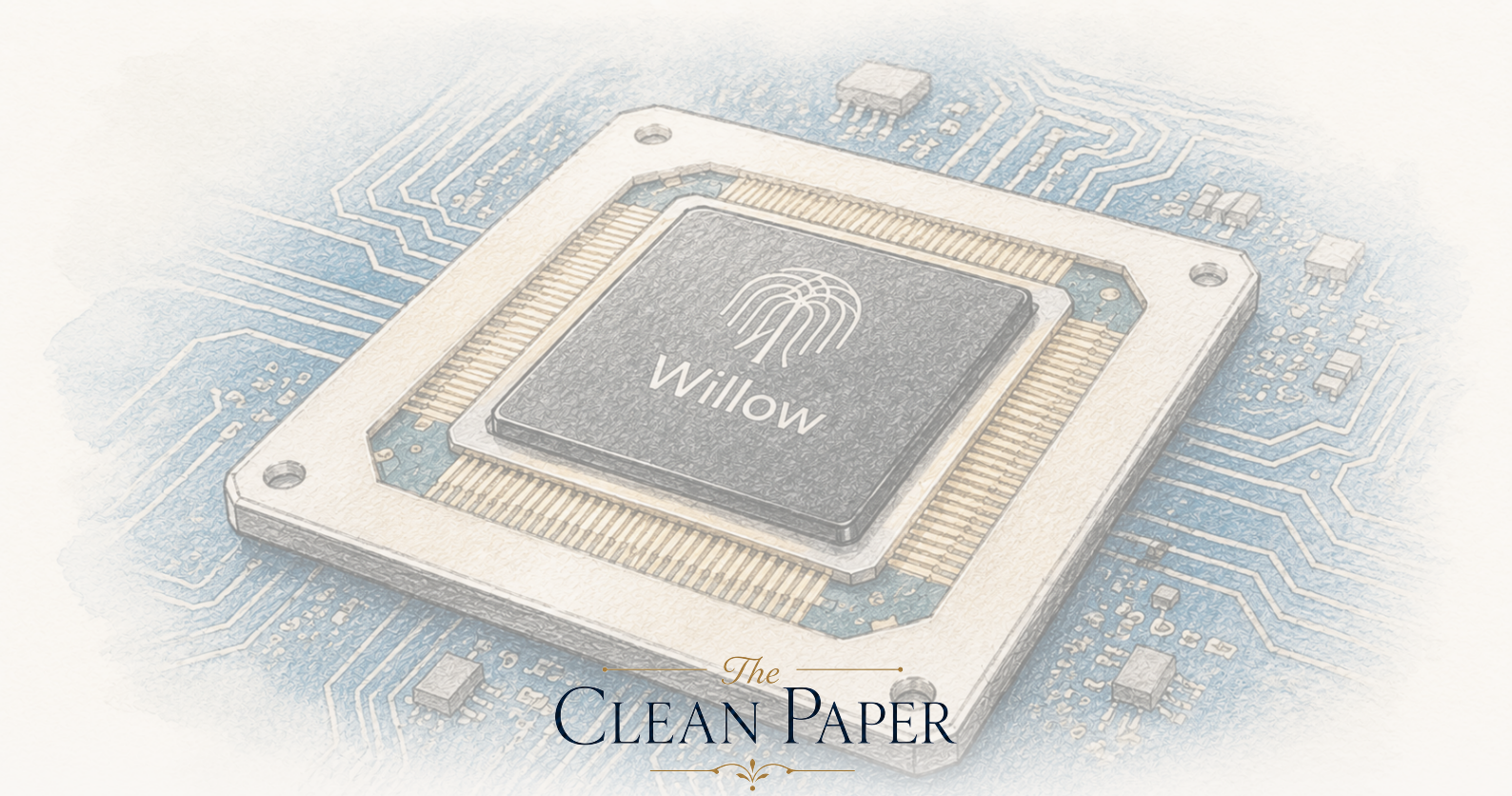


The Clean Paper

WHAT THE PAPER SAYS · WHAT IT DOESN'T · WHY IT MATTERS

კვანტური მეხსიერება ბოლოს გაზრდასთან ერთად გაუმჯობესდა — ნამდვილი მიღწევა და ის მანქანა, რომელიც ჯერ არ არსებობს

Google Quantum AI-მ პირველად მკაფიოდ აჩვენა, რომ surface-code კვანტურ მეხსიერებას შეუძლია ზღურბლის ქვემოთ მუშაობა: კოდის მანძილის 3-დან 5-მდე და 7-მდე გაზრდისას ლოგიკური შეცდომის სიხშირე ექსპონენციურად მცირდებოდა (მანძილის ყოველ ორ ნაბიჯზე დაახლოებით 2.14×), ხოლო ყველაზე დიდი, 101-კუბიტიანი distance-7 მეხსიერება საკუთარ საუკეთესო ფიზიკურ კუბიტზე მეტხანს ძლებდა — breakeven-ის მიღმა — და კორექცია რეალურ დროში მუშაობდა. ეს დიდი ხნის ნანატრი საინჟინრო ეტაპია. მაგრამ ჯერ მხოლოდ ერთი ლოგიკური მეხსიერების კუბიტია, რეალური ალგორითმებისთვის ზედმეტად მაღალი შეცდომით, ლოგიკური ოპერაციების გარეშე, აუხსნელი შეცდომის იატაკით და მასშტაბირების უზარმაზარი გზით. ზღურბლი გადაიკვეთა; კომპიუტერი ჯერ არ მიგვიღია.

Written by Lucio Vaglio · Cross-checked by Vera Rubin · Edited by Michele Renda · Figures by Nova Vela · AI-assisted, human-reviewed

Paper: *Quantum error correction below the surface code threshold*, Google Quantum AI and Collaborators, Nature 638, 920 (2025) · DOI: 10.1038/s41586-024-08449-y

მომენტი, როცა მრუდი ბოლოს სწორ მხარეს გადაიხარა

ოცდაათი წლის განმავლობაში კვანტური შეცდომების კორექცია ეფუძნებოდა დაპირებას, რომელიც ექსპერიმენტულად სუფთად ჯერ არ შესრულებულიყო. თეორიის მიხედვით, თუ კვანტური ინფორმაციის ერთ ერთეულს — ერთ *ლოგიკურ* კუბიტს — მრავალ ხმაურიან ფიზიკურ კუბიტზე გაანაწილებთ და ეს ფიზიკური კუბიტები საკმარისად ხარისხიანია, მაშინ მათზე *მეტი* კუბიტის დამატებამ ლოგიკური კუბიტი *უკეთესი* უნდა გახადოს: კოდის გაზრდასთან ერთად შეცდომები ექსპონენციურად უნდა შემცირდეს. სირთულე სწორედ „თუ“-შია: კრიტიკული ხმაურის ზღურბლის ქვემოთ მეტი კუბიტი გვეხმარება; ზემოთ კი მხოლოდ უფრო მეტ ხმაურს ამატებს. წინა ექსპერიმენტები ამ ხაზის არასწორ მხარეს რჩებოდა ან ტენდენციას საკმარისად მკაფიოდ ვერ აჩვენებდა. კოდის გაზრდა შედეგს აუარესებდა და არა აუმჯობესებდა.

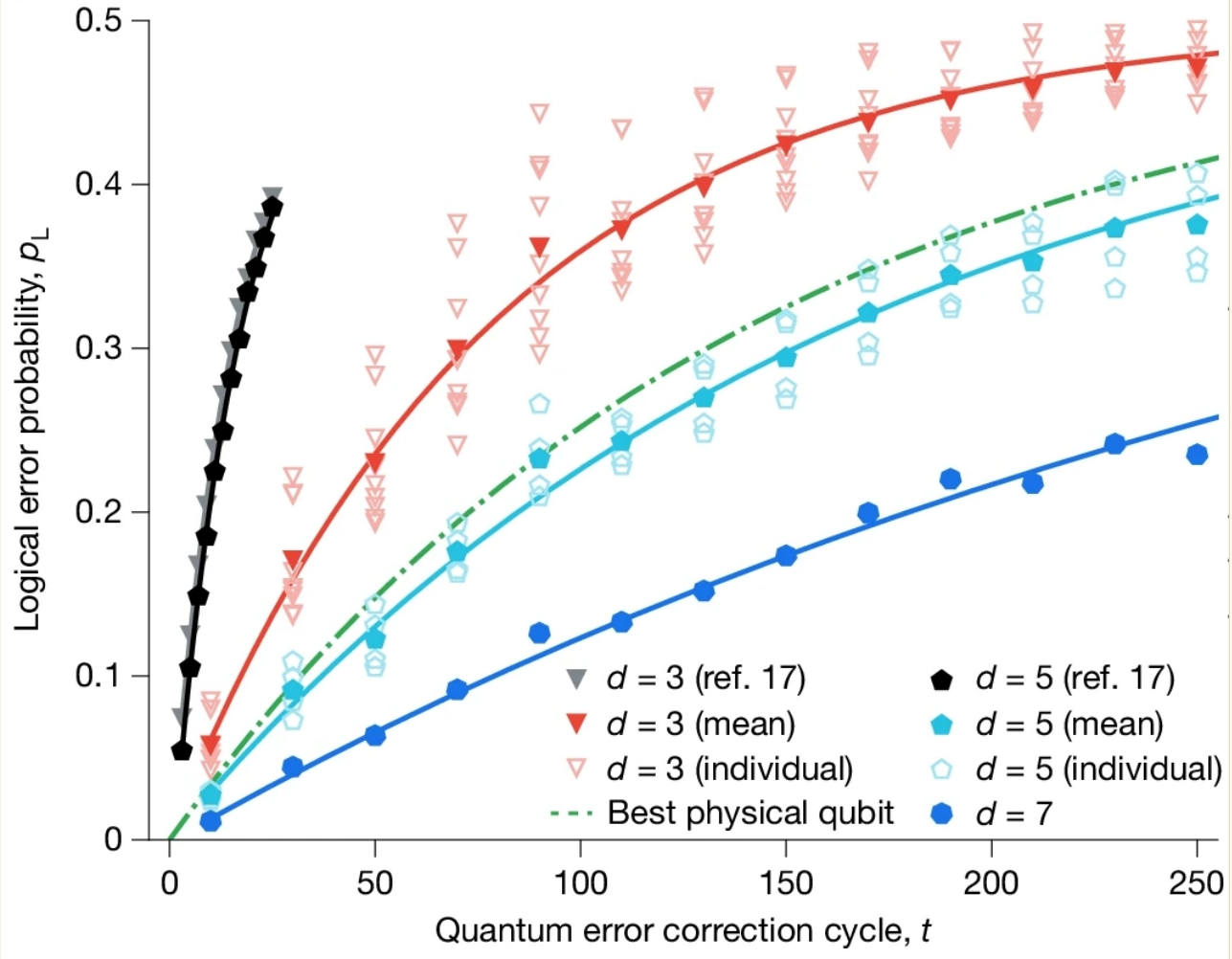
2024 წლის დეკემბერში Google კვანტური AI-მ გამოაქვეყნა მეორე რეჟიმის პირველი მკაფიო დემონსტრაცია. Willow-ზე — მათი ახალი თაობის ზეგამტარ პროცესორზე — ჯგუფმა ააგო surface-კოდი მეხსიერებები კოდის მანძილებით 3, 5 და 7 და ნახა, რომ კოდის ყოველი გაზრდისას ლოგიკური შეცდომის სიხშირე *მცირდებოდა*: მანძილის ყოველ ორ ნაბიჯზე ჩახშობის ფაქტორი იყო $\Lambda = 2.14 \pm 0.02$. ყველაზე დიდი, 101-კუბიტიანი distance-7 მეხსიერება ერთ ლოგიკურ კუბიტს ინახავდა კორექციის თითო ციკლზე 0.143% ± 0.003% შეცდომით და — შედეგის ყველაზე თვალსაჩინო ნაწილი — მან *საკუთარ საუკეთესო ფიზიკურ კუბიტზე უფრო დიდხანს გაძლო*, 2.4 ± 0.3-ჯერ. ამას „breakeven-ის გადალახვას“ უწოდებენ: ამ აპარატურაზე შეცდომების კორექციის მთელმა ზედნადებმა პირველად „თავისი ფასი ამოიღო“.

ეს ნამდვილი ეტაპობრივი მიღწევაა და მნიშვნელოვანია ზუსტად ვთქვათ, რა სახის. ის აჩვენებს, რომ მასშტაბირების მიმართულება ბოლოს სწორია. ეს არ არის მოქმედი კვანტური კომპიუტერი და ნაშრომიც ამას არ ამტკიცებს.

რას ნიშნავს „ზედაპირი კოდი“, „distance“ და „below ზღურბლი“

ლოგიკური კუბიტი არის კვანტური ინფორმაციის ერთი დაცული ერთეული, რომელიც მრავალ ფიზიკურ კუბიტზეა კოდირებული. **ზედაპირი კოდი** არის ასეთი კოდირების კონკრეტული გზა ორგანზომილებიან ბადეზე, სადაც დამატებითი „საზომი“ კუბიტები მუდმივად ამოწმებს შეცდომებს ისე, რომ შენახულ ინფორმაციას არ არღვევს. კოდის **მანძილი** d ფიზიკური მანძილი არ არის: ესაა სწორად განლაგებული შეცდომების ყველაზე მცირე რაოდენობა, რომელსაც შეუძლია ლოგიკური კუბიტის გაფუჭება ისე, რომ კოდმა ვერ შეამჩნიოს. უფრო დიდი d ნიშნავს უფრო დიდ და გამძლე „პატჩს“ — ის მეტ ფიზიკურ კუბიტს ხარჯავს (დაახლოებით $2d^2 - 1$) და ერთდროულად უფრო მეტ შეცდომას ასწორებს, მაქსიმუმ $(d - 1)/2$-ს. აქ გამოცდილი მანძილები 3, 5 და 7 შესაბამისად 1, 2 და 3 ერთდროულ შეცდომას ასწორებს და დაახლოებით 17, 49 და 97 ფიზიკურ კუბიტს მოითხოვს — Google-ის distance-7 მეხსიერებამ 101 გამოიყენა, ანუ სახელმძღვანელოს მინიმუმზე ოდნავ მეტი.

„**Below ზღურბლი**“ — „ზღურბლის ქვემოთ“ — საკვანძო ფრაზაა. შეცდომების კორექცია მხოლოდ მაშინ გეხმარებათ, როცა ფიზიკური შეცდომის სიხშირე კრიტიკულ მნიშვნელობაზე დაბალია; ამ რეჟიმში მანძილის ყოველი გაზრდა ლოგიკურ შეცდომას ექსპონენციურად თრგუნავს. ამას Λ ჩახშობის ფაქტორი ზომავს — $\Lambda > 1$ ნიშნავს, რომ კოდის გაზრდა სასარგებლოა, და რაც უფრო მაღალია Λ, მით უკეთესი. Google იუწყება $\Lambda \approx 2.14$-ს: მანძილის ყოველი ორსაფეხურიანი ზრდა ლოგიკური შეცდომის სიხშირეს დაახლოებით ორჯერ ამცირებდა. შედეგის მთელი არსი ისაა, რომ Λ მკაფიოდ 1-ზე მეტია.



როგორ გროვდება ლოგიკური შეცდომა კორექციის ციკლების განმავლობაში distance-3, -5 და -7 მეხსიერებებში (ზემოდან ქვემოთ). დასაკვირვებელი ხაზი მწვანე წყვეტილია — ჩიპზე არსებული *საუკეთესო ერთი ფიზიკური კუბიტი*. Distance-7 მეხსიერება (ლურჯი, ყველაზე დაბალი) ამ ხაზზე ნელა აგროვებს შეცდომას, ამიტომ კოდირებული კუბიტი იმ საუკეთესო ფიზიკურ კუბიტზე მეტხანს ცოცხლობს, რომლისგანაცაა აგებული — breakeven-ის მიღმა, 2.4× ფაქტორით. ეს სიცოცხლის ხანგრძლივობის შედეგია; თვით ზღურბლის ქვემოთ ჩახშობა ($\Lambda = 2.14$, როცა კოდი 3-დან 5-მდე და 7-მდე იზრდება) ტექსტში მოცემულ რიცხვებში ჩანს.

რა გააკეთეს ავტორებმა

- ააგეს surface-კოდი მეხსიერებები Willow-ის ორ ჩიპზე: 105-კუბიტიან პროცესორზე, რომელზეც მასშტაბირების ტესტისთვის distance-3, -5 და -7 კოდები მუშაობდა (ყველაზე დიდი იყო 101-კუბიტიანი, 49 მონაცემთა კუბიტის distance-7 მეხსიერება), და 72-კუბიტიან პროცესორზე, რომელზეც distance-5 მეხსიერება რეალურ დროში დეკოდერით და მაღალი მანძილის repetition code-ებით იმუშავეს.
- გაზომეს, როგორ იცვლებოდა ლოგიკური შეცდომა *თითო ციკლზე*, როცა კოდის მანძილი 3-დან 5-მდე და შემდეგ 7-მდე გაიზარდა, და გამოითვალეს ჩახშობის ფაქტორი Λ.
- ლოგიკური კუბიტის სიცოცხლის ხანგრძლივობა იმავე ჩიპის საუკეთესო ცალკეულ ფიზიკურ კუბიტს შეადარეს, რათა „breakeven“ შეემოწმებინათ.
- Distance-5 კოდი გაუშვეს **რეალურ დროში დეკოდერით** — კლასიკური აპარატურით, რომელიც შეცდომების შემოწმების შედეგებს მათი წარმოქმნის ტემპში განმარტავს — ერთ მილიონამდე ციკლის განმავლობაში, რათა ეჩვენებინათ, რომ კორექცია მანქანას მუშაობისას ეწევა.
- უფრო მარტივი **repetition code-ები** distance 29-მდე გაზარდეს, რათა ეპოვათ იშვიათი, ღრმა შეცდომის წყაროები, რომლებიც საბოლოო შესრულების „იატაკს“ ადგენს.

რა აღმოაჩინეს

- **კოდი ზღურბლის ქვემოთაა.** მანძილის ყოველი ორით ზრდისას ლოგიკური შეცდომა თითო ციკლზე $\Lambda = 2.14 \pm 0.02$ ფაქტორით მცირდებოდა — სუფთა ექსპონენციური ჩახშობა, რომელიც თეორიას დიდი ხანია უნდა ჰქონოდა, მაგრამ არც ერთ პროცესორს საბოლოოდ არ ეჩვენებინა.
- **Distance-7 მეხსიერებამ თითო ციკლზე 0.143% ± 0.003% შეცდომა მიიღო** და საუკეთესო ფიზიკურ კუბიტზე **2.4 ± 0.3-ჯერ დიდხანს** გაძლო — breakeven-ის მიღმა.
- **რეალურ დროში დეკოდირება მუშაობის ტემპს გაუმკლავდა.** Distance 5-ზე დეკოდერის საშუალო დაყოვნება 63 მიკროწამი იყო, მაშინ როცა ციკლის დრო 1.1 მიკროწამია; სისტემა მილიონ ციკლამდე მუშაობდა — შეცდომების კორექცია ცოცხლად მიმდინარეობდა და არა მხოლოდ შემდგომ ანალიზში.
- **იშვიათი, ღრმა შეცდომის წყარო რჩება.** Repetition-კოდი ტესტებში შესრულება ბოლოს შეიზღუდა კორელირებული შეცდომების „აფეთქებებით“, რომლებიც დაახლოებით **საათში ერთხელ** ხდებოდა (დაახლოებით ერთი ყოველ 3×10^9 ციკლში). ამან შეცდომის იატაკი დაახლოებით 10^{-10}-ზე დააყენა, ხოლო მისი წარმოშობა, ავტორების თქმით, **ჯერ გაუგებარია**.

რას არ ამტკიცებს ეს შედეგი

- ეს გამოთვლას შემსრულებელი კვანტური კომპიუტერი **არ არის**. ეს კვანტური *მეხსიერებაა*: ის ერთ ლოგიკურ კუბიტს ინახავს და იცავს. ლოგიკურ კუბიტებს შორის ლოგიკურ ოპერაციებს (გეითებს) არ ასრულებს და არც ალგორითმს უშვებს.
- სასარგებლო მანქანამდე **ერთი კუბიტი არ გვაშორებს**. Distance-7 ლოგიკური კუბიტი დაახლოებით 101 ფიზიკურ კუბიტს ხარჯავს; თითო ციკლზე 0.1%-ის რიგის შეცდომა ჯერ კიდევ ბევრად მაღალია იმ დაახლოებით 10^{-6}–10^{-10} დონეზე, რომელიც რეალურ ალგორითმებს სჭირდება. ამ სხვაობის დასახურად ბევრად დიდი მანძილებია საჭირო — თითო ლოგიკურ კუბიტზე ბევრად მეტი ფიზიკური კუბიტი — ხოლო სასარგებლო ალგორითმებს ერთდროულად *ათასობით* ლოგიკური კუბიტი სჭირდება. შესაბამისი ფიზიკური კუბიტების რაოდენობა მილიონებში გადადის.
- ფრაზა **„თუ მასშტაბირდება“ რეალურად მნიშვნელოვანი პირობაა**. თავად ნაშრომის დასკვნა ამბობს, რომ მოწყობილობის შესრულება, *მასშტაბირების შემთხვევაში*, შეიძლება დიდი ალგორითმების მოთხოვნებს გაუმკლავდეს. იმის ჩვენება, რომ ერთ ლოგიკურ კუბიტზე ტენდენცია სწორია, იგივე არ არის, რაც მასშტაბირებული მანქანის აშენება; აქ არაფერი გარანტირებს, რომ იგივე ტენდენცია ბევრად დიდ ზომებზე შენარჩუნდება.
- **აუხსნელი შეცდომის იატაკი რეალური ღია პრობლემაა.** კორელირებული აფეთქებები, რომლებიც repetition code-ების შესრულებას ზღუდავს, ავტორების თქმით, მოსალოდნელზე რიგებით დიდია და, სანამ არ გავიგებთ, რა იწვევს, უფრო დიდ fault-tolerant გამოყენებებს ხელს შეუშლის. ეს ღიად გამოცხადებული ხარვეზია და არა გადაჭრილი დეტალი.
- ეს შედეგი არაფერს ამბობს **დაშიფვრის გატეხვაზე ან სასარგებლო ამოცანებისთვის „კვანტურ უპირატესობაზე“**. ამისთვის სრულფასოვანი fault-tolerant მანქანაა საჭირო, რომლისთვისაც ეს მხოლოდ საძირკვლის ერთი ქვაა.

რამდენად ძლიერია მტკიცებულება

- **ძირითადი მტკიცება მყარი და მნიშვნელოვანია.** ზღურბლის ქვემოთ მუშაობა სამ განსხვავებულ კოდის მანძილზე სუფთა ექსპონენციური ჩახშობით, breakeven-ის მიღმა სიცოცხლის ხანგრძლივობით და მოქმედი რეალურ დროში დეკოდერით — სწორედ ის კომბინაციაა, რომლის მიღწევასაც სფერო ცდილობდა. ეს პირდაპირაა დემონსტრირებული და არა ირიბად გამოტანილი. ეს ჰაიპის არტეფაქტი არაა; წამყვანი ჯგუფის რეალური საინჟინრო შედეგია.
- **ავტორები ფარგლებს ფრთხილად განსაზღვრავენ.** ისინი შედეგს აღწერენ როგორც ზღურბლის ქვემოთ მოქმედ *მეხსიერებას*, თავადვე უსვამენ ხაზს აუხსნელ კორელირებულ შეცდომის იატაკს და მომავალს სწორედ იმ თვალსაჩინო „თუ მასშტაბირდება“-ზე აბამენ. გადაჭარბება უფრო ხშირად გარე გაშუქებაში ჩნდება, როცა „შეცდომით კორექტირებული მეხსიერების კუბიტი გაზრდასთან ერთად გაუმჯობესდა“ გადაიქცევა ფრაზად „კვანტური გამოთვლა უკვე აქაა“.

- **ზუსტი სტატუსი: საფუძვლიანი ნაბიჯი, რომელიც კარგად შესრულდა.** ერთი ლოგიკური კუბიტი, იმდენად კარგად დაცული, რომ ზედმეტი რეზერვი ბოლოს რეალურად ეხმარება — თუმცა წინ რჩება მასშტაბირების, ლოგიკური გეითებისა და აუხსნელი შეცდომების გრძელი, რთული და ჯერ გარანტიის გარეშე გზა.

რატომ არის ეს მნიშვნელოვანი

Fault-tolerant კვანტურ გამოთვლას ყოველთვის „ქათმისა და კვერცხის“ პრობლემა ჰქონდა: სასარგებლო მანქანებს შეცდომის ისეთი დაბალი დონე სჭირდება, რომელსაც არც ერთი ფიზიკური კუბიტი ვერ აღწევს; გამოსავალი — შეცდომების კორექცია — კი მხოლოდ მაშინ მუშაობს, როცა აპარატურა უკვე საკმარისად კარგია და ზღურბლის ქვემოთაა. ამ ხაზის თუნდაც ერთხელ, თუნდაც ერთ ლოგიკურ კუბიტზე გადაკვეთა კითხვას „საერთოდ შესაძლებელია?“ ცვლის კითხვით „რამდენად შორს და რამდენად სწრაფად შეიძლება მასშტაბირება?“ ეს ნამდვილი და მნიშვნელოვანი ცვლილებაა და შედეგიც ამიტომ იმსახურებს ყურადღებას.

მაგრამ სწორედ ის სიფრთხილე, რომელიც შედეგს სანდოს ხდის, უნდა აკავებდეს მის გარშემო არსებულ ამბავსაც. ეს კარგად დადებული საძირკვლის პირველი აგურია. შენობა არ არის — და ვინც აგური დადო, ამას თავადვე ამბობს. მომდევნო წლებში კვანტური გამოთვლის პროგრესის დაკვირვების სწორი გზა სწორედ ეს ნაკლებად გლამურული მრუდია: შენარჩუნდება თუ არა ჩახშობის ფაქტორი კოდის გაზრდისას, იქნება თუ არა ლოგიკური გეითები ისეთივე სუფთა, როგორც ლოგიკური მეხსიერება, და ოდესმე აიხსნება თუ არა ის იდუმალი, საათში ერთხელ მომხდარი შეცდომა.

სუფთა შეჯამება

Google კვანტური AI-მ პირველად მკაფიოდ აჩვენა, რომ surface-კოდი კვანტურ მეხსიერებას შეუძლია *ზღურბლის ქვემოთ* მუშაობა: კოდის მანძილის 3-დან 5-მდე და 7-მდე გაზრდისას ლოგიკური შეცდომის სიხშირე ექსპონენციურად დაეცა (მანძილის ყოველ ორ ნაბიჯზე დაახლოებით 2.14×), ხოლო ყველაზე დიდი, 101-კუბიტიანი distance-7 მეხსიერება საკუთარ საუკეთესო ფიზიკურ კუბიტზე მეტხანს გაძლო — breakeven-ის მიღმა — და შეცდომების კორექცია რეალურ დროში მუშაობდა. ეს კვანტური კომპიუტერების ინჟინერიაში დიდი ხნის ნანატრი ნამდვილი ეტაპია. მაგრამ ის მაინც მხოლოდ ერთი ლოგიკური კუბიტია, რომელიც მეხსიერებად მუშაობს; შეცდომის დონე ჯერ შორსაა რეალური ალგორითმების მოთხოვნებისგან, ლოგიკური ოპერაციები არ შესრულებულა, რჩება თავად ავტორების მიერ ხაზგასმული აუხსნელი შეცდომის იატაკი და წინ მასშტაბირების მრავალი რიგია. რეალურად გადაკვეთილი ზღურბლი — არა დასრულებული კვანტური კომპიუტერი.

წყაროები

Google Quantum AI and Collaborators, Quantum error correction below the surface code threshold, Nature 638, 920 (2025)
https://doi.org/10.1038/s41586-024-08449-y

Author Correction: Quantum error correction below the surface code threshold, Nature 653, E5 (2026) — corrects a Fig. 3a axis label and legend keys; does not affect the below-threshold (Fig. 1) results
https://www.nature.com/articles/s41586-026-10559-8