrektion, för olika konfigurationer. För varje konfiguration har 1000 iterationer körts på kvantprocessorn *IBM Nighthawk r1*. Kolumnen ”rundor” delas upp i antalet stabilisatorrundor före merge, under merge, respektive efter split. Till vänster visas antalen utfall när ZZ mättes till 0, och till höger när de mättes till 1. Till skillnad från i figur 4.5 är andelarna relativt de 1000 totala körningarna. Utfallen utan split-rundor är relativt lika och nära det teoretiskt förväntade, medan förekomsten av split-rundor ökar antal gånger tillståndet som mäts inte är ett Belltillstånd.

		$M_{ZZ} = 0$				$M_{ZZ} = 1$			
Fall	Rundor	$\|00\rangle_L$	$\|01\rangle_L$	$\|10\rangle_L$	$\|11\rangle_L$	$\|00\rangle_L$	$\|01\rangle_L$	$\|10\rangle_L$	$\|11\rangle_L$
Utan	(3, 6, 0)	16,1%	9,9%	10,8%	13,5%	11,1%	14,8%	13,7%	10,1%
Med	(3, 6, 0)	22,9%	2,8%	3,0%	21,0%	4,0%	23,7%	18,0%	4,6%
Utan	(3, 3, 0)	16,0%	9,7%	10,4%	14,9%	9,6%	13,8%	15,8%	9,8%
Med	(3, 3, 0)	23,1%	3,3%	4,6%	21,7%	3,7%	19,6%	21,6%	2,4%
Utan	(6, 6, 0)	15,1%	8,8%	11,5%	14,0%	9,0%	15,1%	15,5%	11,0%
Med	(6, 6, 0)	23,2%	3,1%	3,0%	20,2%	4,4%	21,5%	21,1%	3,5%
Utan	(3, 3, 3)	14,4%	10,5%	11,2%	16,6%	10,3%	13,8%	14,3%	8,9%
Med	(3, 3, 3)	19,4%	7,4%	7,3%	17,0%	8,3%	16,2%	18,1%	6,3%
Teoretisk	(oberoende)	25%	0%	0%	25%	0%	25%	25%	0%

5
Diskussion

Diskussionen behandlar inledningsvis simuleringar med förutsägbara fel samt lattice surgery under påverkan av slumpmässiga fel på kretsnivå, för att därefter övergå till resultaten från den fysiska kvantprocessorn. Avslutningsvis förs en diskussion kring potentiella förbättringar av avkodningsprocessen, samt vilka generella implikationer och etiska aspekter arbetet medför.

5.1 Simuleringar

Från resultatet framgår det som förväntat att en ytkod kan bevara ett logiskt kvanttillstånd över ideala stabilisatorrundor. Även enstaka fel kunde korrigeras så att tillståndet förblev detsamma, vilket framgick av testerna med injicerade fel. Därav är det också rimligt att den logiska felsannolikheten blir nästintill noll vid små värden på mät- och grindfel, vilket syntes i figur 4.1. Däremot, när sannolikheten för mät- och grindfel ökades, ökade även den logiska felsannolikheten och vid $p_{\text{mät}} = p_{\text{grind}} = 4\%$ var utfallen av de logiska operatorerna nästintill slumpmässiga. Från figur 4.2 syns att antalet stabilisatorrundor för en ytkod också ökar den logiska felsannolikheten, både med och utan felkorrektion. Detta är rimligt med tanke på att fler fel förväntas med fler stabilisatorrundor.

5.1.1 Lattice surgery

Från tabell 4.1, som baseras på brusfria simuleringar, framgår det att logiska tillstånd i samma bas som typen av merge (exempelvis ett logiskt tillstånd i Z-basen, $|00\rangle_L$ och ZZ-merge) bevaras under lattice surgery. Logiska tillstånd i motsatt bas kollapsade istället vid merge, där till exempel $M_{ZZ} = Z_L^A Z_L^B$ mättes. Efter kollapsen vid ZZ-merge bevarades den totala pariteten av $Z_L^A Z_L^B$. Detta framgick av den slutgiltiga mätningen av $Z_L^A Z_L^B$ som alltid överensstämde med M_{ZZ}. För fallet där båda de logiska tillstånden initialiserades i motsatt bas, innebar det att tillstånden kollapsade till $|00\rangle_L + |11\rangle_L$ om $M_{ZZ} = 0$ eller $|01\rangle_L + |10\rangle_L$ om $M_{ZZ} = 1$, som är Bell-tillstånd.

Från figur 4.3 framgår det att felkorrigeringen för lattice surgery hade liknande effekt som felkorrigeringen av en enskild ytkod. Jämfört med en ytkod var den korrigerade och icke-korrigerade logiska felsannolikheten lite högre. Detta är rimligt med tanke på att fler stabilisatorrundor genomfördes (totalt 9) och att kretsen bestod av ungefär dubbelt så många individuella kvantbitar. Vid ca $p_{\text{mät}} = p_{\text{grind}} = 2\%$ verkar utfallen av de logiska operatorerna vara helt slumpmässiga, även med korrektion.

5.2 Fysisk körning

Prestandan för den enkla ytkoden som presenteras i figur 4.4 kan anses tillräcklig för ändamålen i de här experimenten, med mindre än 7% logiska fel som lägst. Sannolikheten för logiska fel är

lägre och verkar öka långsammare med avkodningen jämfört med utan avkodning. I genomsnitt är Z-detektordensiteten låg, och relativt oberoende av antalet rundor, men det högsta antalet detektorer per runda ökar stadigt likt de logiska felsannolikheterna. Eftersom en distans-3-kod endast kan korrigera 1 fysiskt fel, och fysiska datafel som högst kan aktivera två detektorer, borde en högsta densitet runt 1,6 (per runda) vid 15 rundor innebära att det inträffar okorrigerbara logiska fel i ett betydande antal iterationer. Vid 15 rundor förekom fler än 2 detektorer hos någon av rundorna vid 13% av iterationerna, och i 5% av iterationerna vid 6 rundor. Dock förbättrades inte prestandan avsevärt ifall iterationerna med fler detektorer förkastades. Det tyder på att förbättringar i MWPM-grafen troligen också hade behövts.

Kvantprocessorn *IBM Nighthawk r1* är i skrivande stund ”experimentell” enligt IBM [29], där bland annat villkorade grindar inte stöds, och framförallt att mätfelssannolikheten är runt en tiopotens större än för de mer etablerade *Heron r3*-processorerna (med annan topologi). En högre mätfelssannolikhet medför förstås en ökad risk för logiska fel, dels vid Z_L-mätningen på slutet, och dels eftersom det kan ta spatiala kanters plats i MWPM-grafen, beroende på hur väl avkodaren kan urskilja det mest sannolika. Virtuella rundan bör även ha 2-4 gånger högre mätfelssannolikhet eftersom samtliga datakvantbitar snarare än enskilda ancillabitar mäts per stabilisator, som dock delvis möjligen kompenseras av att virtuella rundan inte utför grindar. Vid vanliga stabilisatormätningar behöver förstås endast en kvantbit mätas.

Överlag visar resultaten på kvantdatorn att avkodningen relativt väl kan återskapa de rena Z_L-tillstånd som förväntas efter initialiseringen till antingen ren Z_L eller X_L. Avkodningen har därmed enbart behövt göras i en av operatorerna, eftersom Z_L förstås kommuterar med Z_L och X_L med X_L. Däremot har det inte testats några fall med samtidig Z- och X-avkodning, vilket hade krävts för att avkoda godtyckliga tillstånd $|\psi\rangle_L = \alpha\,|0\rangle_L + \beta\,|1\rangle_L$. Tekniken *tillståndsinjektion* kan användas för att skapa ytkoder för godtyckliga tillstånd, genom att initialisera en enstaka kvantbit och därefter successivt sammanfläta till resten av ytkoden [15], vilket hade kunnat användas för mer omfattande testning.

5.3 Felkällor och potentiella förbättringar

En potentiell förbättring av MWPM-felkorrigeringen grundas i att Z- och X-fel behandlades som oberoende. Detta är en förenkling då det lika gärna kan förekomma Y-fel bestående av ett samtidigt Z- och X-fel, vilket medför en korrelation. Det finns olika metoder för att avkoda korrelerade fel, såsom *Iterative Reweighting Minimum Weight Perfect Matching* (IRMWPM) där MWPM upprepas och vikterna justeras varje iteration efter korrelationer [34].

Algoritmen som förberedde MWPM-grafen för avkodning av mätdatan, associerade en konstant vikt för alla spatiala kanter, hook-kanter och närliggande temporala kanter, där endast tvåstegs-kanterna motsvarande mätfel var dynamiskt anpassade till de rapporterade mätfelen hos kvantdatorn. Eftersom sannolikheten för Pauli-fel skiljer sig mellan olika fysiska kvantbitar, och för CZ-grindarna mellan, är det därför endast en approximation att anta konstanta vikter, men det hade då krävts antingen (1) att de fysiska felen propagerar genom en kretsmodell, eller (2) att vikterna anpassas dynamiskt med exempelvis maskininlärning. Vidare hade också sannolikheterna hos den sista virtuella rundan kunnat justeras efter mätfelssannolikheten hos datakvantbitarna. Kretsmodellen som användes vid simuleringar modellerade endast mät- och grindfel, och hade till fördel kunnat utökas för att också inkludera passiva Pauli-fel, och hur de påverkar MWPM-vikterna.

En dynamisk justering av MWPM-vikterna framstår som en komplex uppgift att implementera enbart utifrån en modell av kretsen. Ett växande forskningsområde är att tillämpa grafneurala

nätverk eller andra varianter, vilket har möjligheten att förbättra felkorrektionsprestandan, ofta på bekostnad av att avkodningen blir mer beräkningstung [17]. En annan potentiell förbättring hade varit att läsa analoga mätsignaler, eftersom det i teorin då är möjligt att få en uppskattning på signifikansen hos mätningarna, och möjligen anpassa MWPM-vikterna därefter. En möjlig metod kallas *soft information decoding* [35].

5.4 Samhälleliga och etiska aspekter

Mycket av den digitala kommunikationen idag är beroende av kryptering för att skydda känslig information mot potentiell avlyssning, vilket konventionellt (RSA) bygger på att vissa problem såsom primtalsfaktorisering är svåra för klassiska datorer att utföra [36]. Det har länge varit känt att kvantdatorer effektivt kan faktorisera tal med Shors algoritm [1], och betraktas därför som ett hot mot konventionell kryptografi. Hotet har bland annat lett till så kallad *post-kvantkryptografi*, som bygger på andra matematiska problem som inte med kända algoritmer kan lösas mer effektivt av kvantdatorer. Samtidigt finns risken att aktörer kan lagra krypterad information idag, med hoppet att det i framtiden enkelt kommer kunna dekrypteras i framtiden [36]. Det finns olika uppskattningar på hur många felkorrigerade logiska kvantbitar som behövs för Shor, men exempelvis enligt [37] hade det krävts runt 1600 logiska kvantbitar och en miljon fysiska för att bryta RSA-2048. Detta är förstås många ordningar högre än de två logiska kvantbitarna som högst uppnås i det här arbetet, men utgör en risk i det större sammanhanget.

5.5 Slutsats

I det här arbetet har kvantfelskorrektion med ytkoder undersökts med fokus på lattice surgery och dess tillämpningsområden, där det har visats hur metoden på en fysisk kvantdator kan användas för att skapa ett Belltillstånd, med relativt hög andel korrekta utfall. Kvantitativt har också ytkodernas prestanda undersökts på kvantdatorn och i simuleringar. För felkorrigering har MWPM kunnat ge goda resultat i kretsmodell-simuleringar och relativt goda för fysiska körningar. Det finns dock troligen förbättringsmöjligheter för hur vikterna i MWPM-grafen mest effektivt ska väljas, då grafen i det här arbetet mestadels använder konstanta vikter.

Vidare implementerades och verifierades CNOT med lattice surgery genom simuleringar, där kretsen dock inte kunde testas på kvantdatorn på grund av ett begränsat antal välfungerande kvantbitar. Trots hårdvarubegränsningarna demonstrerar simuleringarna av CNOT potentialen av lattice surgery. Med hjälp av lattice surgery kan den fundamentala logiska CNOT-operationen genomföras på en QPU med kvadratisk topologi, vilket tyder på att roterade ytkoder och lattice surgery kan vara en effektiv metod för att möjliggöra feltoleranta kvantberäkningar. När större kvantprocessorer finns tillgängliga kan ytkoderna troligen både göras större, för att öka kodavståndet, och fler, för att möjliggöra mer komplexa operationer.

Litteratur

1. Shor P. Algorithms for quantum computation: discrete logarithms and factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science*. 1994 :124–34. DOI: 10.1109/SFCS.1994.365700
2. G. S. Quantum Computing: What It Is, Why We Want It, and How We're Trying to Get It. *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2018 Symposium*. Washington, DC: The National Academies Press, 2019 :6–7. ISBN: 978-0-309-48750-4. DOI: 10.17226/25333
3. Grover LK. A fast quantum mechanical algorithm for database search. *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. STOC '96. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996 :212–9. ISBN: 0897917855. DOI: 10.1145/237814.237866
4. Harrow AW, Hassidim A och Lloyd S. Quantum Algorithm for Linear Systems of Equations. Phys. Rev. Lett. 2009 Oct; 103(15):150502. DOI: 10.1103/PhysRevLett.103.150502
5. Peruzzo A, McClean J, Shadbolt P, Yung MH, Zhou XQ, Love PJ, Aspuru-Guzik A och O'Brien JL. A variational eigenvalue solver on a photonic quantum processor. Nature Communications 2014 Jul; 5:4213. ISSN: 2041-1723. DOI: 10.1038/ncomms5213
6. National Institute of Standards and Technology. Quantum Computing Explained. [Hämtad 2026-04-16]
7. Roffe J. Quantum Error Correction: An Introductory Guide. Contemporary Physics 2019; 60:226–45. DOI: 10.1080/00107514.2019.1667078
8. Wootters WK och Zurek WH. A single quantum cannot be cloned. Nature 1982; 299:802–3
9. Shor PW. Scheme for reducing decoherence in quantum computer memory. Physical Review A 1995; 52:R2493–R2496. DOI: 10.1103/physreva.52.r2493
10. IBM. Threshold theorem. [Hämtad 2026-04-21]. URL: https://quantum.cloud.ibm.com/learning/en/courses/foundations-of-quantum-error-correction/fault-tolerant-quantum-computing/threshold-theorem
11. Bravyi SB och Kitaev AY. Quantum codes on a lattice with boundary. 1998. DOI: 10.48550/ARXIV.QUANT-PH/9811052
12. Quantum repetition code. *The Error Correction Zoo*. Utg. av Albert VV och Faist P. 2022. URL: https://errorcorrectionzoo.org/c/quantum_repetition
13. Lee YC, Brell CG och Flammia ST. Topological quantum error correction in the Kitaev honeycomb model. Journal of Statistical Mechanics: Theory and Experiment 2017 Aug; 2017:083106. ISSN: 1742-5468. DOI: 10.1088/1742-5468/aa7ee2
14. Ivezic M. Surface Code Quantum Error Correction. https://postquantum.com/quantum-computing/surface-code-qec/. [Hämtad 28-04-2026]
15. Horsman D, Fowler AG, Devitt S och Van Meter R. Surface code quantum computing by lattice surgery. New Journal of Physics 2012; 14:123011. DOI: 10.1088/1367-2630/14/12/123011
16. Higgott O och Gidney C. Sparse Blossom: correcting a million errors per core second with minimum-weight perfect matching. Quantum 2023 Aug; 7:1087. ISSN: 2521-327X. DOI: 10.22331/q-2023-08-31-1087
17. Lange M, Havström P, Srivastava B, Bengtsson I, Bergentall V, Hammar K, Heuts O, Nieuwenburg E van och Granath M. Data-driven decoding of quantum error correcting codes using graph neural networks. Phys. Rev. Res. 2025 May; 7(2):023181. DOI: 10.1103/PhysRevResearch.7.023181
18. Mandelbaum R. Scaling for quantum advantage and beyond. Hämtad: 2026-03-24. 2025 Nov 12. URL: https://www.ibm.com/quantum/blog/qdc-2025
19. Schneider J och Smalley I. What is a qubit? IBM. 2026 Apr 2. URL: https://www.ibm.com/think/topics/qubit

20. Benenti G, Casati G, Rossini D och Strini G. Principles of Quantum Computation and Information: A Comprehensive Textbook. Singapore: World Scientific Publishing Company, 2019. ISBN: 9789813237223
21. Nielsen MA och Chuang IL. Quantum Computation and Quantum Information. Cambridge: Cambridge University Press, 2010. ISBN: 9781107002173
22. Gottesman D. Stabilizer codes and quantum error correction. Diss. Pasadena, CA: California Institute of Technology, 1997
23. Williams CP. Quantum Gates. *Explorations in Quantum Computing*. London: Springer London, 2011 :51–122. ISBN: 978-1-84628-887-6. DOI: 10.1007/978-1-84628-887-6_2
24. Kottmann K. (Clifford + T) Gate Set. https://pennylane.ai/compilation/clifford-t-gate-set. Accessed: 2026-05-11. 2025
25. Gottesman D. An Introduction to Quantum Error Correction and Fault-Tolerant Quantum Computation. arXiv preprint 2009. arXiv: 0904.2557 [quant-ph]. URL: https://arxiv.org/abs/0904.2557
26. Fowler AG, Mariantoni M, Martinis JM och Cleland AN. Surface codes: Towards practical large-scale quantum computation. Physical Review A 2012; 86:032324
27. IBM Quantum Learning. Discretization of Errors. Accessed: 2026-05-15. 2025. URL: https://quantum.cloud.ibm.com/learning/en/courses/foundations-of-quantum-error-correction/correcting-quantum-errors/discretization-of-errors
28. O'Rourke AR och Devitt S. Compare the pair: Rotated versus unrotated surface codes at equal logical error rates. Physical Review Research 2025; 7:033074. DOI: 10.1103/PhysRevResearch.7.033074
29. IBM Quantum. Nighthawk and the latest Heron are now available. Product Updates. IBM. 2026 Jan 5. URL: https://quantum.cloud.ibm.com/announcements/en/product-updates/2026-01-05-nighthawk [Hämtad 2026-05-08]
30. Higgott O. PyMatching: A Python Package for Decoding Quantum Codes with Minimum-Weight Perfect Matching. ACM Transactions on Quantum Computing 2022 Aug; 3. DOI: 10.1145/3505637
31. Vuillot C, Lao L, Criger B, Almudéver CG, Bertels K och Terhal BM. Code Deformation and Lattice Surgery Are Gauge Fixing. 2018. arXiv: 1810.10037v2 [quant-ph]
32. Josh Schneider IS. What is a QPU (Quantum Processing Unit)? https://www.ibm.com/think/topics/qpu. [Hämtad 2026-04-28]
33. IBM. View Backend Details. Hämtad: 2026-05-11. URL: https://quantum.cloud.ibm.com/docs/en/guides/qpu-information#native-gates
34. Tian Y, Zheng YC, Wang X och Lai CY. Enhancing Fault-Tolerant Surface Code Decoding with Iterative Lattice Reweighting. ArXiv 2025; abs/2509.06756. URL: https://arxiv.org/abs/2509.06756
35. Hanisch MD, Hetényi B och Wootton JR. Soft information decoding with superconducting qubits. APS Open Sci. 2026 May; 1:000019. DOI: 10.1103/y9fh-4x6n
36. National Institute of Standards and Technology. What Is Post-Quantum Cryptography. https://www.nist.gov/cybersecurity/what-post-quantum-cryptography. Hämtad: 2026-02-09
37. Gidney C. How to factor 2048 bit RSA integers with less than a million noisy qubits. 2025. arXiv: 2505.15917 [quant-ph]. URL: https://arxiv.org/abs/2505.15917

A

Stabilisatorer

I tabell A.1 och A.2 visas stabilisatorerna för den kompakta respektive konventionella formen av lattice surgery. Strukturen hos kvantbitarna för respektive form visas i figur 2.5 respektive 2.4.

Tabell A.1: Stabilisatorer för den kompakta formen, i samband med ZZ-merge av två 3x3 roterade ytkoder, se figur 2.5. Stabilisatorerna $S_{\text{innan}} = S_1 \cup S_2$ anger stabilisatorerna innan ZZ-merge och stabilisatorerna i $S_{\text{efter}} = S_1 \cup S_2 \cup S_{\text{nya\_x}} \cup S_{\text{nya\_z}} \setminus S_{\text{borttagna}}$ anger stabilisatorerna efter ZZ-merge. För XX-merge gäller samma logik, bara att Z byter plats mot X på alla ställen.

S_1	S_2	**Borttagna**	**Nya** X
$Z_1Z_2Z_4Z_5$	$Z_9Z_{10}Z_{12}Z_{13}$		
$Z_3Z_4Z_6Z_7$	$Z_{13}Z_{14}Z_{16}Z_{17}$		
Z_0Z_1	$Z_{10}Z_{11}$		
Z_7Z_8	$Z_{15}Z_{16}$		
$X_0X_1X_3X_4$	$X_{10}X_{11}X_{13}X_{14}$		
$X_4X_5X_7X_8$	$X_{12}X_{13}X_{15}X_{16}$		
X_3X_6	X_9X_{12}	X_9X_{12}	$X_2X_9X_5X_{12}$
X_2X_5	$X_{14}X_{17}$	X_2X_5	
Nya Z			
Z_2Z_9			
$Z_5Z_{12}Z_8Z_{15}$			

Tabell A.2: Stabilisatorer för den konventionella formen, i samband med ZZ-merge av två 3x3 roterade ytkoder. Stabilisatorerna i $S_{\text{innan}} = S_1 \cup S_2$ anger stabilisatorerna innan ZZ-merge och stabilisatorerna i $S_{\text{efter}} = S_1 \cup S_2 \cup S_{\text{nya\_x}} \cup S_{\text{nya\_z}} \setminus S_{\text{borttagna}}$ anger stabilisatorerna efter ZZ-merge. För XX-merge gäller samma logik, bara att Z byter plats mot X på alla ställen.

S_1	S_2	**Borttagna**	**Nya** X
$Z_1Z_2Z_4Z_5$	$Z_{10}Z_{11}Z_{13}Z_{14}$		
$Z_3Z_4Z_6Z_7$	$Z_{12}Z_{13}Z_{15}Z_{16}$		
Z_0Z_1	Z_9Z_{10}		
Z_7Z_8	$Z_{16}Z_{17}$		
$X_0X_1X_3X_4$	$X_9X_{10}X_{12}X_{13}$		
$X_4X_5X_7X_8$	$X_{13}X_{14}X_{16}X_{17}$		
X_3X_6	$X_{12}X_{15}$	$X_{12}X_{15}$	$X_{G_1}X_{12}X_{G_2}X_{15}$
X_2X_5	$X_{11}X_{14}$	X_2X_5	$X_2X_5X_{G_0}X_{G_1}$
Nya Z			
$Z_2Z_{G_0}$			
$Z_{G_0}Z_9Z_{G_1}Z_{12}$			
$Z_5Z_{G_1}Z_8Z_{G_2}$			
$Z_{G_2}Z_{15}$			

B

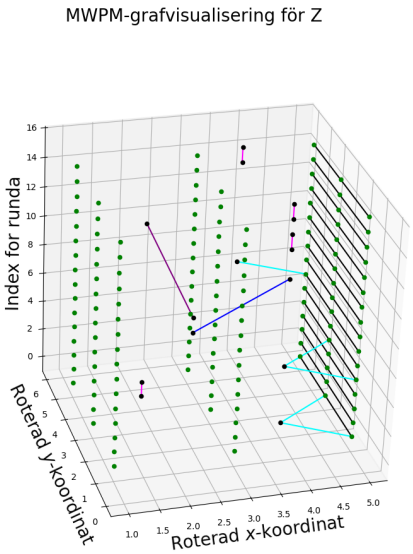
Kod

All kod som användes för att konstruera kvantkretsarna med biblioteken `qiskit` och `stim`, simulera, avkoda och köra på kvantdatorn, finns på GitHub vid `https://github.com/NguyenJT/LatticeSurgery`.

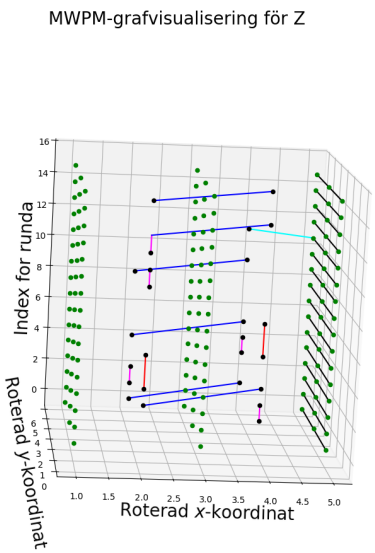
C

MWPM-grafer

Exempel på MWPM-graferna som bildades vid avkodning, för den enskilda ytkoden, visas i figur C.1 respektive figur C.2 där det övergår från fel till rätt med korrektion, respektive tvärtom. Axlarna i planet motsvarar ytkoden spatialt, där datakvantbitar är markerade som gröna prickar, och tidsaxeln är uppåt. Figurerna visar hur både spatiala kanter mellan detektorer (blå) och till gränsnoden (cyan) har dragits, och även temporala kanter (röd) och kanter motsvarande ancillafel (rosa). De möjliga kanterna som kan dras vid lattice surgery visas i figur 3.1.



Figur C.1: Realiserad MWPM-graf vid en av körningarna där den icke-korrigerade logiska observabeln inte överensstämde med initialtillståndet medan den korrigerade logiska observabeln gjorde det. Avkodaren valde en vanlig spatial kant (mörkblått), tre spatiala rand-kanter (cyan), fyra temporala ancilla-fel-kanter (rosa) och en hook-kant (lila).



Figur C.2: Realiserad MWPM-graf vid en av körningarna där den icke-korrigerade logiska observabeln överensstämde med initialtillståndet medan den korrigerade logiska observabeln inte gjorde det. Avkodaren valde sex vanliga spatiala kanter (mörkblå), en spatial rand-kant (cyan), två temporala mätfels-kanter (rött) och fem temporal ancilla-fel-kanter (rosa).

D

AI-verktyg

För att generera första utkastet av särskilda funktioner i koden, inklusive det första utkastet av MWPM-koden, användes hjälp av LLM-verktyg. Även om vissa mindre funktioner har återanvänts som de är, skrevs dock mycket av den koden om och/eller redigerades mycket manuellt i takt med att kodbasen fick unit-tester, och också för att säkerställa att koden gick att förstå på djupet. För koden som kördes på kvantdatorn har majoriteten skrivits för hand, där det automatiskt skrivna har tydligt lästs igenom och verifierats. En senare iteration av MWPM-koden med visualiseringsstöd skrevs också manuellt från första principer, mestadels oberoende av det första utkastet. Språkmodellerna användes också för en del ”manuellt” arbete såsom att generera vissa plots och omslagsbilden som föreställer en ytkod. Språkmodeller användes i begränsad utsträckning för att omformulera text i rapporten för att få ett bättre flyt, utan att lägga till nya koncept, fakta eller definitioner. De få ställen där språkmodeller har använts i texten har gåtts igenom noggrant och har kontrollerats för korrekthet.

INSTITUTIONEN FÖR FYSIK OCH ASTRONOMI
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige
www.chalmers.se

CHALMERS