1QBit

Redefine Intractable

TopQAD™ Tutorial

IEEE Quantum Week 2025

Evaluate and Design Quantum Computers: Automated FTQC Architecture Design and Resource Estimation Using TopQAD

Opening remarks

Tamiko Masuda (*2 min*)

Business Operations Lead, 1QBit

Features and Releases

Topological Quantum Architecture Design (TopQAD) Software Suite

Beta access – special offer for QCE25 TUT21 attendees

- Free portal and SDK access (time limited)
- Noise Profiler, Compiler, Quantum Resource Estimation services
- Unlimited jobs, one job at a time
- One device

TopQAD™

September 2025

Today's Tutorial

Account activation

Portal interaction

SDK interaction

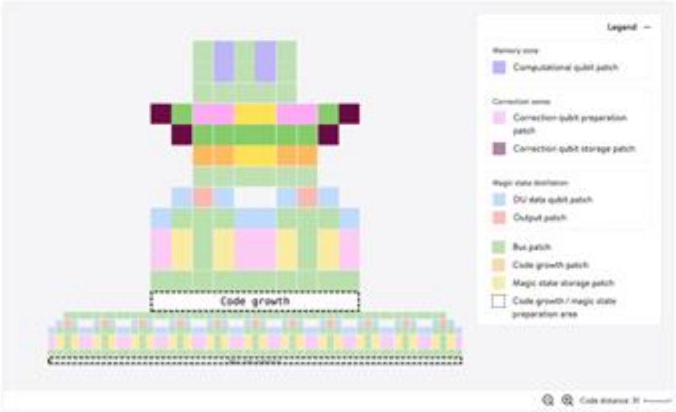
October 2025

Beta Update

Circuit file upload

QRE Lite

Architecture visualizer



Q1 2026

Commercial Launch

Paid product

Multiple jobs at a time

Multiple devices

1QBit

Agenda

Talk Setup Interaction

Session 1

- Resource estimators of today, operating systems of tomorrow (Pooya)
- TopQAD account creation and activation (Katie)
- Fault-tolerant compilation (Zak)
- Design and optimization of an FTQC architecture (Allyson)
- Interaction with the TopQAD portal (Allyson, Katie)

Session 2

- Session 1 summary (Allyson, Katie)
- FTQC protocol performance (Abdullah)
- FTQC emulation using the TopQAD SDK (Abdullah, Katie)
- Quantum resource estimation using the TopQAD SDK (Allyson, Katie)






Before We Get Started

- Agenda
- Presentation handout
- TopQAD hands-on interaction
  - Download an MFA app (e.g. Google Authenticator)
  - TopQAD account, verification, terms & policies
  - Emails about upcoming releases
  - Installation
  - Documentation
- Partnership and collaboration enquiries beyond beta

https://1qbit.com/qce25-tutorial/

topqad@1qbit.com
tamiko.masuda@1qbit.com

Resource estimators of today, operating systems of tomorrow

Pooya Ronagh (*20 min*)

CTO, 1QBit

1QBit

Noise in Quantum Computers

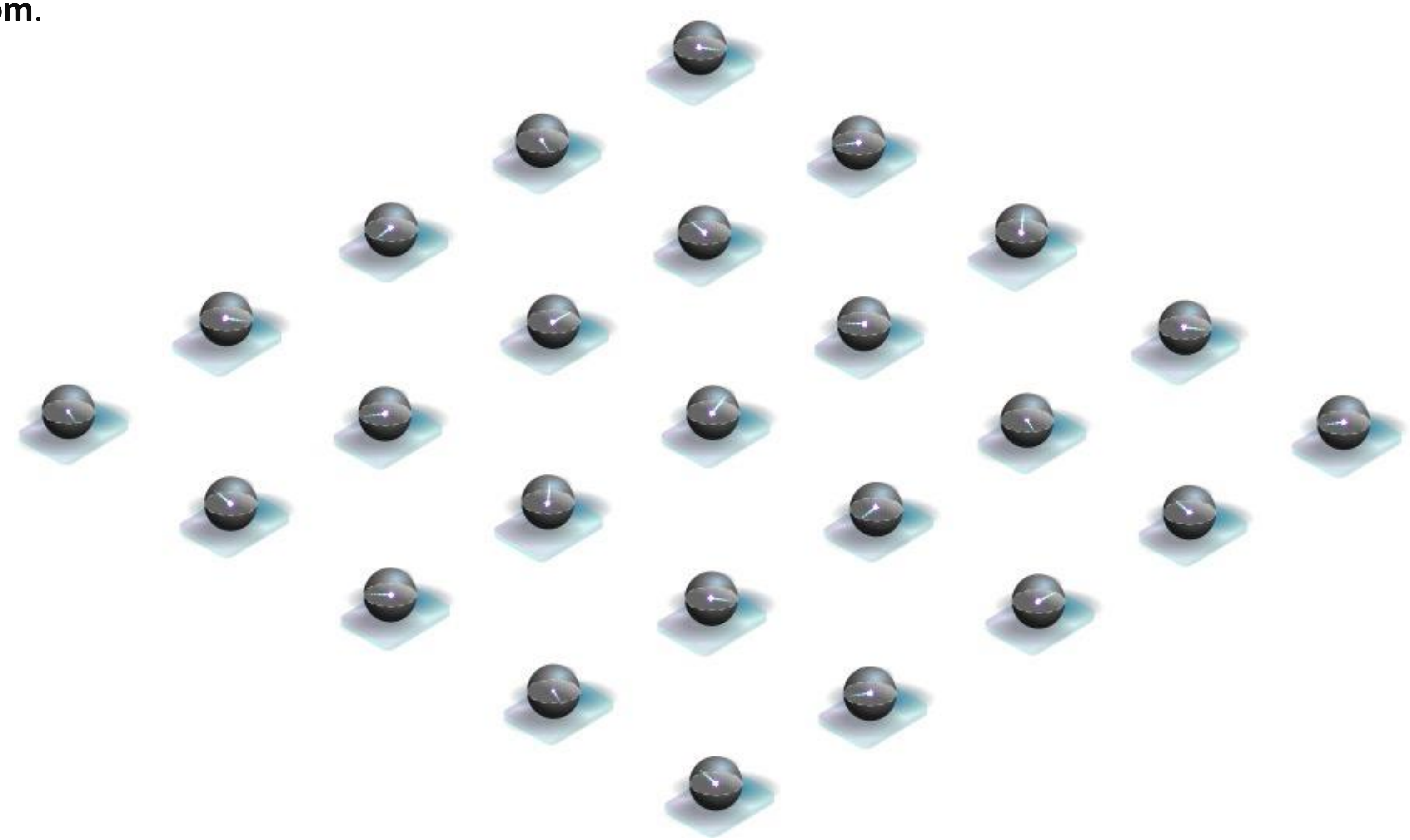
- Qubits are (always going to be) **fragile and erroneous**.
- Utility-scale quantum computation requires running circuits that may take **hours, days, or even weeks** to execute.
- Physical qubits ⇒ quantum error correcting **(QEC) codes**
- **Logical gates** between QEC codes do not look like the **physical gates** of the QPU and are themselves long circuits.
- QEC codes ⇒ fault-tolerant quantum computation **(FTQC)**
- Designing a *fault-tolerant quantum architecture* requires having **detailed knowledge about hardware noise** at compile time.
- This makes it difficult to know the **exact size and wall-clock time** of the computer for an application.

Our goal: To commoditize the assessment of QC utility, by automating QEC and hiding away its details.

1QBit

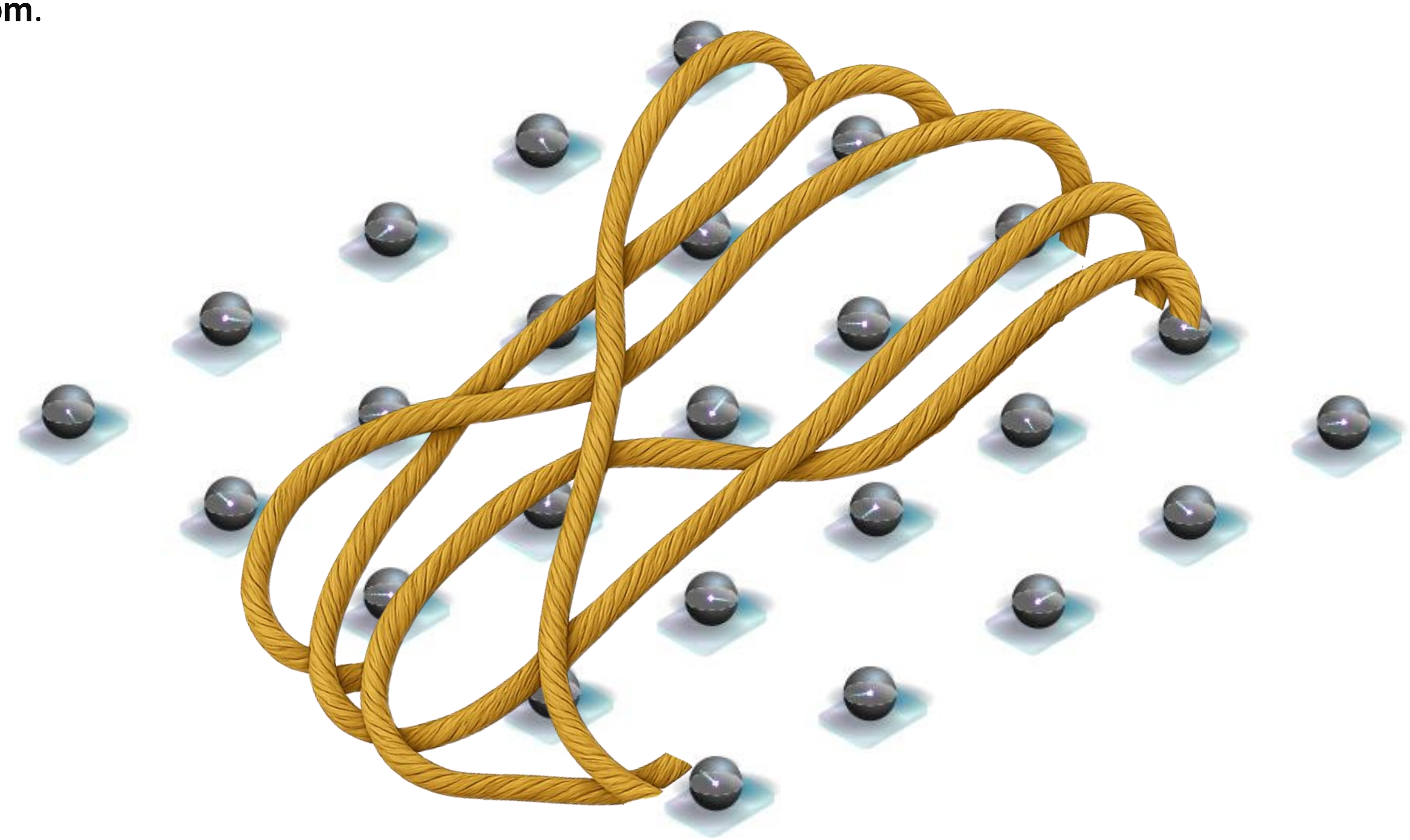
Topological QEC Codes

Arrays of physical qubits are used to store quantum information in their **topological degrees of freedom**.



Topological QEC Codes

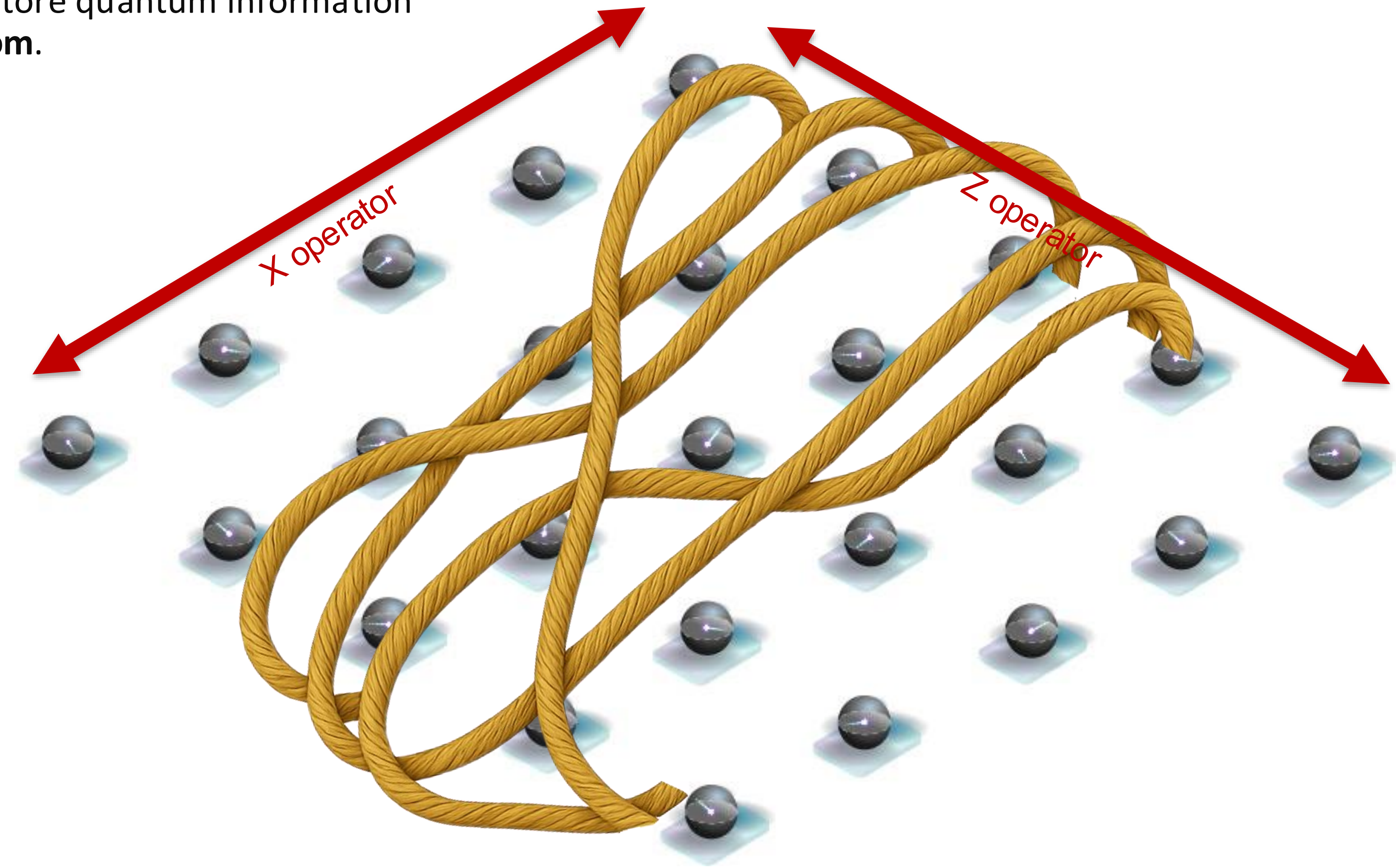
Arrays of physical qubits are used to store quantum information in their **topological degrees of freedom**.



1QBit

Topological QEC Codes

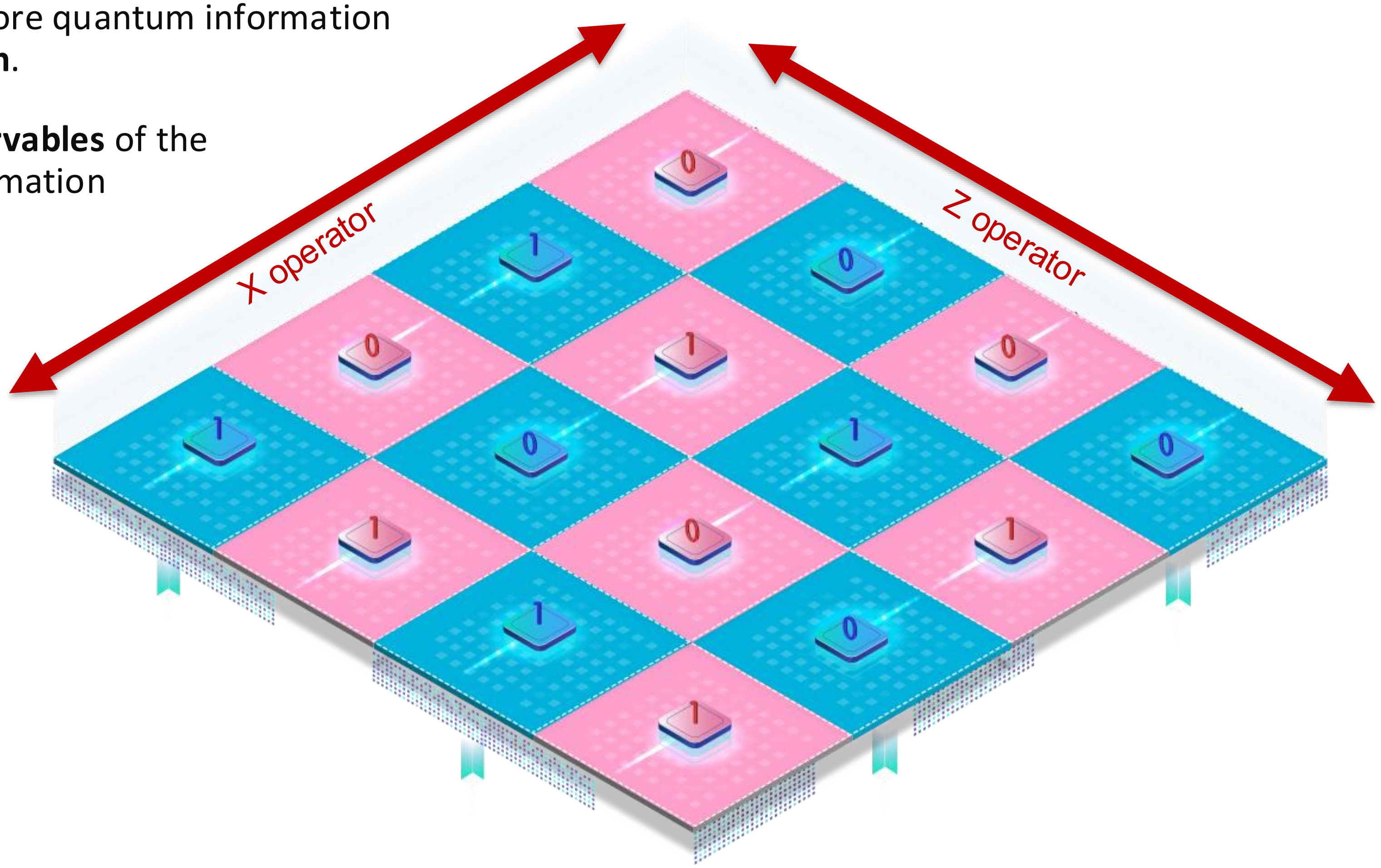
Arrays of physical qubits are used to store quantum information in their **topological degrees of freedom**.



Topological QEC Codes

Arrays of physical qubits are used to store quantum information in their **topological degrees of freedom**.

Quantum measurements of **local observables** of the topological state provide classical information that probe the code for broken order.



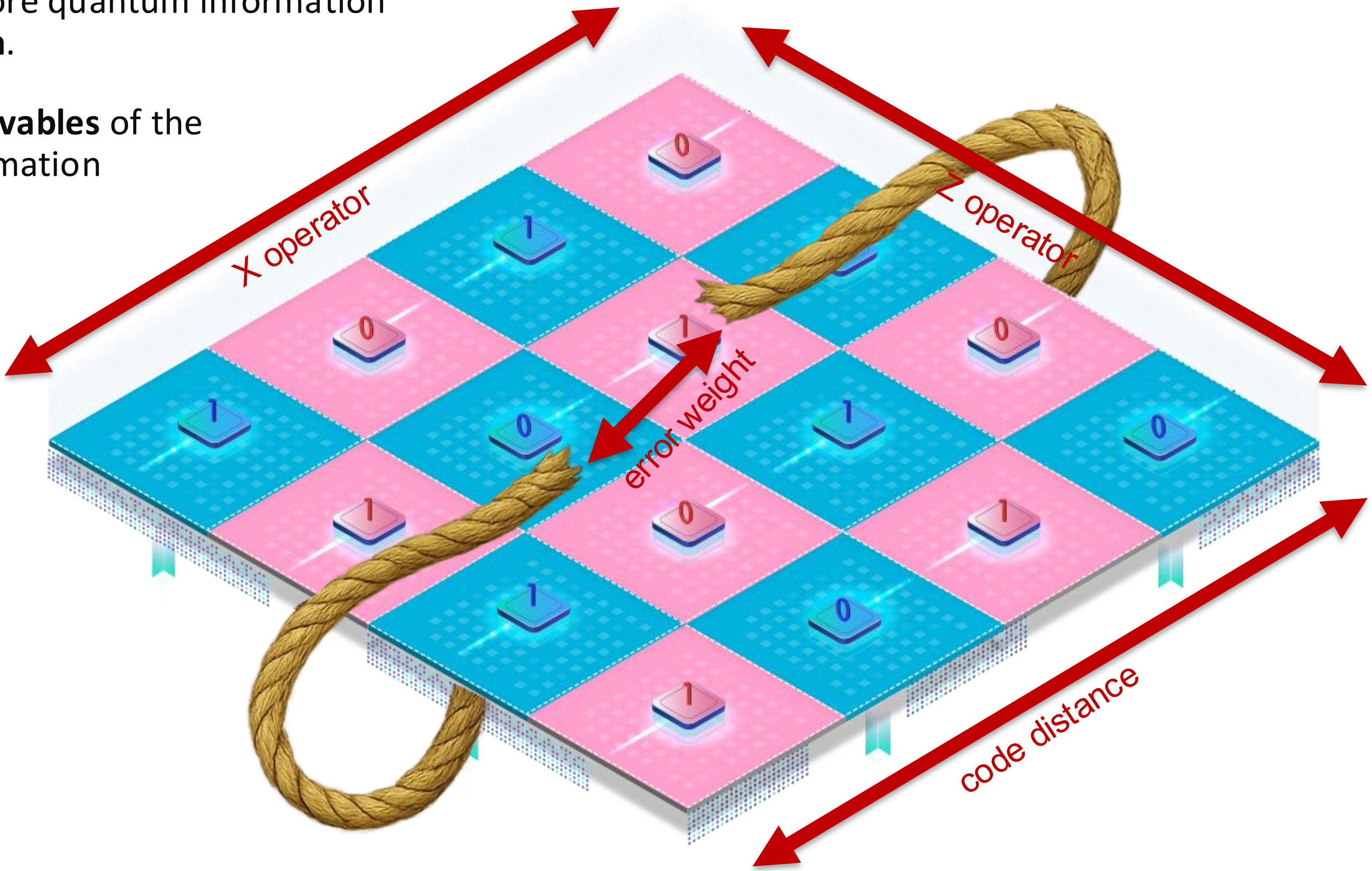
1QBit

Topological QEC Codes

Arrays of physical qubits are used to store quantum information in their **topological degrees of freedom**.

Quantum measurements of **local observables** of the topological state provide classical information that probe the code for broken order.

More physical qubits ⇒ larger codes distance ⇒ more protection



1QBit

Topological QEC Codes

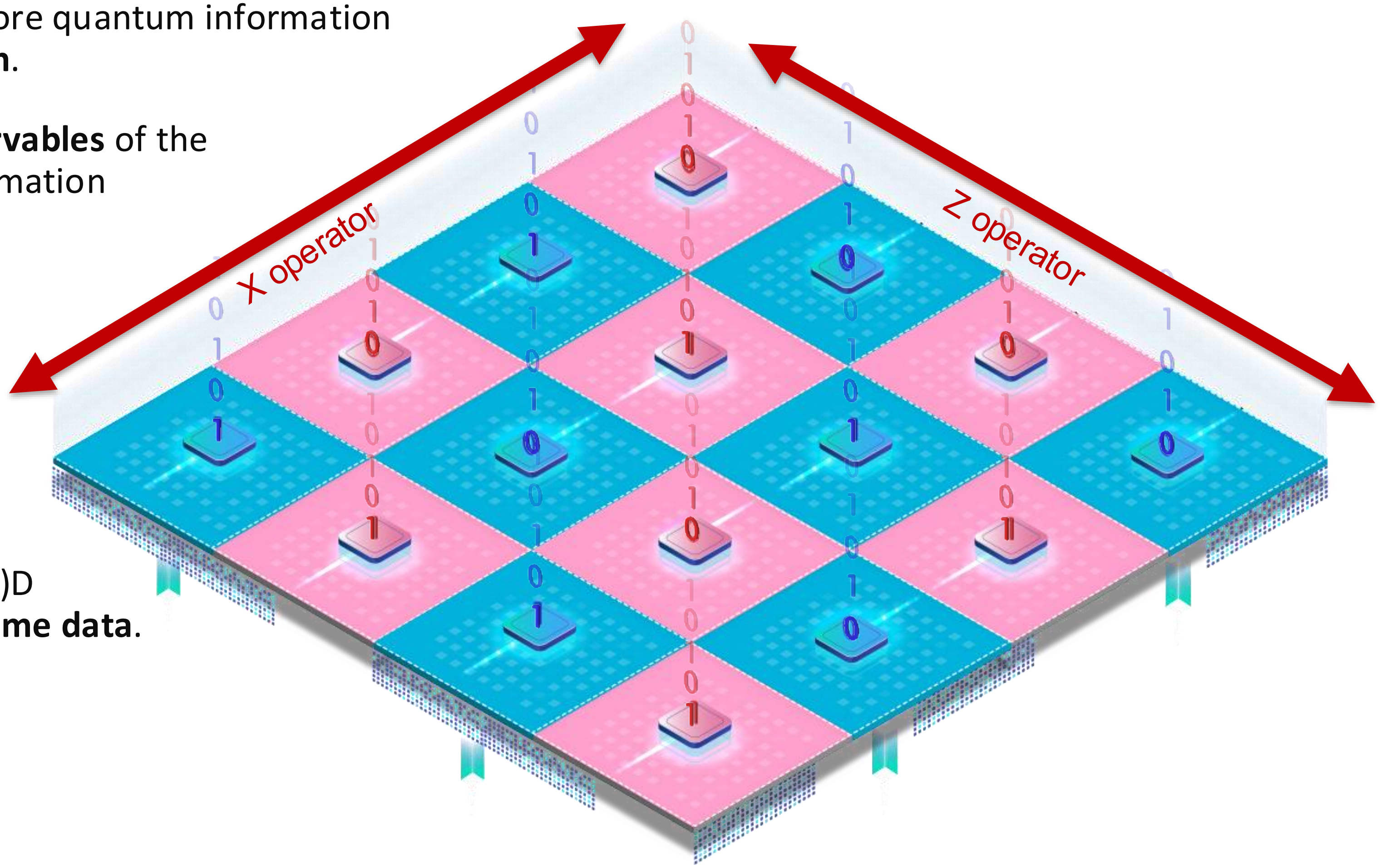
Arrays of physical qubits are used to store quantum information in their **topological degrees of freedom**.

Quantum measurements of **local observables** of the topological state provide classical information that probe the code for broken order.

More physical qubits ⇒ larger codes distance ⇒ more protection

Physical gates of the computer form **stabilization (parity check)** cycles for local observable readout.

Repeated measurements produce (2+1)D arrays of classical bits known as **syndrome data**.



1QBit

Topological QEC Codes

Arrays of physical qubits are used to store quantum information in their **topological degrees of freedom**.

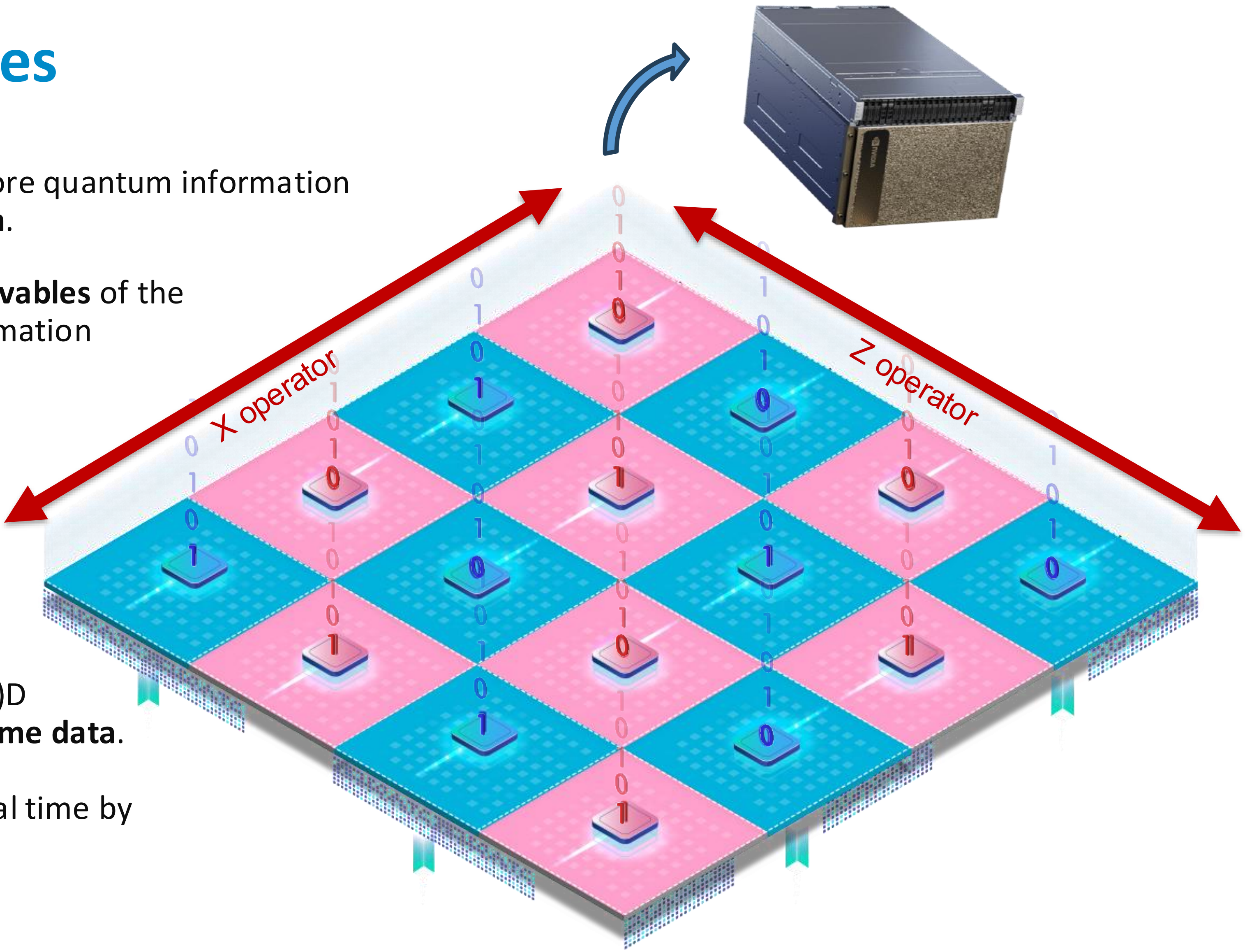
Quantum measurements of **local observables** of the topological state provide classical information that probe the code for broken order.

More physical qubits ⇒ larger codes distance ⇒ more protection

Physical gates of the computer form **stabilization (parity check)** cycles for local observable readout.

Repeated measurements produce (2+1)D arrays of classical bits known as **syndrome data**.

The syndrome data are processed in real time by classical processors called **decoders**.



1QBit

Topological QEC Codes

Arrays of physical qubits are used to store quantum information in their **topological degrees of freedom**.

Quantum measurements of **local observables** of the topological state provide classical information that probe the code for broken order.

More physical qubits ⇒ larger codes distance ⇒ more protection

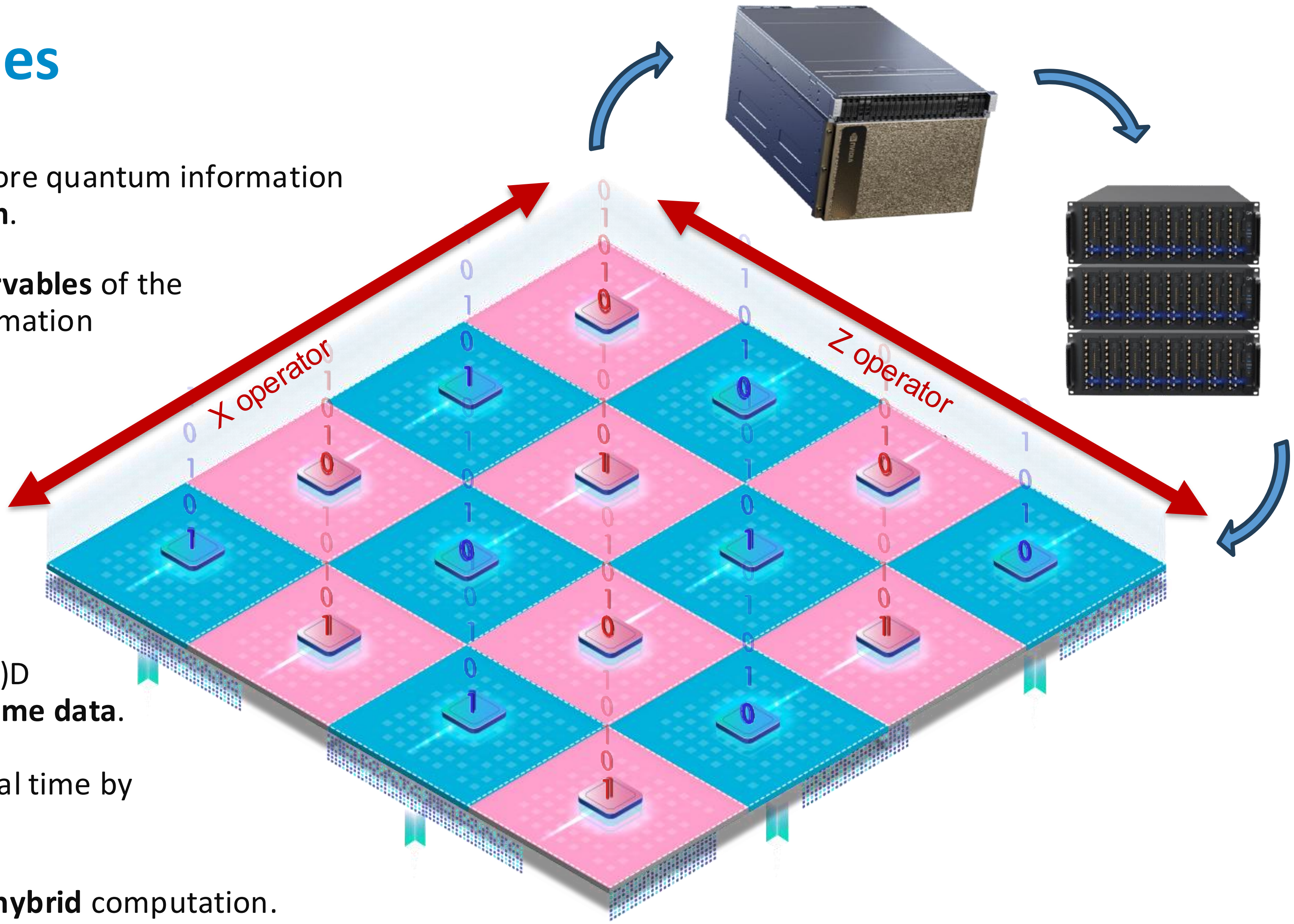
Physical gates of the computer form **stabilization (parity check)** cycles for local observable readout.

Repeated measurements produce (2+1)D arrays of classical bits known as **syndrome data**.

The syndrome data are processed in real time by classical processors called **decoders**.

FTQC is naturally a **quantum–classical hybrid** computation.

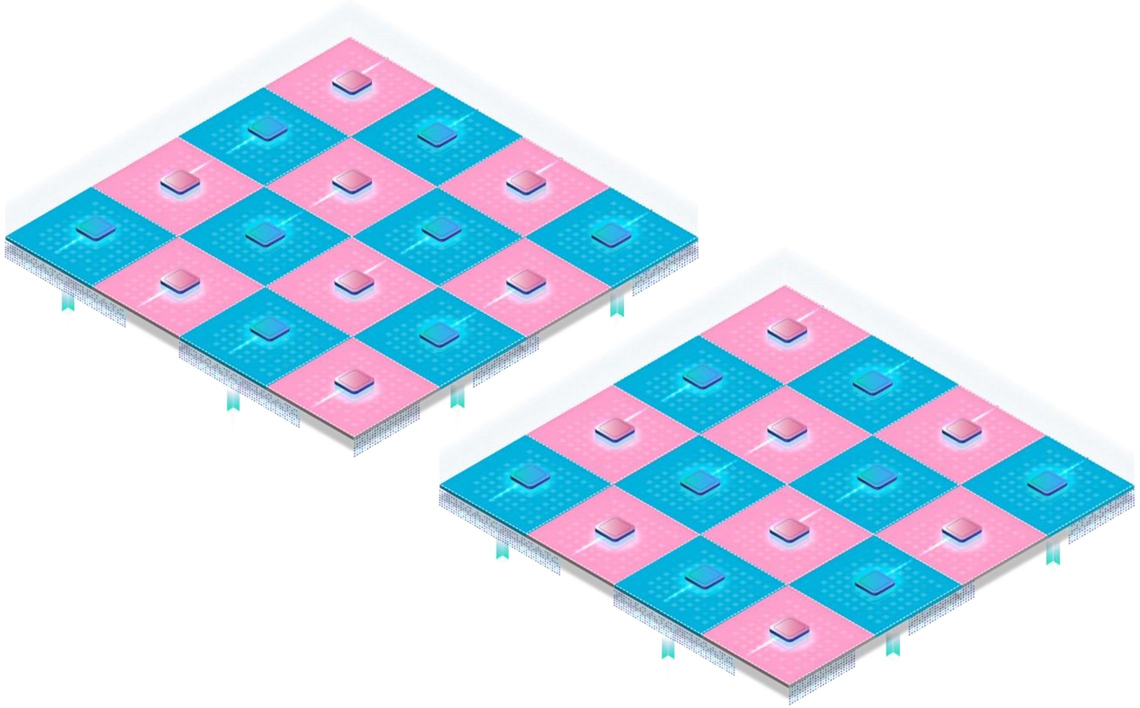
The **reaction time** (read out + decode + control) is a fundamental speed limit for it.



1QBit

Fault-Tolerant Architecture

A logical variant of physical entangling gates is required to perform FTQC via error correcting codes.

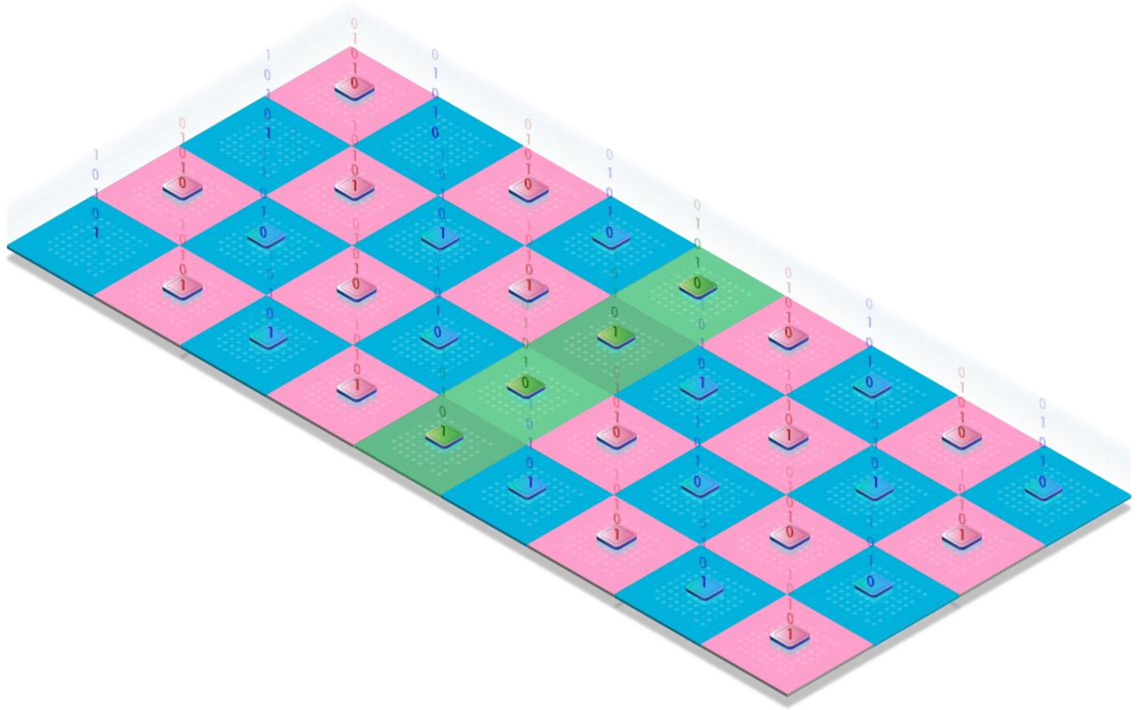


Fault-Tolerant Architecture

A logical variant of physical entangling gates is required to perform FTQC via error correcting codes.

We achieve such entanglement using **lattice surgery** between the topological patches.

Fault tolerance demands decoding the syndrome data generated across the entire layout.



1QBit

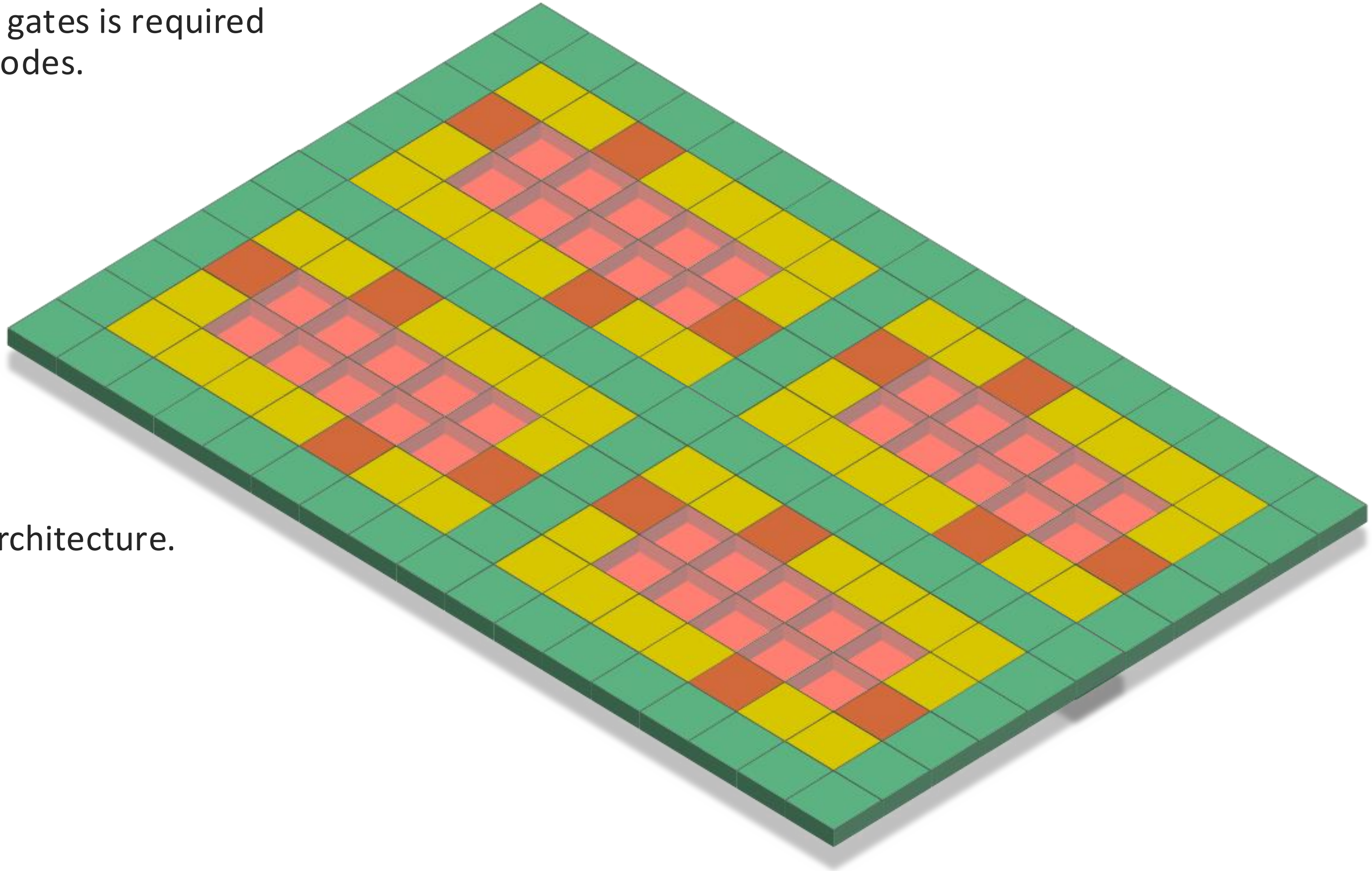
Fault-Tolerant Architecture

A logical variant of physical entangling gates is required to perform FTQC via error correcting codes.

We achieve such entanglement using **lattice surgery** between the topological patches.

Fault tolerance demands decoding the syndrome data generated across the entire layout.

We use a **quantum bus** to produce long-range entanglement across the architecture.



1QBit

Fault-Tolerant Architecture

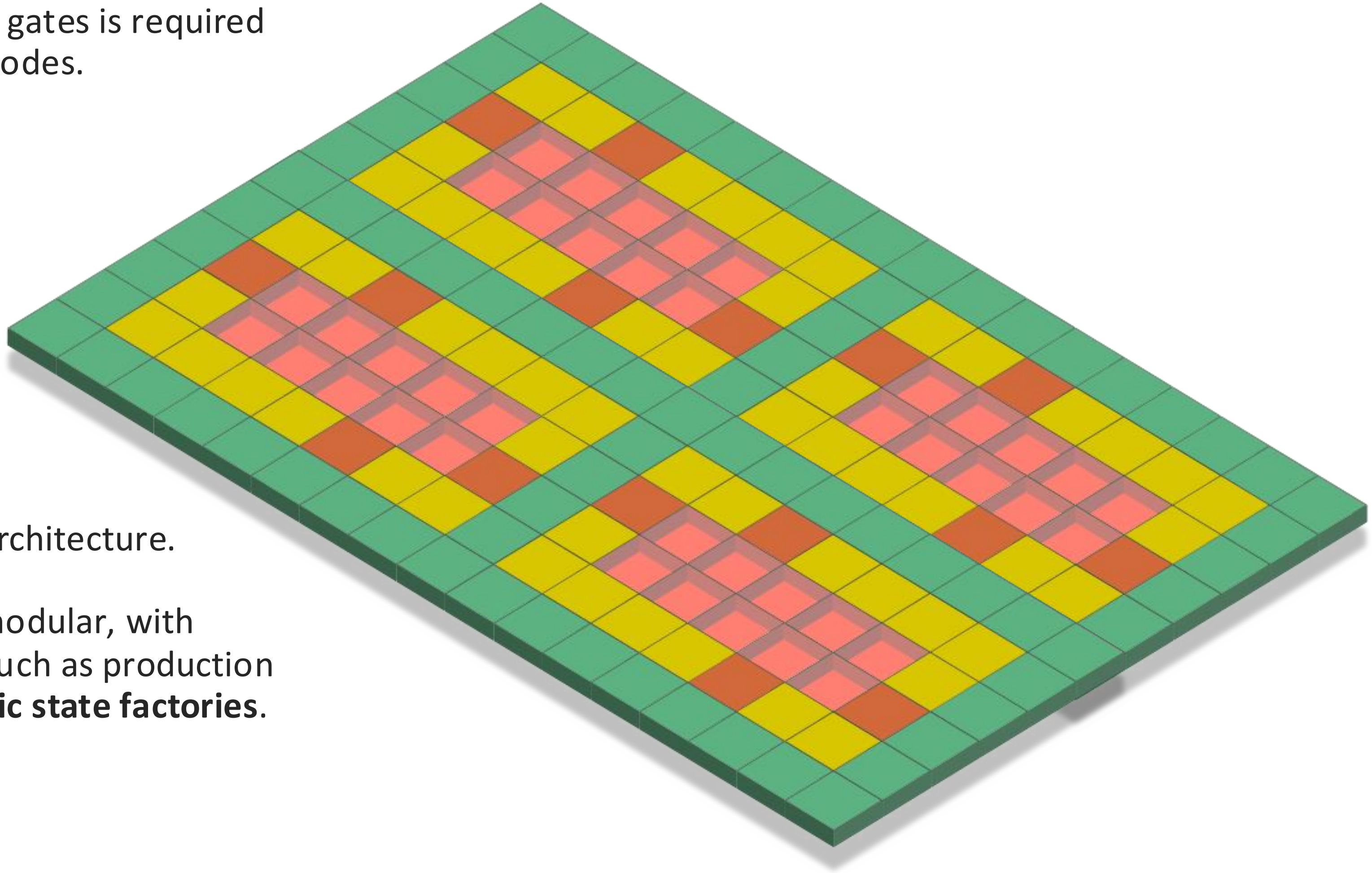
A logical variant of physical entangling gates is required to perform FTQC via error correcting codes.

We achieve such entanglement using **lattice surgery** between the topological patches.

Fault tolerance demands decoding the syndrome data generated across the entire layout.

We use a **quantum bus** to produce long-range entanglement across the architecture.

Our **fault-tolerant architectures** are modular, with designated zones for different tasks, such as production and distillation of **magic states** in **magic state factories**.

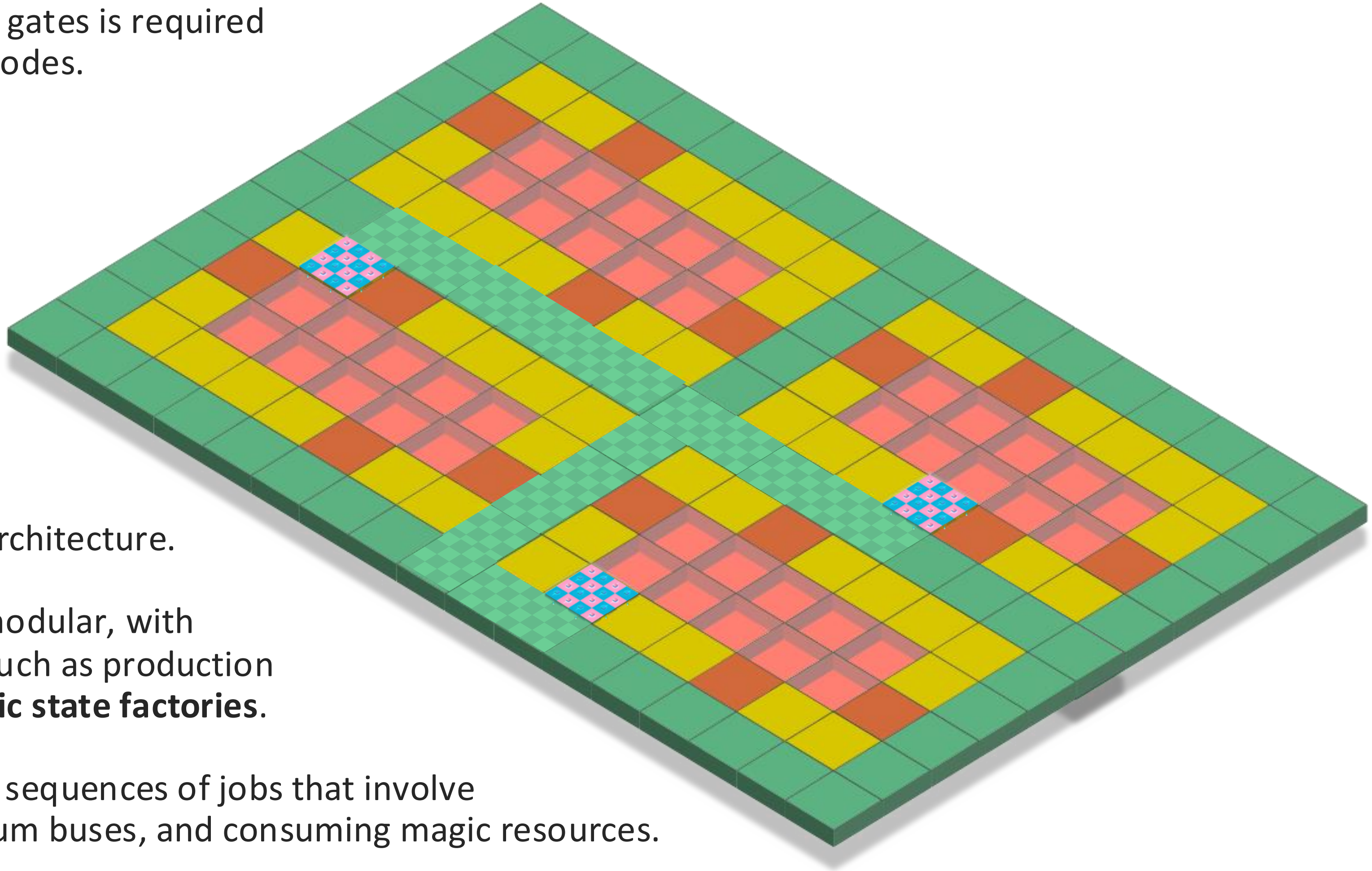


1QBit

Fault-Tolerant Architecture

A logical variant of physical entangling gates is required to perform FTQC via error correcting codes.

We achieve such entanglement using **lattice surgery** between the topological patches.



Fault tolerance demands decoding the syndrome data generated across the entire layout.

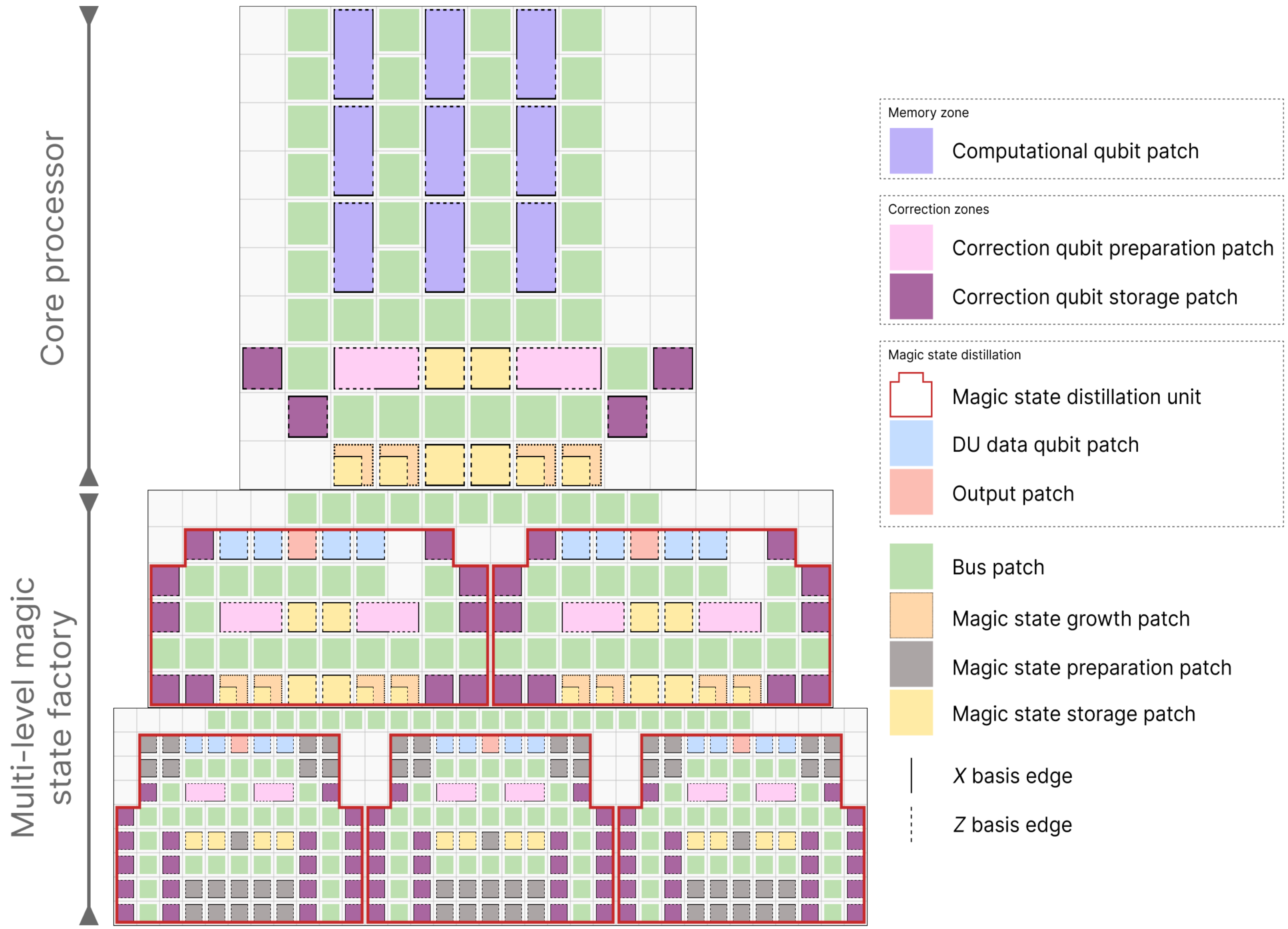
We use a **quantum bus** to produce long-range entanglement across the architecture.

Our **fault-tolerant architectures** are modular, with designated zones for different tasks, such as production and distillation of **magic states** in **magic state factories**.

Our **fault-tolerant compilers** optimize sequences of jobs that involve operating on data qubits, using quantum buses, and consuming magic resources.

1QBit

Architecture Design as a Supply Chain Problem



1QBit

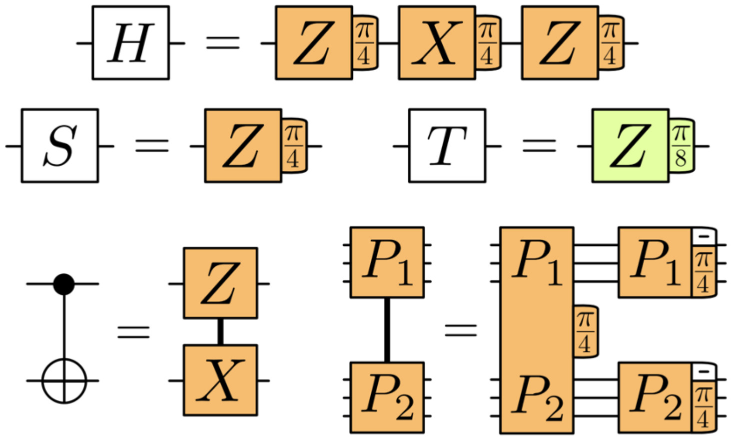
Architecture Design as a Supply Chain Problem

Our methodology:

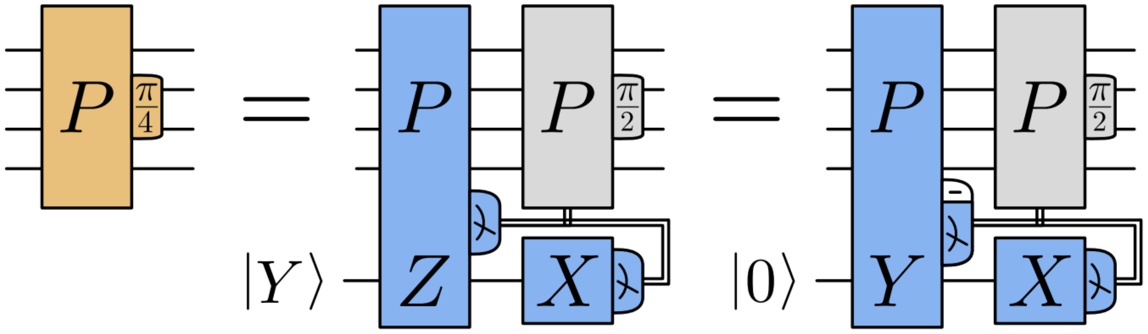
- Break down the logical microarchitecture into interconnected modules
- Each module consumes/produces input/output resource states with certain rates
- Architecture is optimal when the rates are balanced
- Intentional undersupply or oversupply:
  - Space–time trade-offs for utility-scale circuits
  - Analyzing decoder's impact
  - Optimizing the architecture and compilation route using decoder's performance info

1QBit

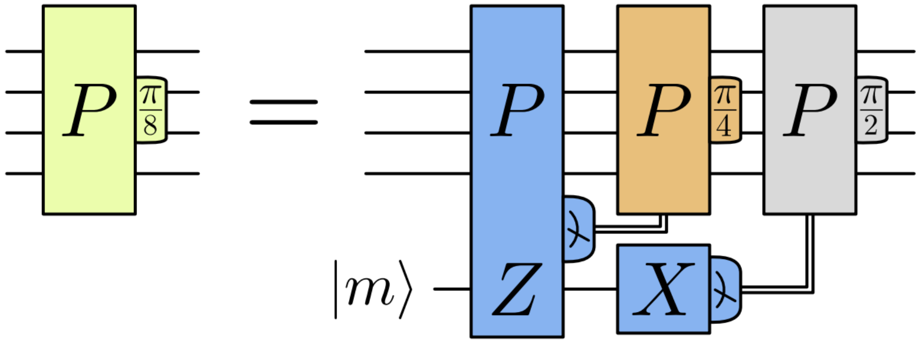
Instruction Set Architectures (ISA)



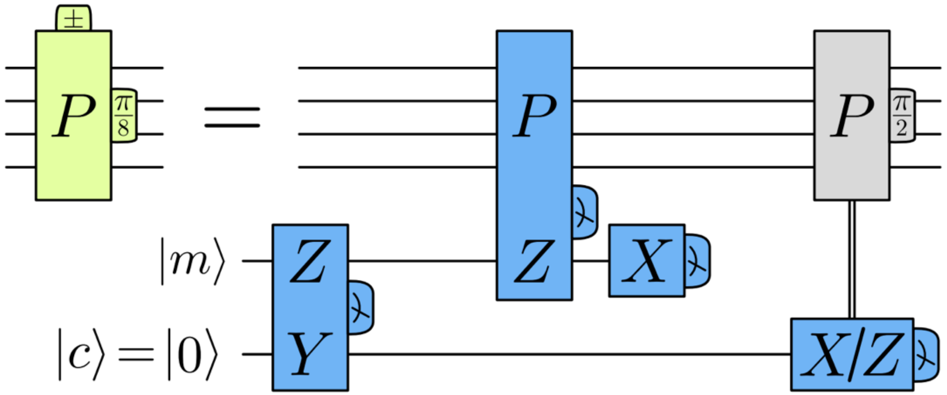
Pauli-product rotations ISA



π/4 rotations by Y surgeries



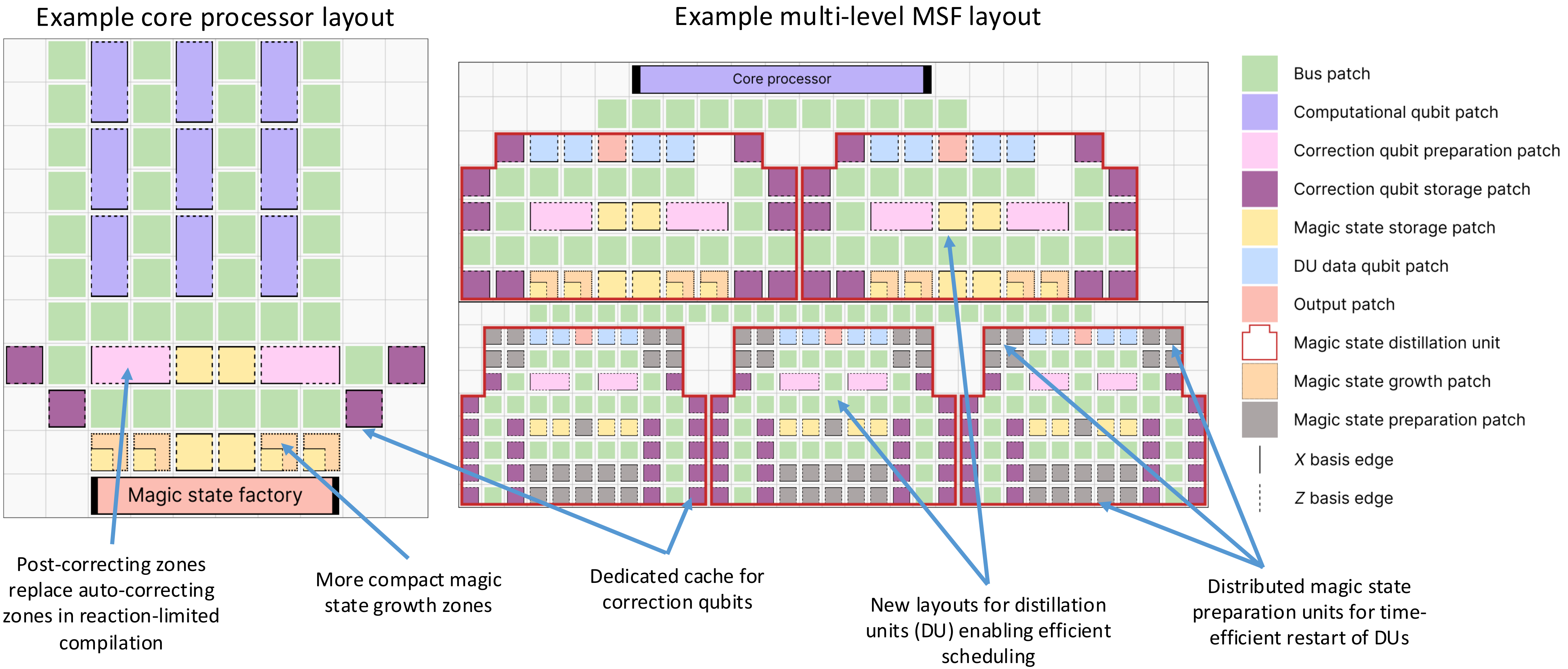
π/8 rotations by consuming magic states



Post-corrected π/8 rotations using an ancilla

Litinski, D., Quantum, 3, (2019):128.

More Complex ISAs, More Complex Microarchitectures



Silva, A. et al., arXiv:2411.04270 (2024).

1QBit

Fault-Tolerant Compilation

Depends on the logical microarchitecture (layout) of the core processors (but not their noise models).

Main components:

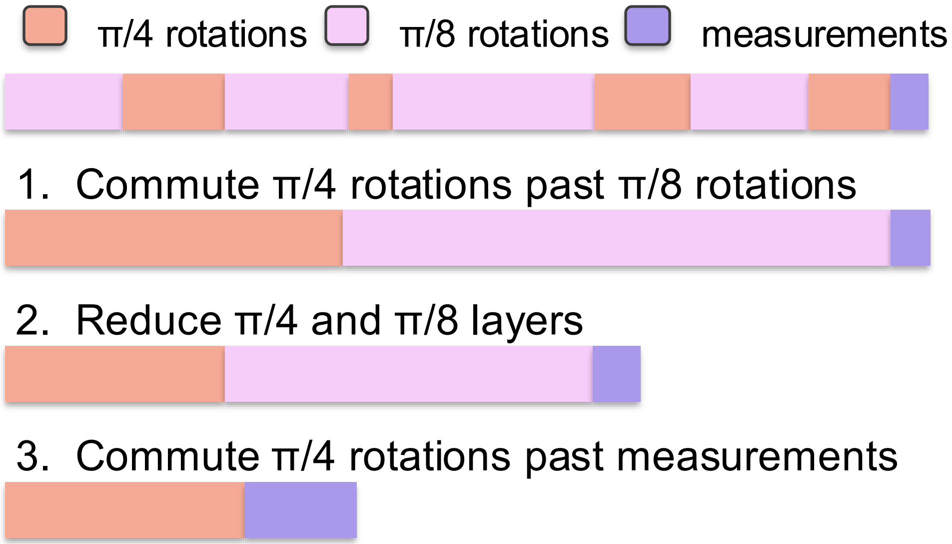
- Decomposer: CNOT + Analog rotations ⇒ ISA

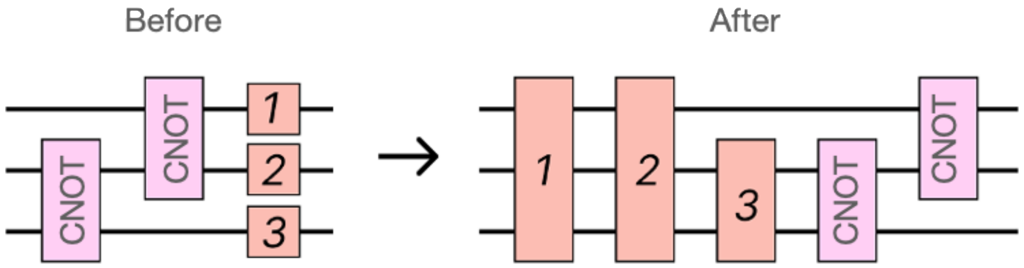
Fault-Tolerant Compilation

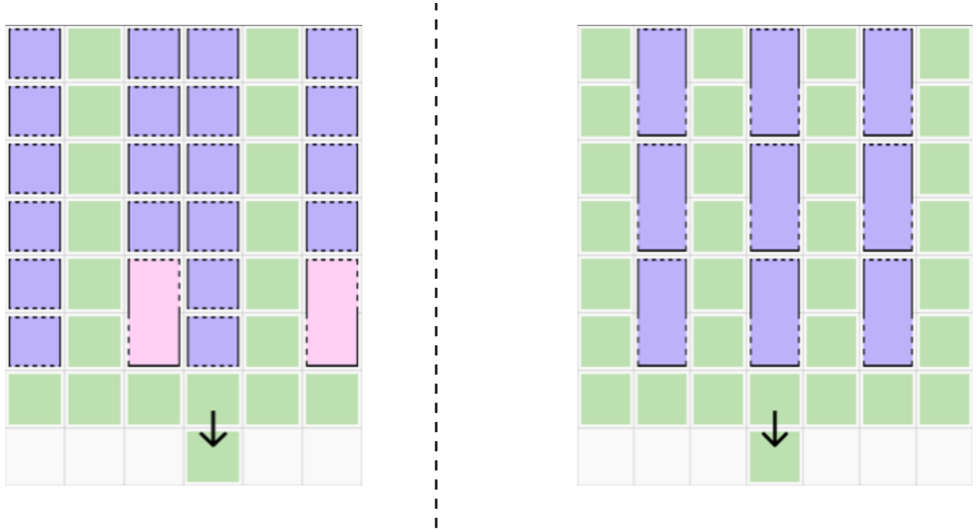
Depends on the logical microarchitecture (layout) of the core processors (but not their noise models).

Main components:

- Decomposer: CNOT + Analog rotations ⇒ ISA
- Optimizer







Compact layout

Fast layout

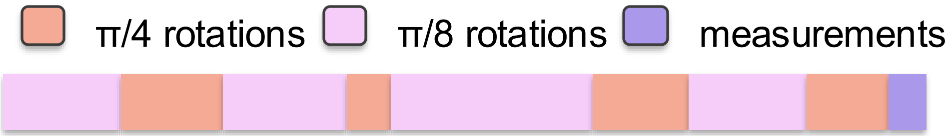
1QBit

Fault-Tolerant Compilation

Depends on the logical microarchitecture (layout) of the core processors (but not their noise models).

Main components:

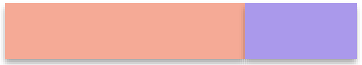
- Decomposer: CNOT + Analog rotations ⇒ ISA
- Optimizer



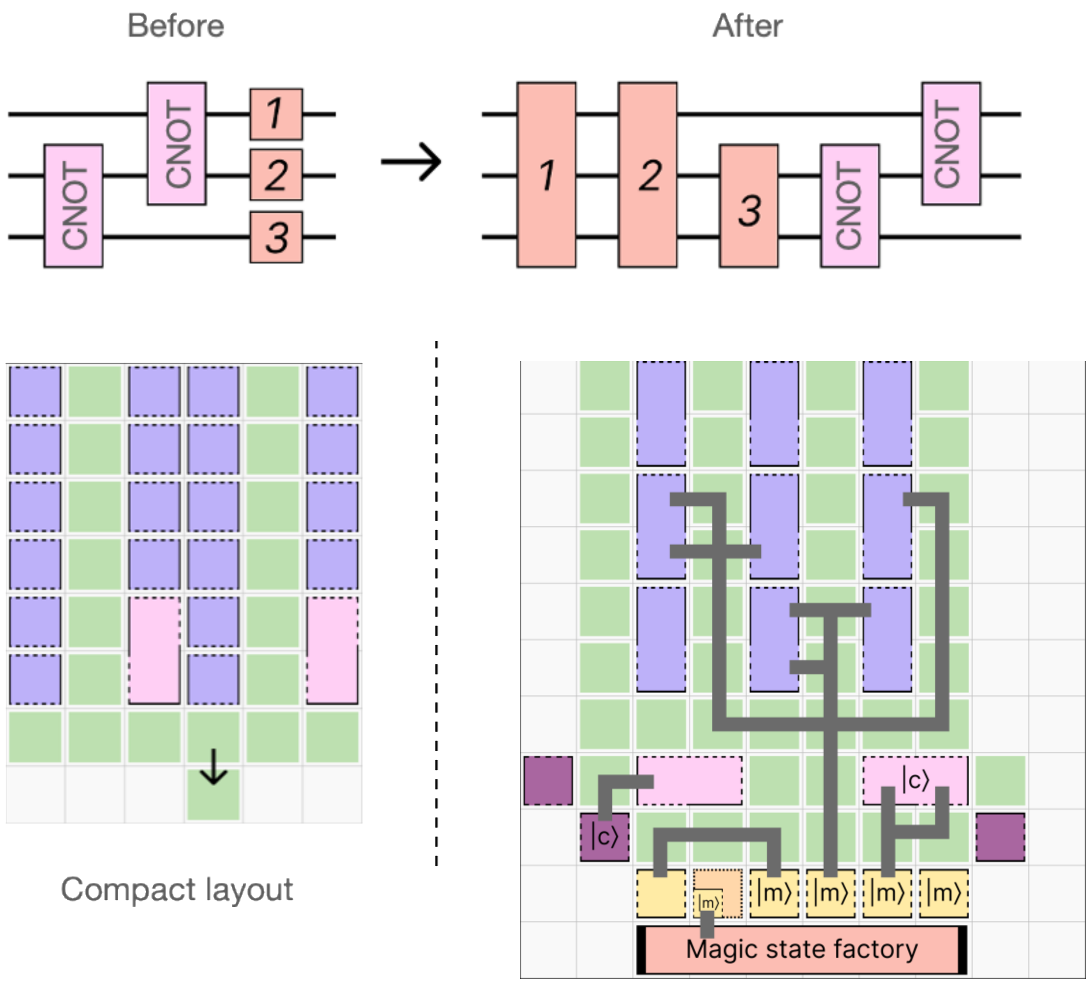
1. Commute π/4 rotations past π/8 rotations

2. Reduce π/4 and π/8 layers

3. Commute π/4 rotations past measurements



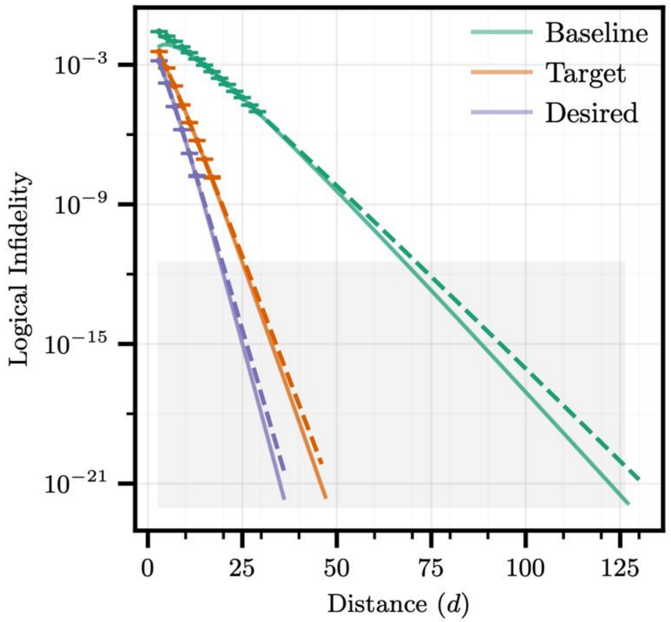
- Scheduler



1QBit

Having Realistic Noise Models Matters!

Hardware Parameter	Baseline	Target	Desired
T_1, T_2 times	100 μs	200 μs	340 μs
T_1 tailedness	71 μs	23 μs	23 μs
Single-qubit gate error	0.0004	0.0002	0.00012
Two-qubit gate error	0.003	0.0005	0.00029
State preparation error	0.02	0.01	0.00588
Measurement error	0.01	0.005	0.00294
Reset error	0.01	0.005	0.00294
Single-qubit gate time	25 ns	25 ns	25 ns
Two-qubit gate time	25 ns	25 ns	25 ns
State preparation time	1 μs	1 μs	1 μs
Measurement time	200 ns	100 ns	100 ns
Reset time	200 ns	100 ns	100 ns



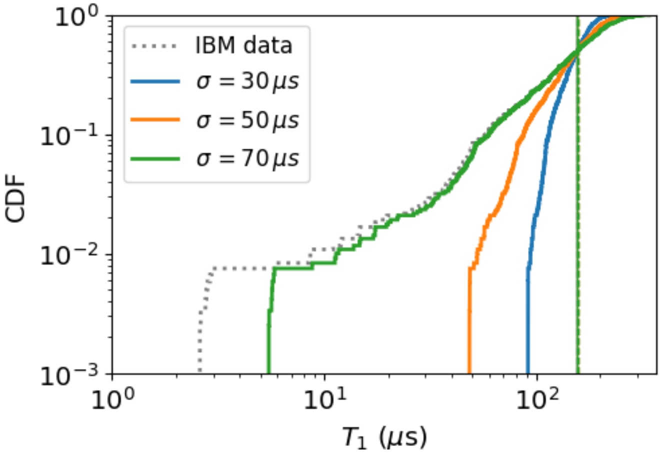
Quantum Physics

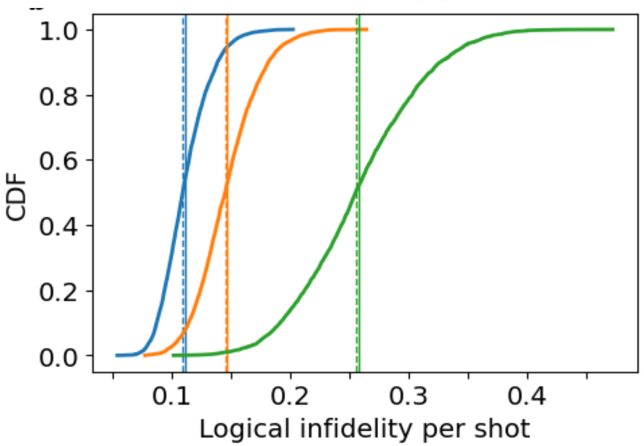
arXiv:2411.10406 (quant-ph)

[Submitted on 15 Nov 2024 (v1), last revised 31 Jan 2025 (this version, v2)]

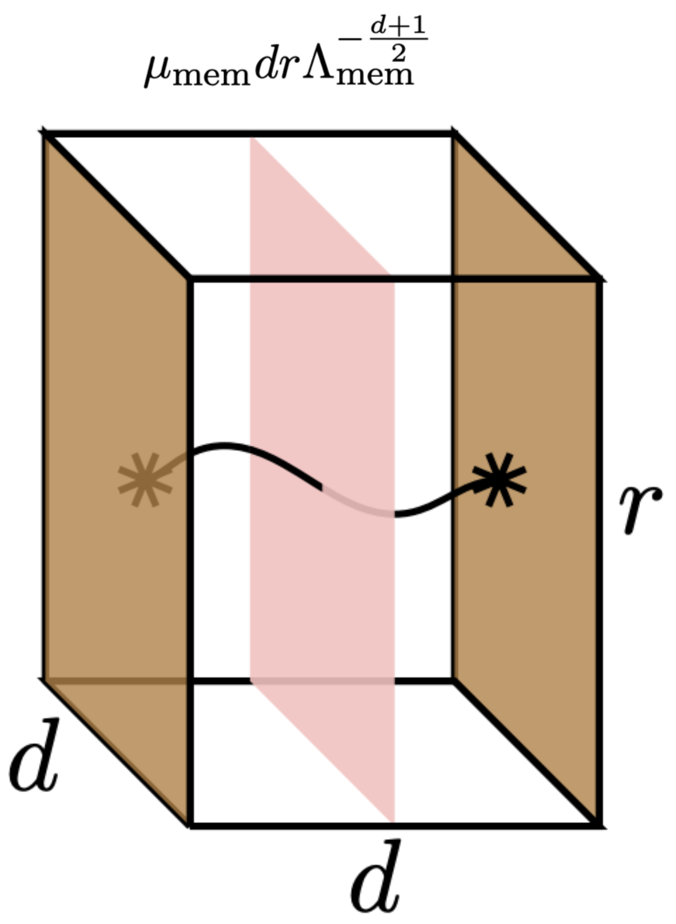
How to Build a Quantum Supercomputer: Scaling from Hundreds to Millions of Qubits

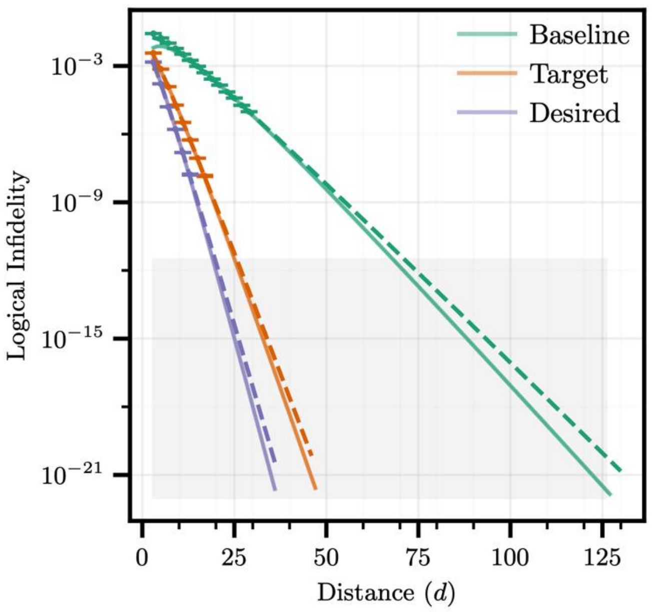
Masoud Mohseni, Artur Scherer, K. Grace Johnson, Oded Wertheim, Matthew Otten, Navid Anjum Aadit, Yuri Alexeev, Kirk M. Bresniker, Kerem Y. Camsari, Barbara Chapman, Soumitra Chatterjee, Gebremedhin A. Dagnew, Aniello Esposito, Farah Fahim, Marco Fiorentino, Archit Gajjar, Abdullah Khalid, Xiangzhou Kong, Bohdan Kulchytskyy, Elica Kyoseva, Ruoyu Li, P. Aaron Lott, Igor L. Markov, Robert F. McDermott, Giacomo Pedretti, Pooja Rao, Eleanor Rieffel, Allyson Silva, John Sorebo, Panagiotis Spentzouris, Ziv Steiner, Boyan Torosov, Davide Venturelli, Robert J. Visser, Zak Webb, Xin Zhan, Yonatan Cohen, Pooya Ronagh, Alan Ho, Raymond G. Beausoleil, John M. Martinis

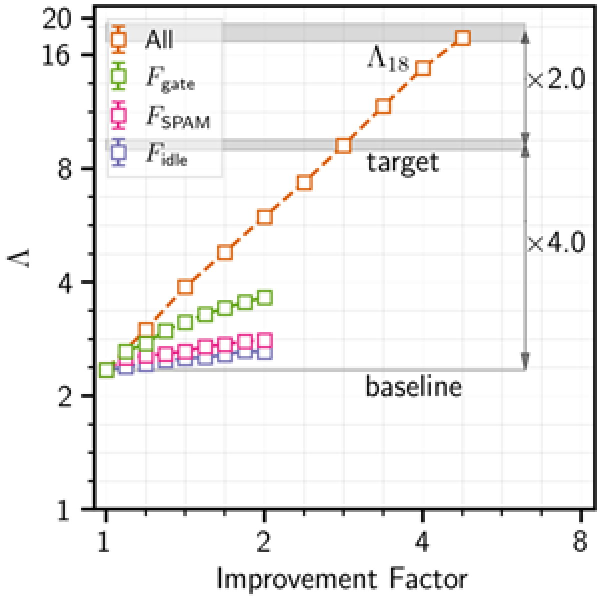


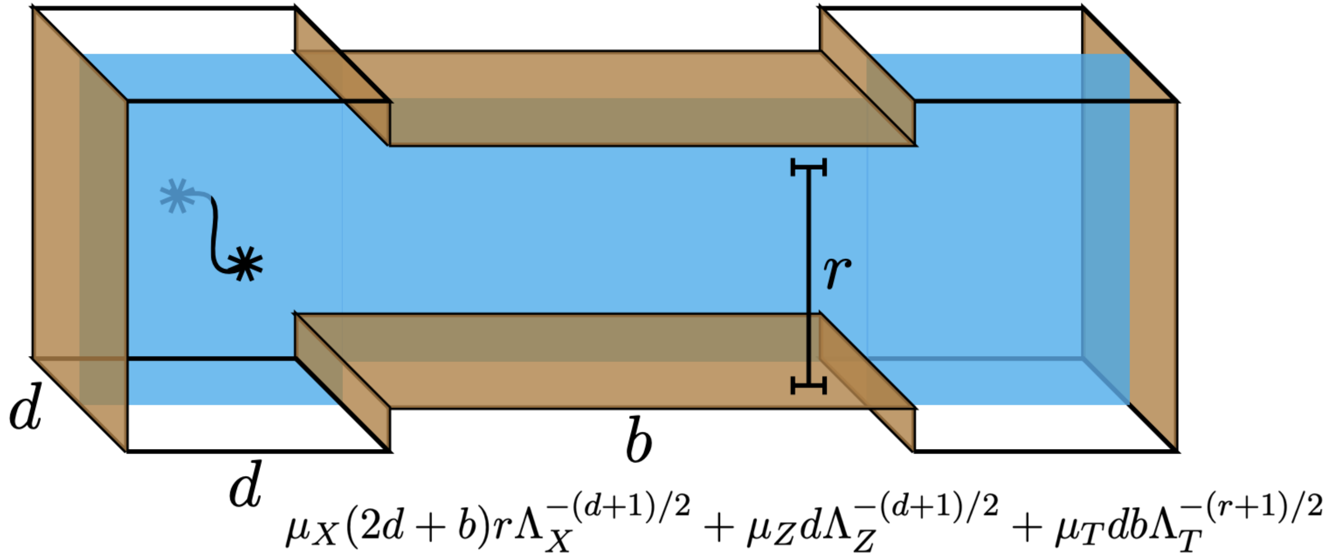


FTQC Emulation

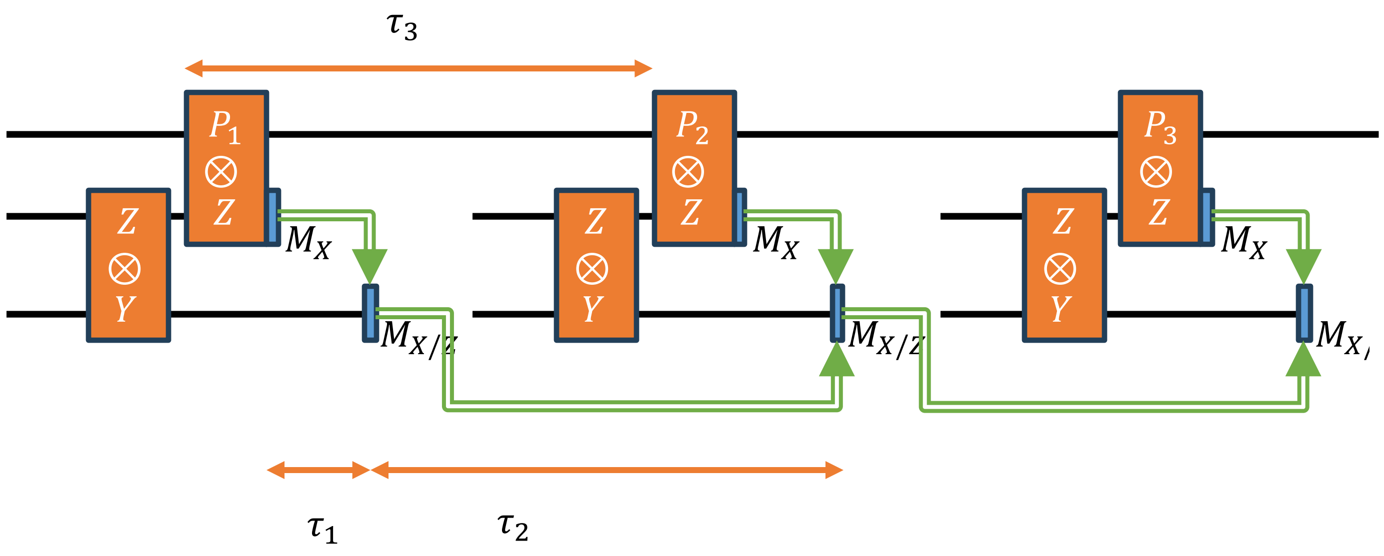








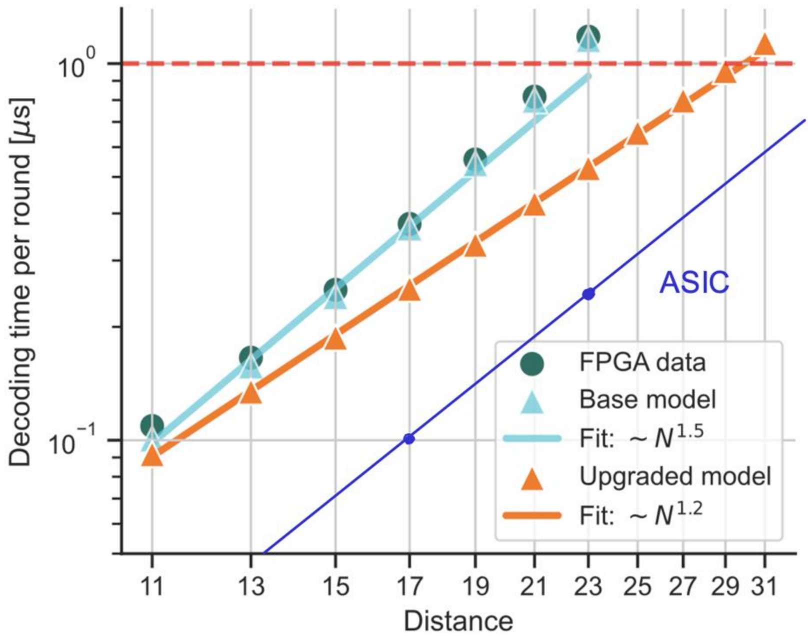
Impact of the Decoder Delay



$\gamma = \max(\tau_1, \tau_2, \tau_3)$ $\quad P_{\pi/8}(d) = P_{LS,ZY}(d) + P_{LS,PY}(d) + \gamma P_{\text{mem}}(d)$

- Spatial and temporal parallel decoding ⇒ decoder delay $\tau \coloneqq \tau_1 \simeq \tau_2$.
- Number of "bare" decoders scales with the physical qubit count.
- Fast qubits ⇒ $\tau_3 \ll \tau_1, \tau_2$ ⇒ single-core architecture.
- Slow qubits ⇒ $\tau_3 \gg \tau_1, \tau_2$ ⇒ multi−core architecture.

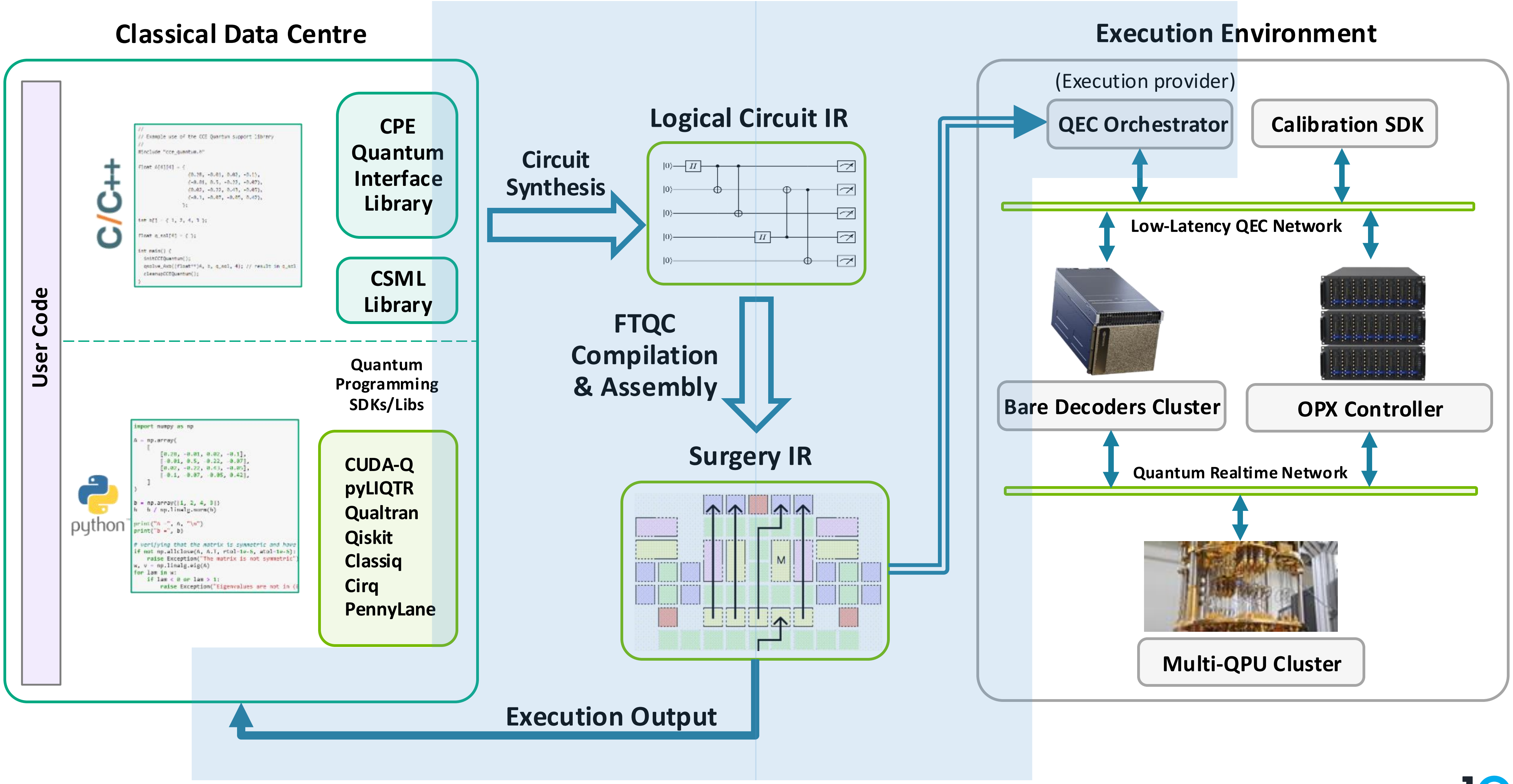
$\gamma \simeq 25$ logical cycles at $d = 35$ for target and desired using SOTA FPGA decoders



Barber, B. et al., Nat Electron 8 (2025): 1–8.
Lin, S. F. et al., Quantum Sci. Technol. 10, 035007 (2025).
Skoric, L. et al., Nat Commun 14, 7040 (2023).

1QBit

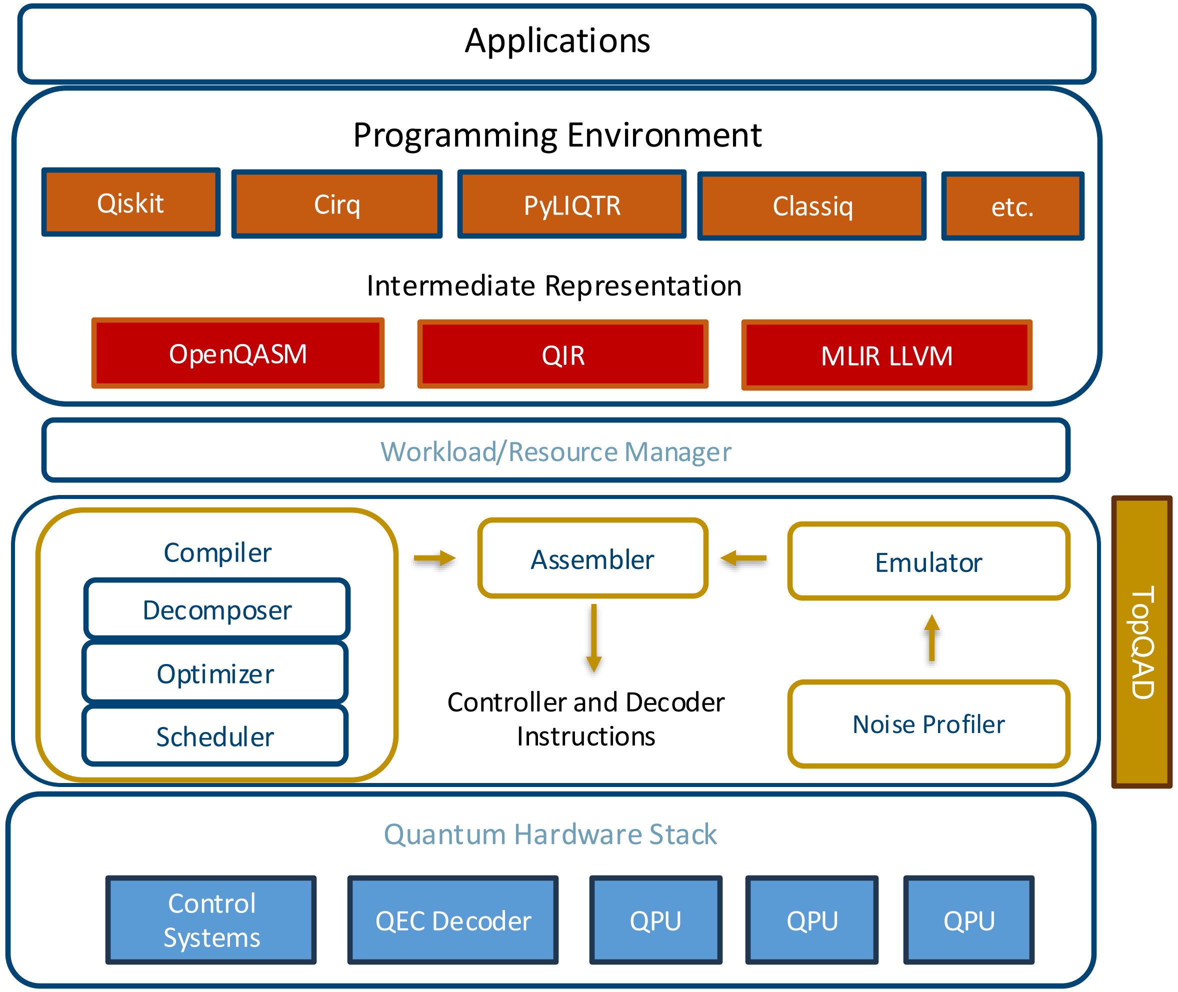
Toward a Quantum OS



1QBit

Main Components of FTQC Resource Estimator

	Input	*Output*
Compiler	Quantum **applications** coded in quantum assembly or other intermediate representation languages (e.g., QASM or QIR)	FTQC **compiled program** (lattice surgeries, state preparations, and decoding tasks for the core QPUs)
Noise Profiler	**Schedule** of QCVV (quantum characterization, validation, and verification) protocols of the QPUs	Realistic **noise models** incorporating hardware spec. distributions, cross-talk, leakage, etc.
Emulator	Realistic **noise models** from the previous step and a set of required FTQC protocols (multi-qubit lattice surgeries, QEC code growth and switching, magic state distillation, etc.)	Logical **error rate predictions** for FTQC protocols involving 1000+ noisy physical qubits, and incorporating decoder latencies and performance
Assembler	A multi-DR FTQC **layout**, FTQC protocol logical **error rate predictions** from the emulator, and FTQC compiled program from the compiler	**Machine-level instructions** for controllers and decoders (stabilizer measurement cycles, mid-circuit logical measurements, and conditional recovery operations)



1QBit

TopQAD account creation and activation

Katie Olfert (*10 min*)

Software Development Lead, 1QBit

1QBit

TopQAD Portal: The Browser Experience

https://topqad.1qbit.com

Trial Code: **********

Sign-up Checklist

- ✅ Visit topqad.1qbit.com and follow the “Portal” link
- ☑ Sign up -> enter desired account details
- ☑ Verify email
- ☑ Enter trial code
- ☑ Accept terms and conditions
- ☑ Authenticate with MFA

1QBit

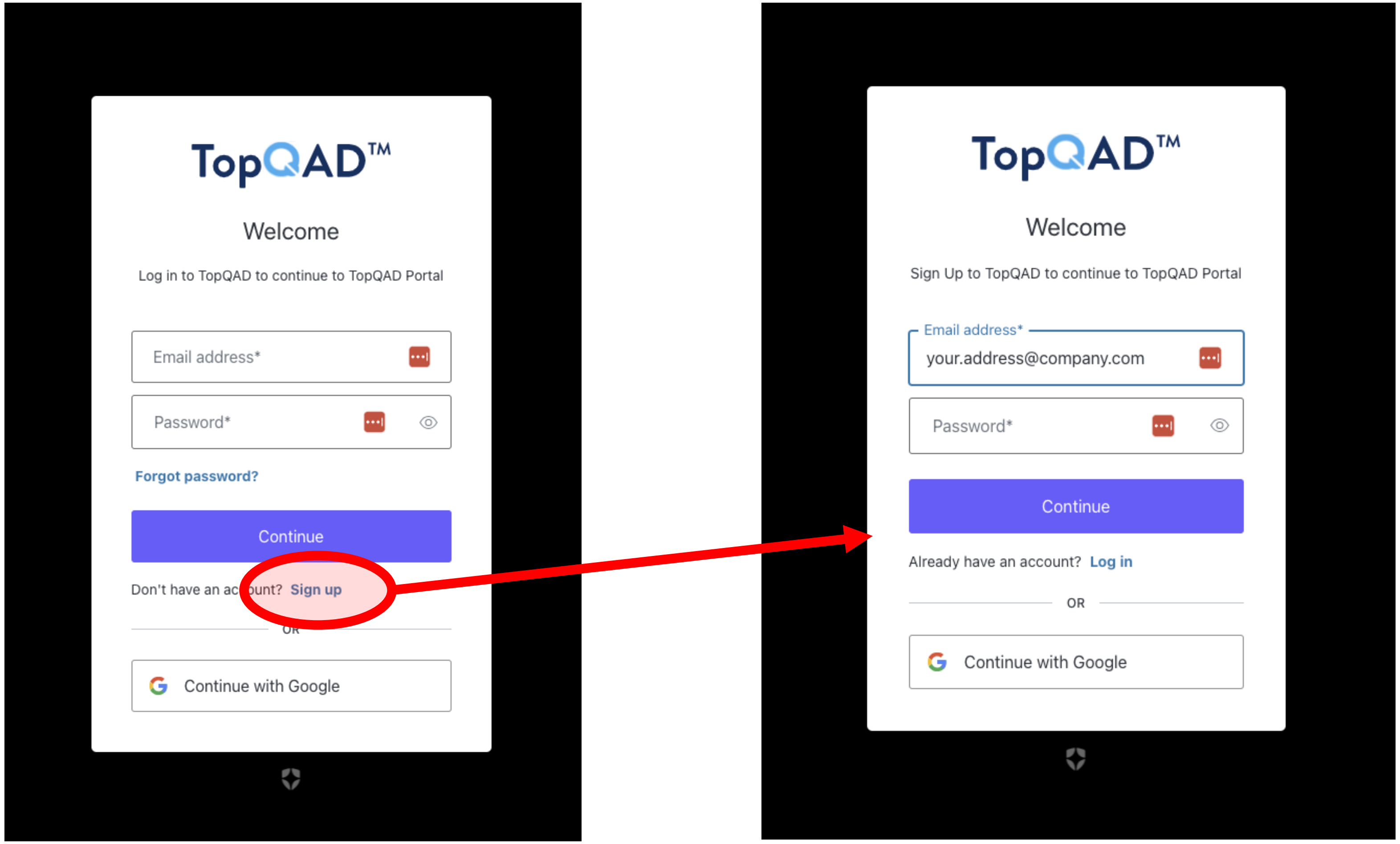
Create a TopQAD Account

https://topqad.1qbit.com

Trial Code: **********

Sign-up Checklist

- ✅ Visit topqad.1qbit.com and follow the “Portal” link
- ✅ Sign up -> enter desired account details
- ☑ Verify email
- ☑ Enter trial code
- ☑ Accept terms and conditions
- ☑ Authenticate with MFA



1QBit

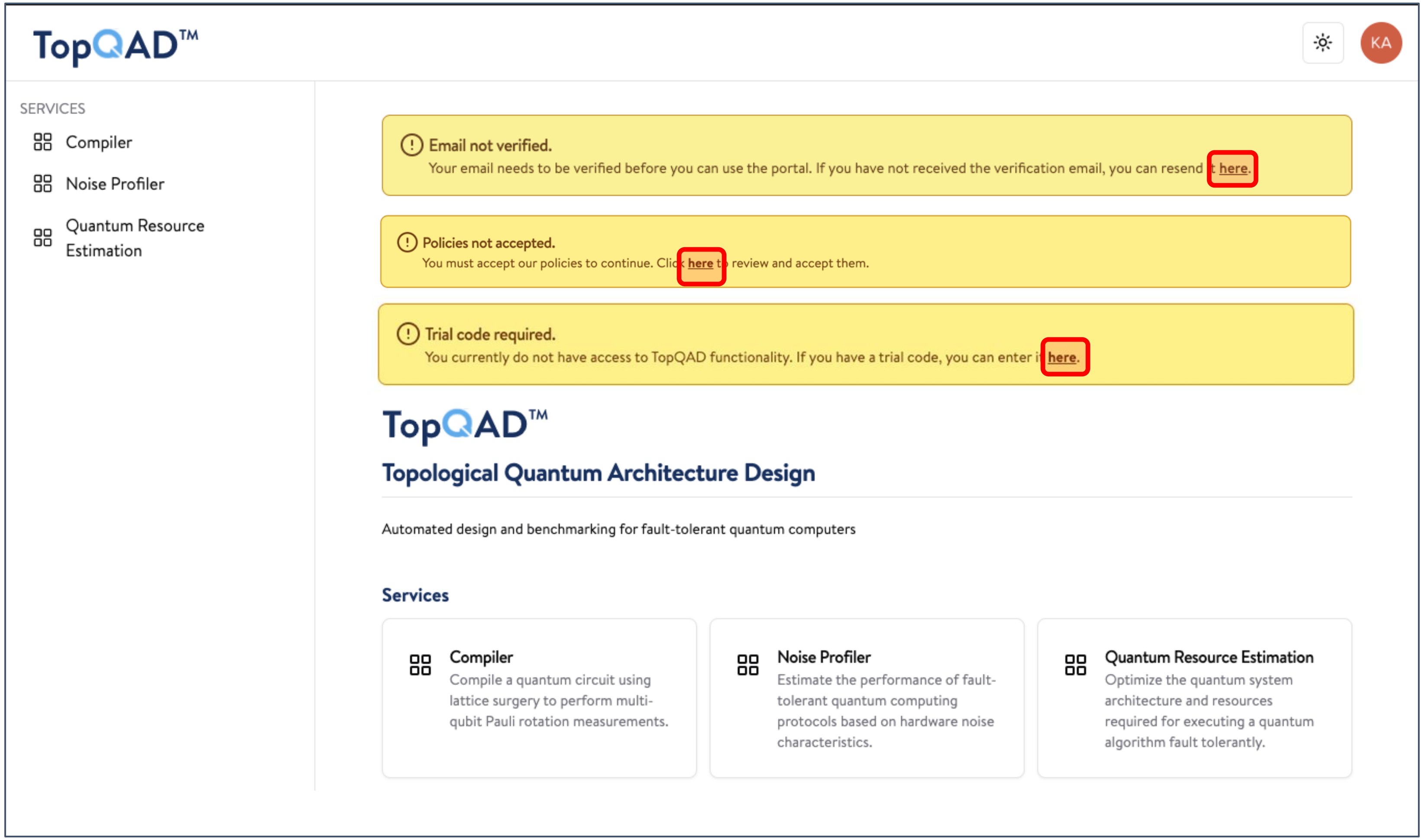
Verify Email

https://topqad.1qbit.com

Trial Code: **********

Sign-up Checklist

- ✅ Visit topqad.1qbit.com and follow the “Portal” link
- ✅ Sign up -> enter desired account details
- ✅ Verify email
- ☑ Enter trial code
- ☑ Accept terms and conditions
- ☑ Authenticate with MFA



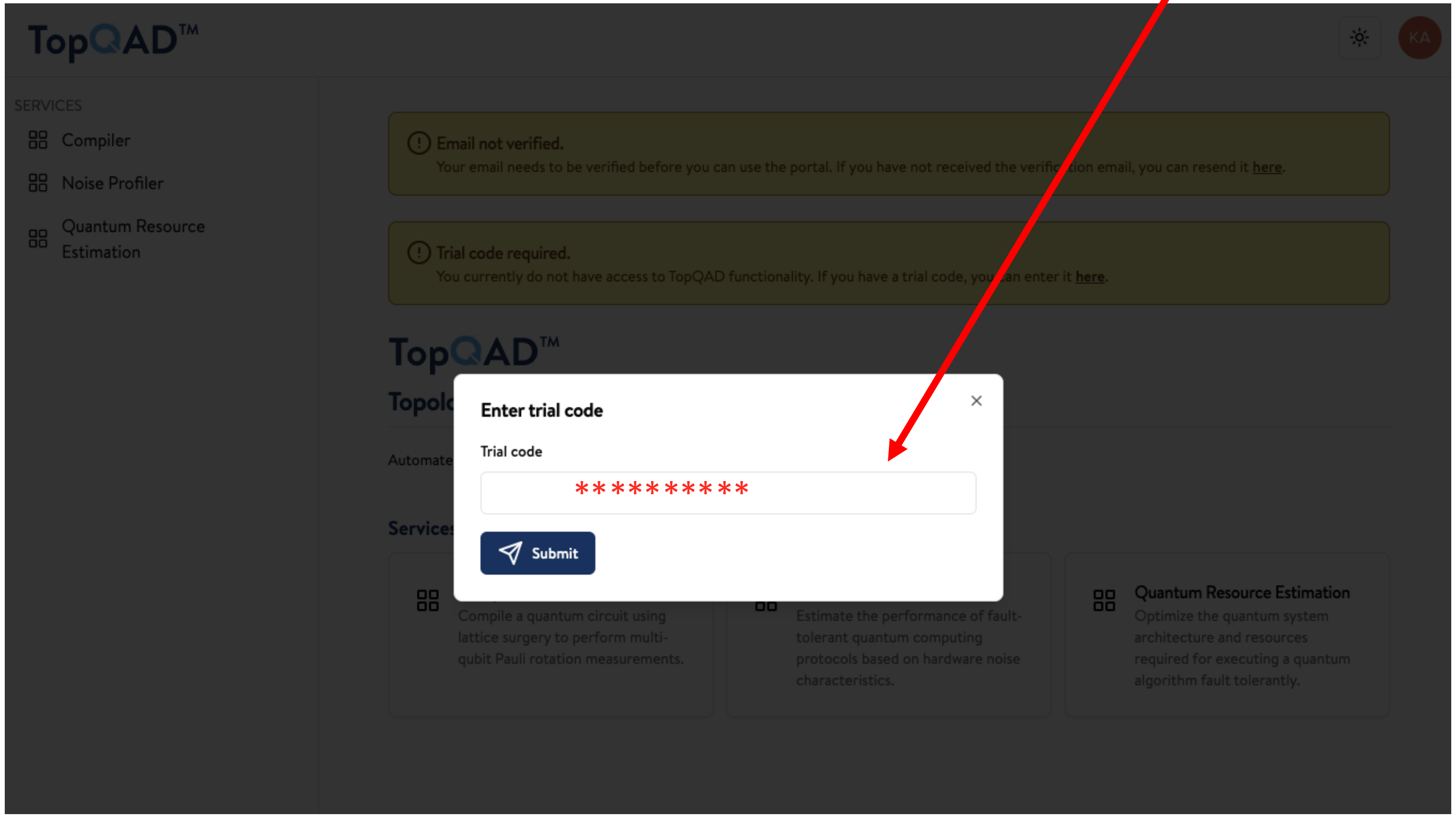
Trial Code

https://topqad.1qbit.com

Trial Code: **********

Sign-up Checklist

- ✅ Visit topqad.1qbit.com and follow the “Portal” link
- ✅ Sign up -> enter desired account details
- ✅ Verify email
- ✅ Enter trial code
- ☑ Accept terms and conditions
- ☑ Authenticate with MFA



1QBit

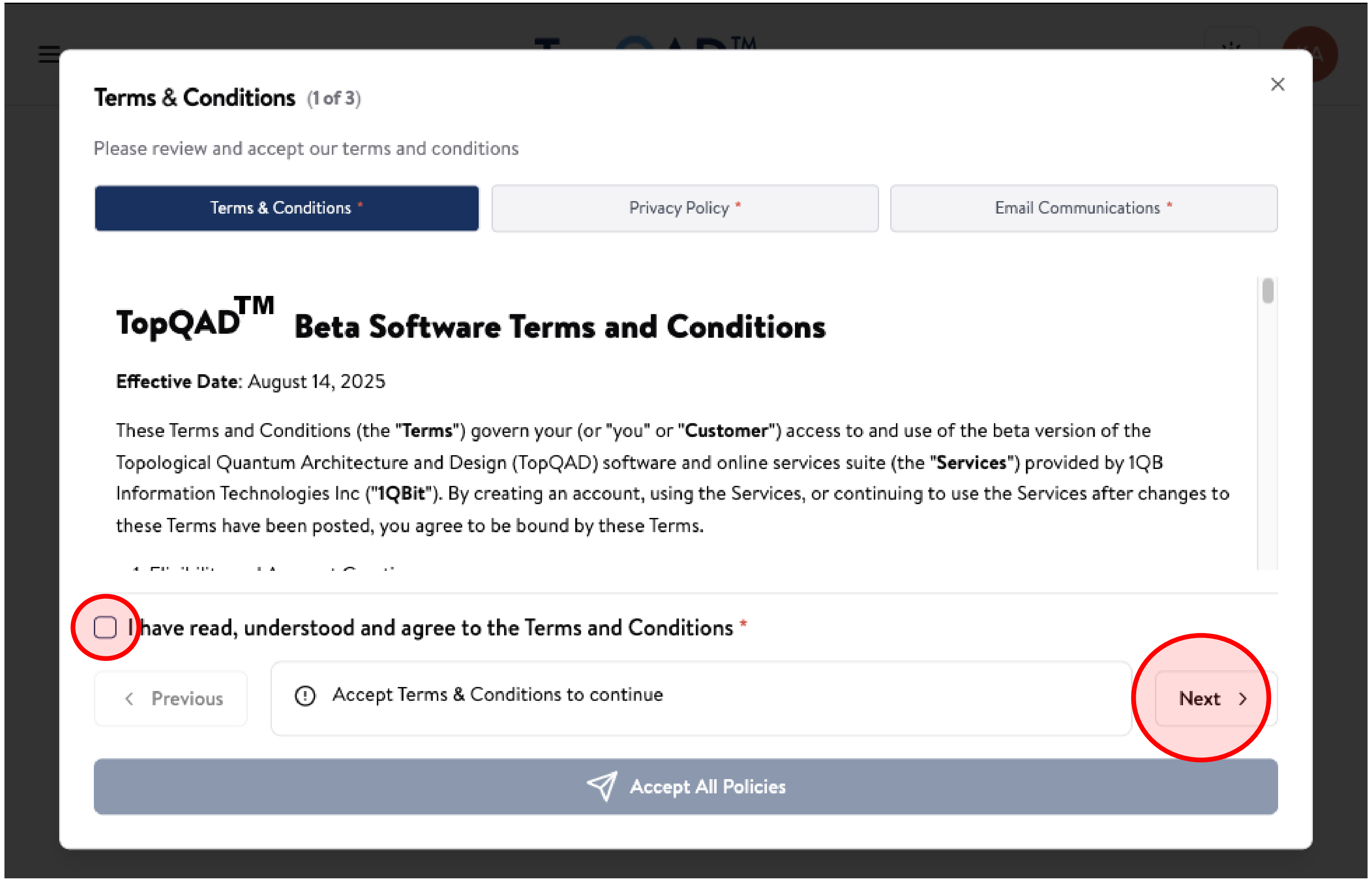
Terms and Conditions

https://topqad.1qbit.com

Trial Code: **********

Sign-up Checklist

- ✅ Visit topqad.1qbit.com and follow the "Portal" link
- ✅ Sign up -> enter desired account details
- ✅ Verify email
- ✅ Enter trial code
- ✅ Accept terms and conditions
- ☑ Authenticate with MFA



1QBit

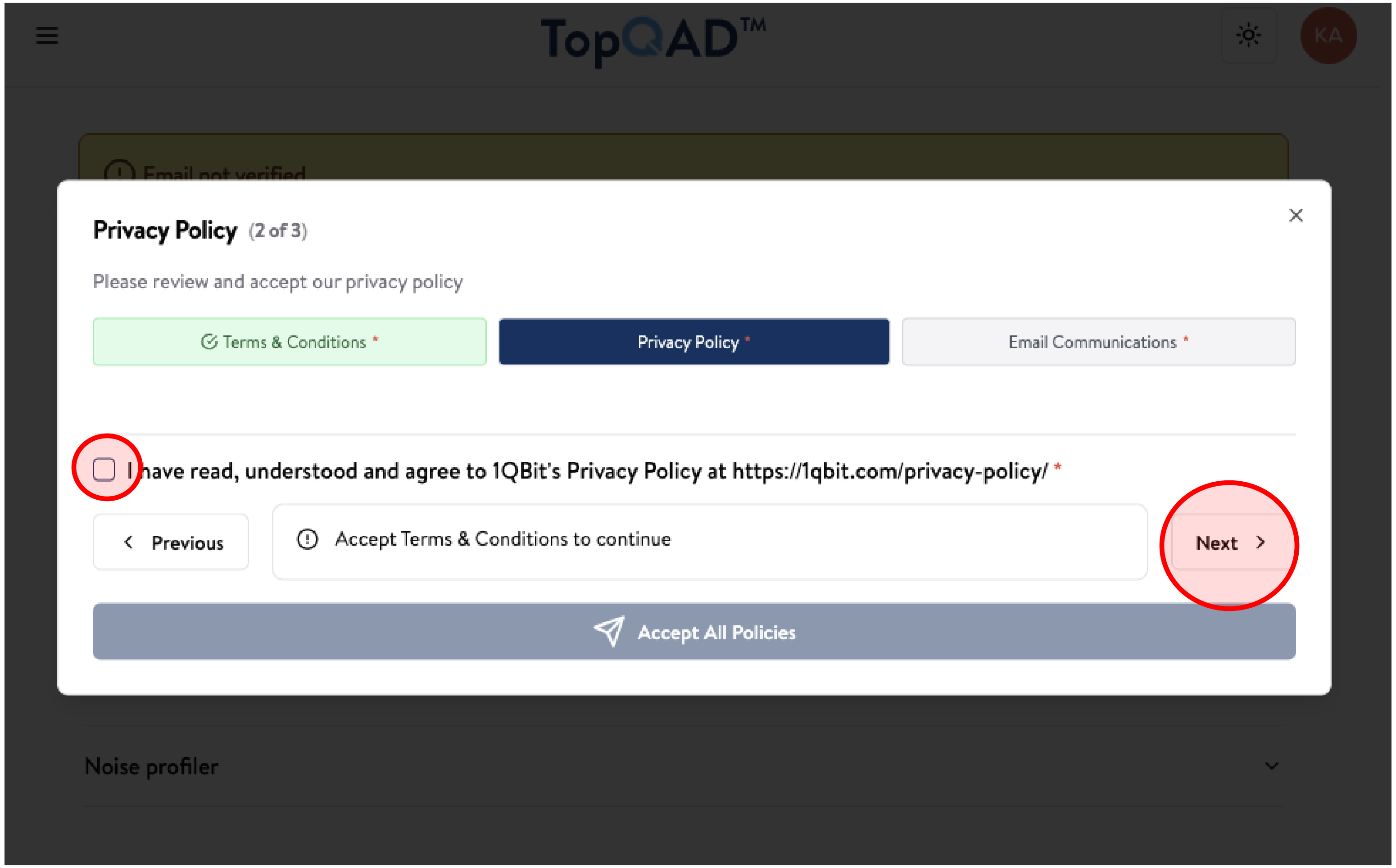
Terms and Conditions

https://topqad.1qbit.com

Trial Code: **********

Sign-up Checklist

- ✅ Visit topqad.1qbit.com and follow the “Portal” link
- ✅ Sign up -> enter desired account details
- ✅ Verify email
- ✅ Enter trial code
- ✅ Accept terms and conditions
- ☑ Authenticate with MFA



1QBit

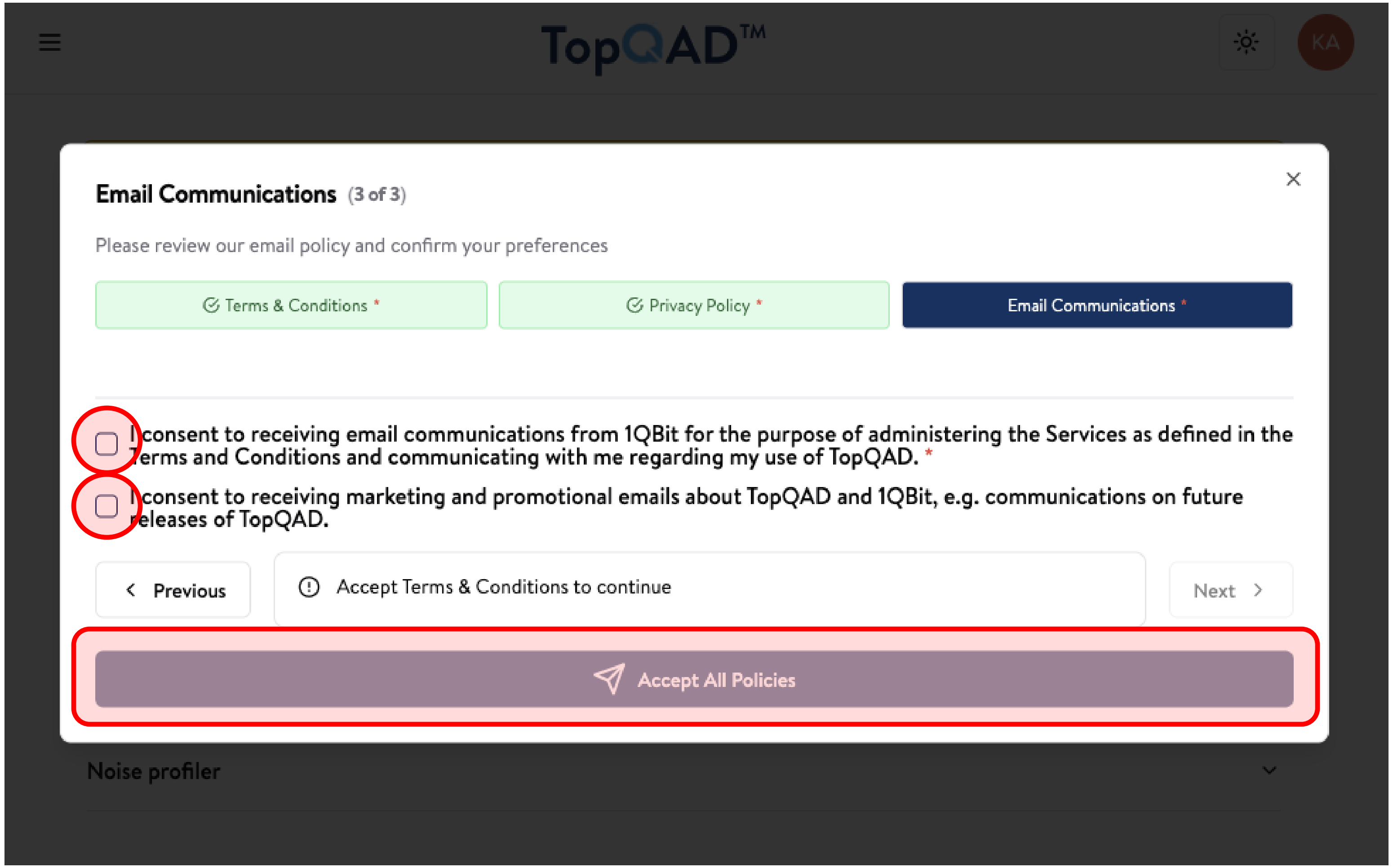
Terms and Conditions

https://topqad.1qbit.com

Trial Code: **********

Sign-up Checklist

- ✅ Visit topqad.1qbit.com and follow the “Portal” link
- ✅ Sign up -> enter desired account details
- ✅ Verify email
- ✅ Enter trial code
- ✅ Accept terms and conditions
- ☑️ Authenticate with MFA



1QBit

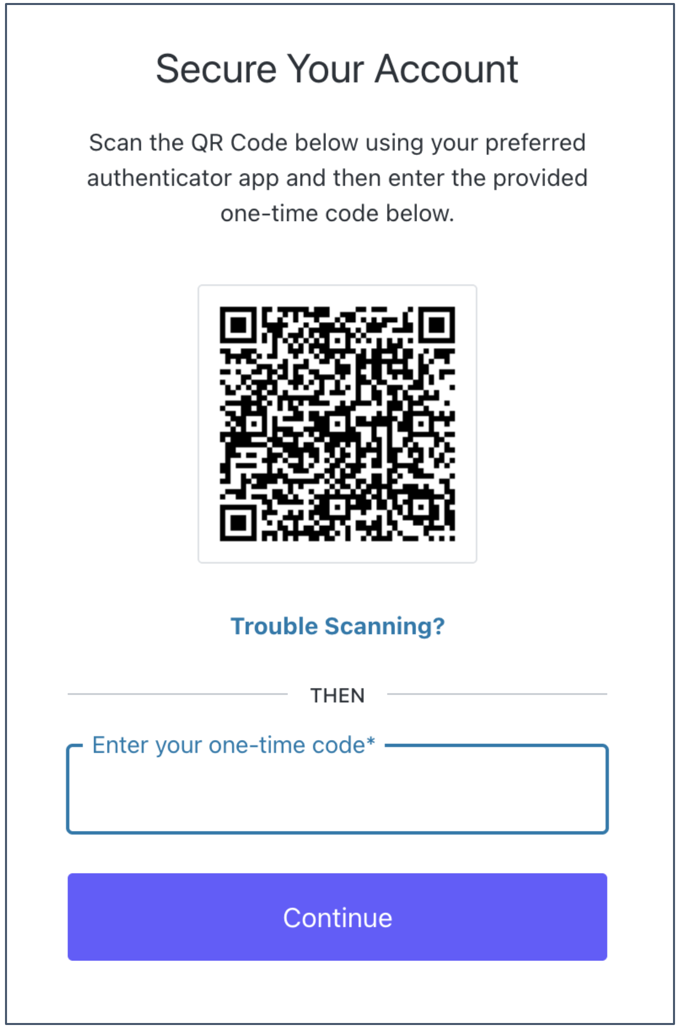
Multi-factor Authentication (MFA)

https://topqad.1qbit.com

Trial Code: **********

Sign-up Checklist

- ✅ Visit topqad.1qbit.com and follow the “Portal” link
- ✅ Sign up -> enter desired account details
- ✅ Verify email
- ✅ Enter trial code
- ✅ Accept terms and conditions
- ✅ Authenticate with MFA

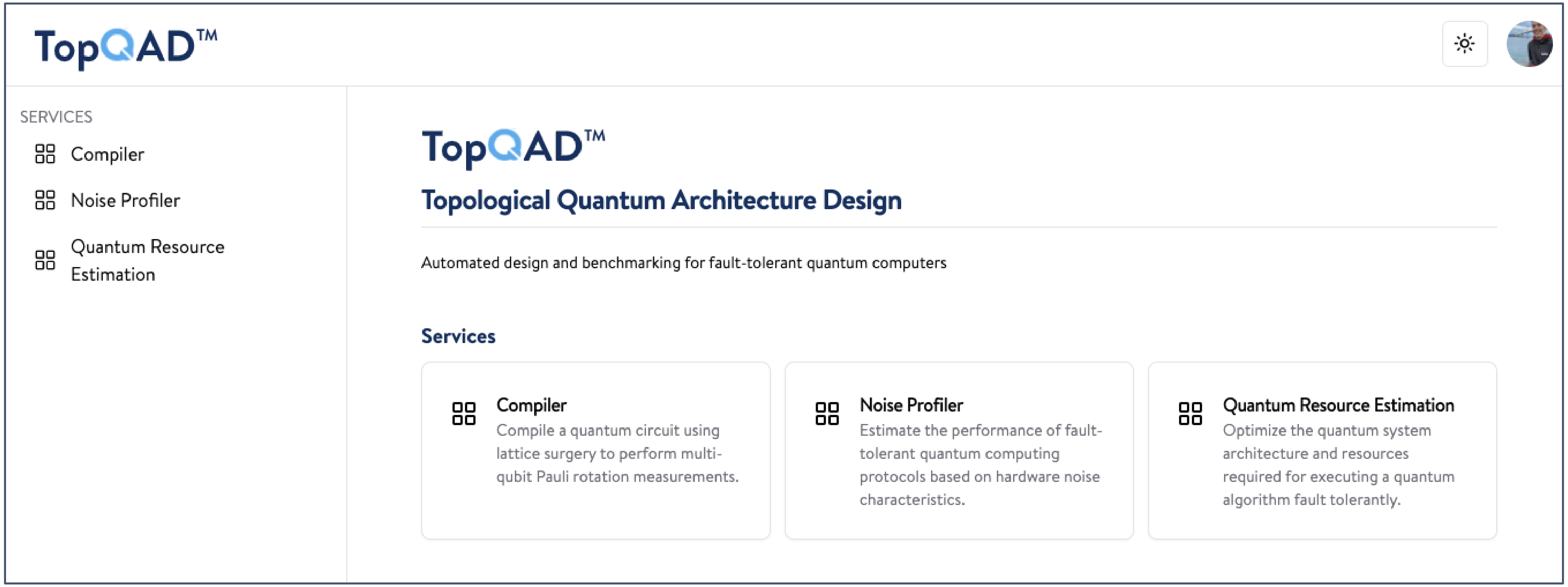


1QBit

Ready to Explore

https://topqad.1qbit.com

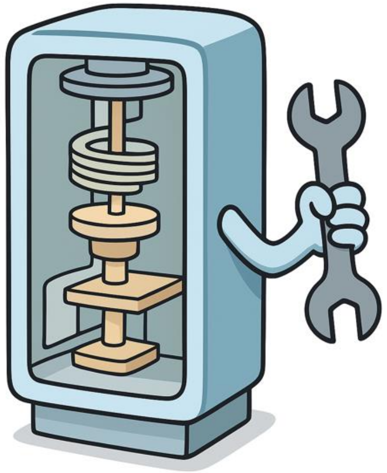
Trial Code: **********



Fault-tolerant compilation

Zak Webb (*25 min*)
Senior Scientist, 1QBit

Fault-Tolerant Circuit Synthesis

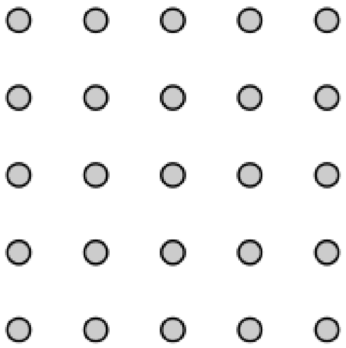


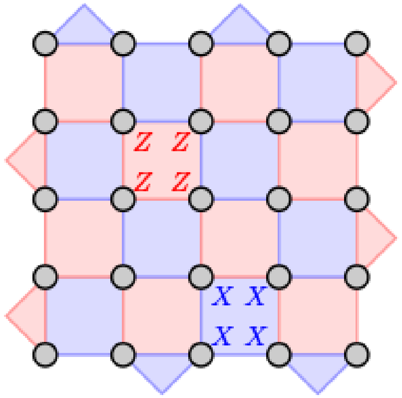
- Recap:
  - Need to protect against errors **throughout** computation
  - **QEC code** decoding at each timestep
  - **Fault tolerance** pertains to ensuring QEC overheads do not cause additional errors
- To achieve a **universal logical gate set**:
  - Clifford gates are often easier to implement using **transversal** operations or lattice surgeries
  - Since a universal transversal gate set does not exist:
    - **Magic states** used to implement the “hard” gates for a given QEC code
    - Switching between **two codes** to use transversality of different codes for different gates
- Circuit synthesis: the process of transforming an input circuit to a fault-tolerant circuit

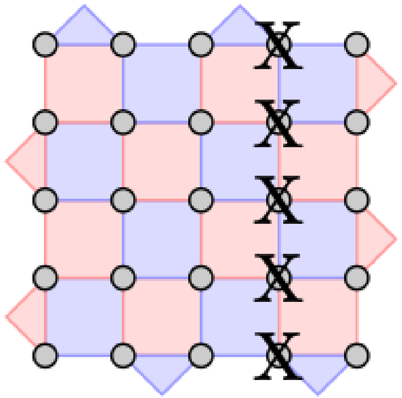
1QBit

Surface Code

- Useful QEC code for fault tolerance due to ease of physical implementation:
  - Qubits arranged in 2D grid and are required to interact only with nearest neighbours
- Encode one qubit into a $d \times d$ square lattice (**a patch**)
  - Can embed more than one qubit using multiple patches
- **Stabilizer measurement**: performed using interacting neighbouring qubits to detect physical errors
- Logical errors correspond to **string operators** running across the lattice
  - Short excitation strings are detected and corrected for by the decoder



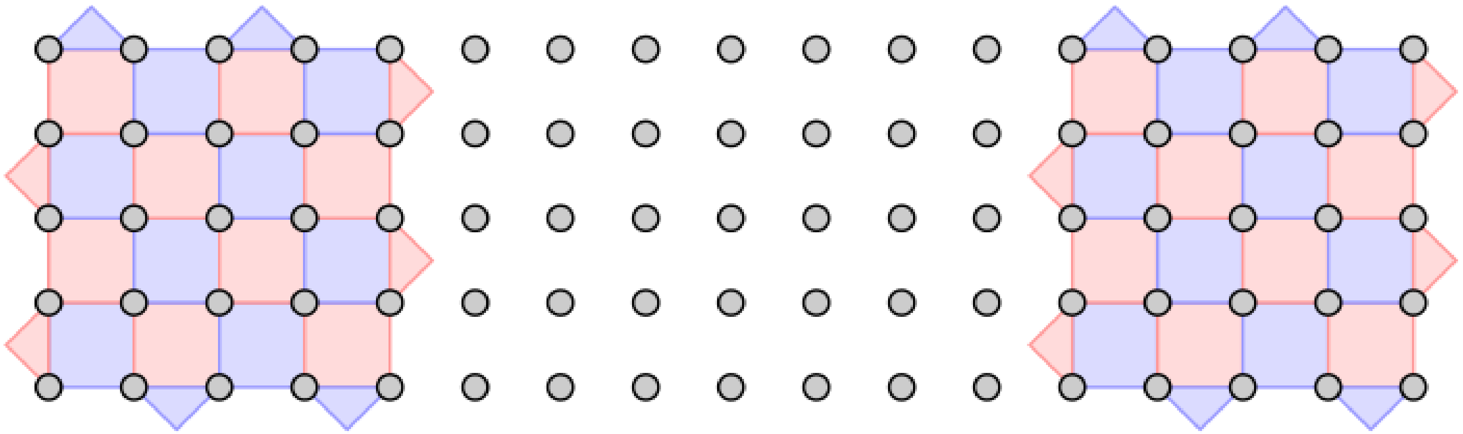




1QBit

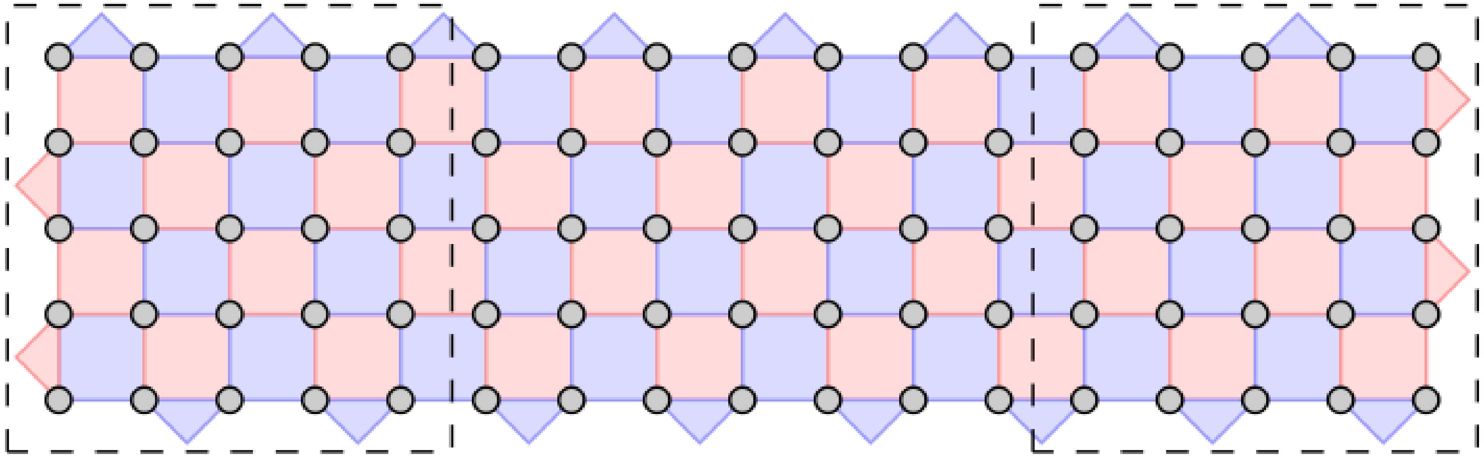
Lattice Surgery

- Method of implementing many gates fault-tolerantly on surface code
  - **Pauli gates**: all products and tensor products of X, Y, and Z
  - **Clifford gates**: all products of Pauli gates, Hadamard gates, S gates, and CNOT gates
- Corresponds to measuring Pauli-product operators on logical system
- Implemented by extending stabilizer measurements to lattice between encoded qubits
  - The Pauli measured depends on which edge we connect



Lattice Surgery

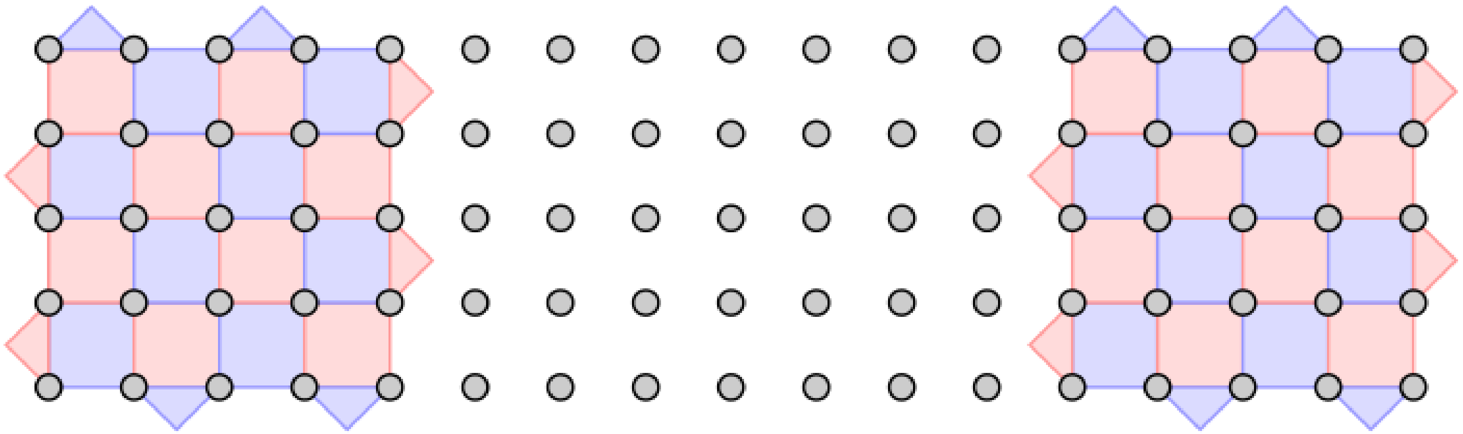
- Method of implementing many gates fault-tolerantly on surface code
  - **Pauli gates**: all products and tensor products of X, Y, and Z
  - **Clifford gates**: all products of Pauli gates, Hadamard gates, S gates, and CNOT gates
- Corresponds to measuring Pauli-product operators on logical system
- Implemented by extending stabilizer measurements to lattice between encoded qubits
  - The Pauli measured depends on which edge we connect



1QBit

Lattice Surgery

- Method of implementing many gates fault-tolerantly on surface code
  - **Pauli gates**: all products and tensor products of X, Y, and Z
  - **Clifford gates**: all products of Pauli gates, Hadamard gates, S gates, and CNOT gates
- Corresponds to measuring Pauli-product operators on logical system
- Implemented by extending stabilizer measurements to lattice between encoded qubits
  - The Pauli measured depends on which edge we connect



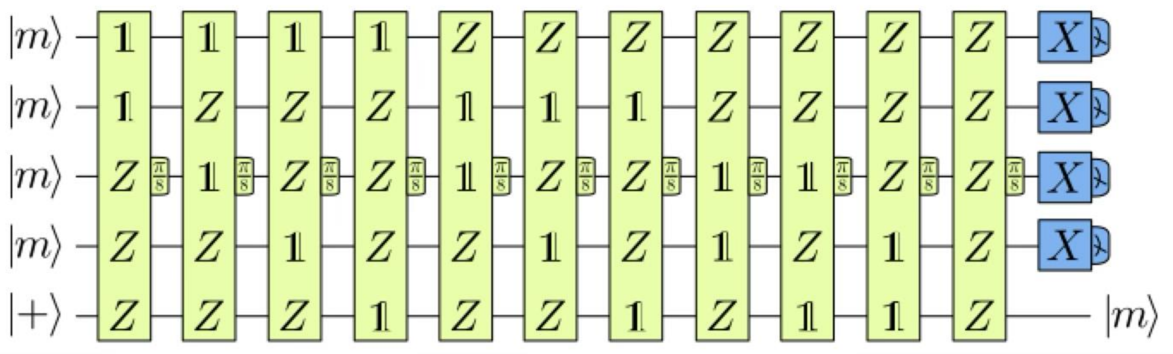
Magic States

- Need some gate outside Clifford gate set for universal computation on surface code
- $\boldsymbol{T}$ **gates** are a common addition to the Clifford gate set
  - $\boldsymbol{Z}$ **rotation** by an angle of $\boldsymbol{\frac{\pi}{8}}$
  - Fourth root of Z gate
- Implementable via **Clifford gates** and a $\boldsymbol{T}$ **state**
  - Does require some correction
- Means that lattice surgery plus a source of magic states can perform any quantum computation

1QBit

Preparing Magic States with High Fidelity

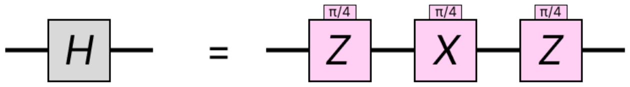
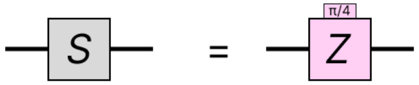
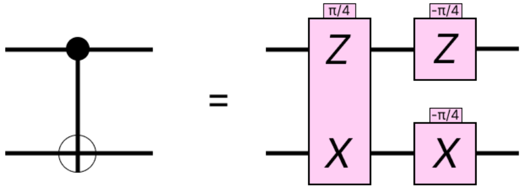
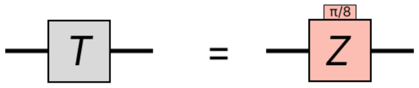
- Magic states **cannot** be prepared **inside** a QEC code
- Need special area outside of fault-tolerant space to create magic states
  - Start from **physical** magic state
  - Implement **magic state distillation** protocol to transform "bad" magic states into "better" magic states
  - Repeat until desired fidelity reached
  - Can optimize different protocols



The Pauli-Product Rotations Instruction Set

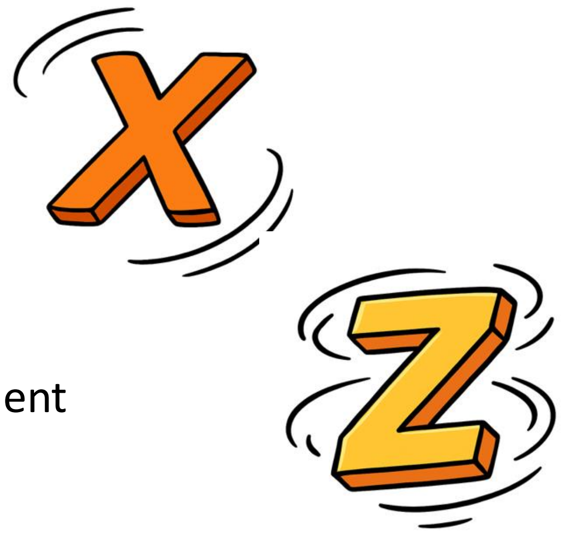
- Native gate set of lattice surgery slightly different from Clifford+T
  - In general, change gate set to one that is easy to implement via lattice surgery
- **Pauli-product rotations** instruction set:
  - Each gate composed of a multi-qubit Pauli operator and angle
    - Angle is only $\frac{\pi}{2}$, $\frac{\pi}{4}$, or $\frac{\pi}{8}$
  - Rotation about operator by the angle
  - Easy to convert Clifford+ T gates to this gate set

$$|\psi\rangle \mapsto \exp(-i\,\theta P)\,|\psi\rangle$$

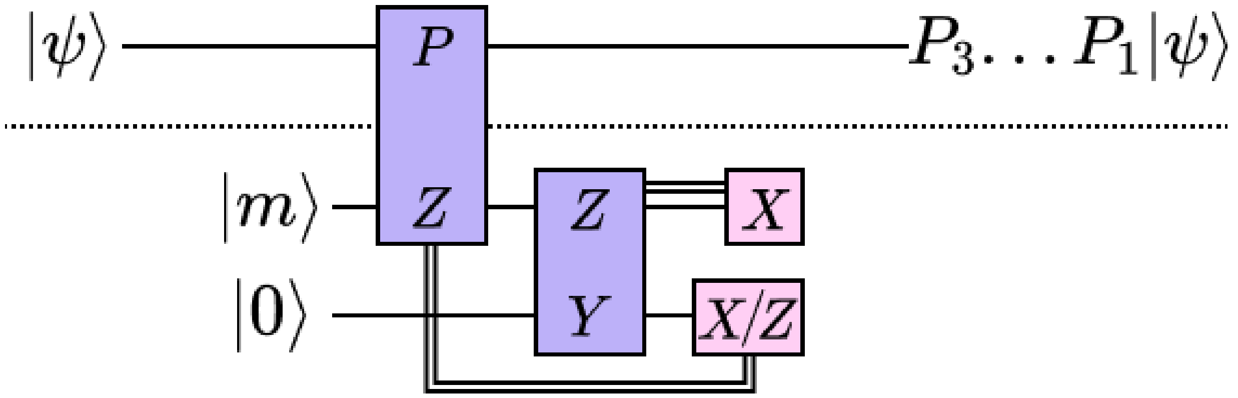
Gate	Conversion rule
Hadamard	
Phase	
CNOT	
T	

1QBit

Implementation of the ISA via Lattice Surgery

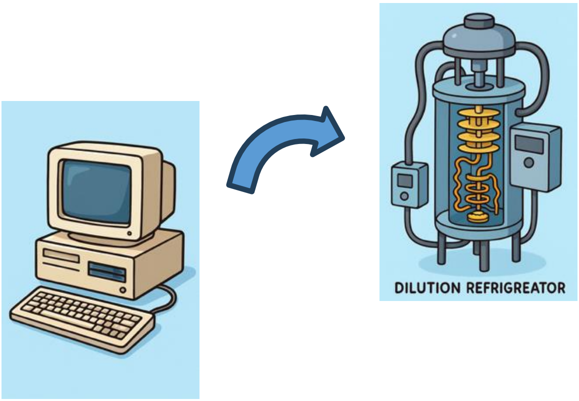


- Simple implementation in lattice surgery for most gates
  - **Pauli** gates change meaning of underlying measurements
    - Implemented in software
  - **Clifford** gates can be implemented with ancilla $|0\rangle$ and Pauli measurement
    - Uses easy-to-create ancilla
  - $\boldsymbol{T}$ **gates** implemented with ancilla magic states and Pauli measurements
    - Uses hard-to-create ancilla



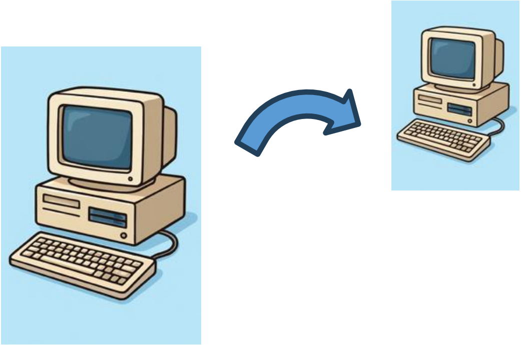
Circuit Synthesis

- **Circuit synthesis** is process of creating a quantum circuit of a target form.
  - Often involves transforming to a standard universal gate set (Clifford+T)
- Need to ensure the total error of the circuit remains below target threshold while resource overhead remains reasonable
  - Decomposition of circuit into universal gate set
  - Implementation of individual gates
- **Solovay–Kitaev theorem** shows how to decompose single qubit gates into small gate sets
- **Threshold theorem** states that increasing code distance causes **exponential suppression** of error
  - Some assumptions on error model
- Some improved algorithms over those in these theorems:
  - gridsynth is much faster than Solovay–Kitaev



Circuit Optimization

- Circuit synthesis only creates circuit using target gate set with **no optimality claim**

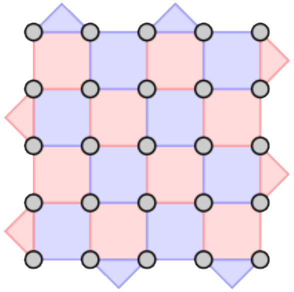


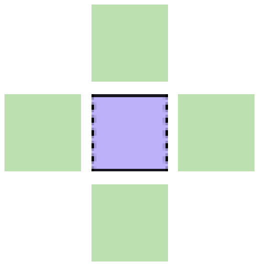
- Can then send circuit through **simple transforms** to reduced size:
- Neighbouring Hadamard gates cancel out
  - Cliffords can be exchanged

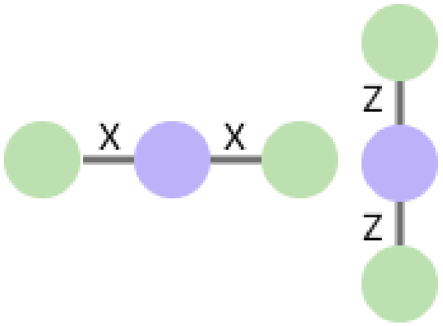
- Specific gate sets allow for additional operations:
  - The Pauli rotations ISA allows us to commute any two neighbouring gates (with modifications)
    - This allows us to remove all $\frac{\pi}{4}$ gates from circuit
    - This operation greatly reduces parallelization in circuit
- These operations have **computational cost**
  - We then need to balance decrease in quantum costs with increase in classical costs
- Once desired trade-offs are known, we can produce an **optimized fault-tolerant circuit**

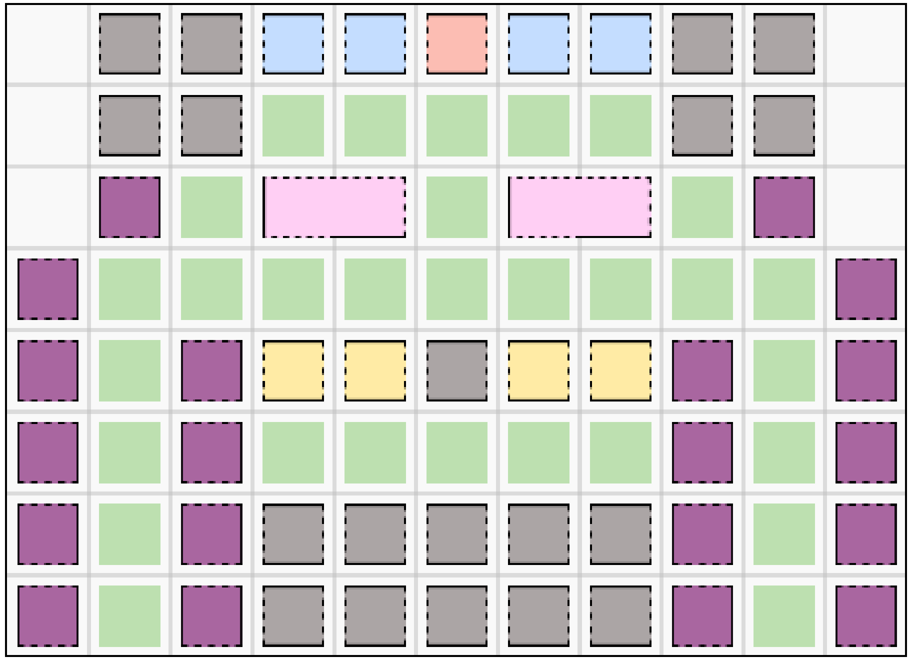
Lattice Surgery Abstraction

- Assuming all qubits are encoded in a surface code, we can then abstract away physical system and examine the encoded logical spaces
- Now have a lattice of encoded qubits
- Can define each encoded qubit to have specific job
  - Data qubit
  - Bus qubit
  - Etc.





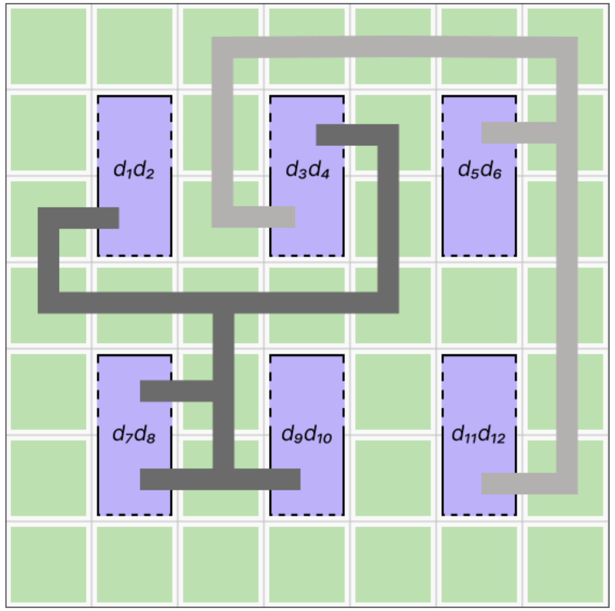




1QBit

Scheduling of Lattice Surgeries

- Physical systems have additional restrictions over logical circuits
- Lattice surgery has limit on number of interactions with a single qubit per timestep
  - Two X edges and two Z edges
- Each bus qubit cannot be in two operations
  - Bus conflicts
- Optimize bus use
  - Determine bus for a single operation via Steiner trees
- Create a list of circuit restrictions
  - DAG data structure of gate order



Quantum Resource Estimation Service Steps

QRE Execution Steps

- ✓ Synthesizing
  - ✓ Decomposing into Clifford+T gates
  - ✓ Removing Clifford gates and converting to Pauli product rotation gates
- ✓ Scheduling
  - ✓ Parsing rotations
  - ✓ Scheduling operations
- ✓ Noise profiling
- ✓ Assembling
- ✓ Generating report

Transform gates

Optimize

Transform input circuit into optimized fault-tolerant circuit

We need to transform gate sets to one we can perform fault-tolerantly and then optimize the circuit.

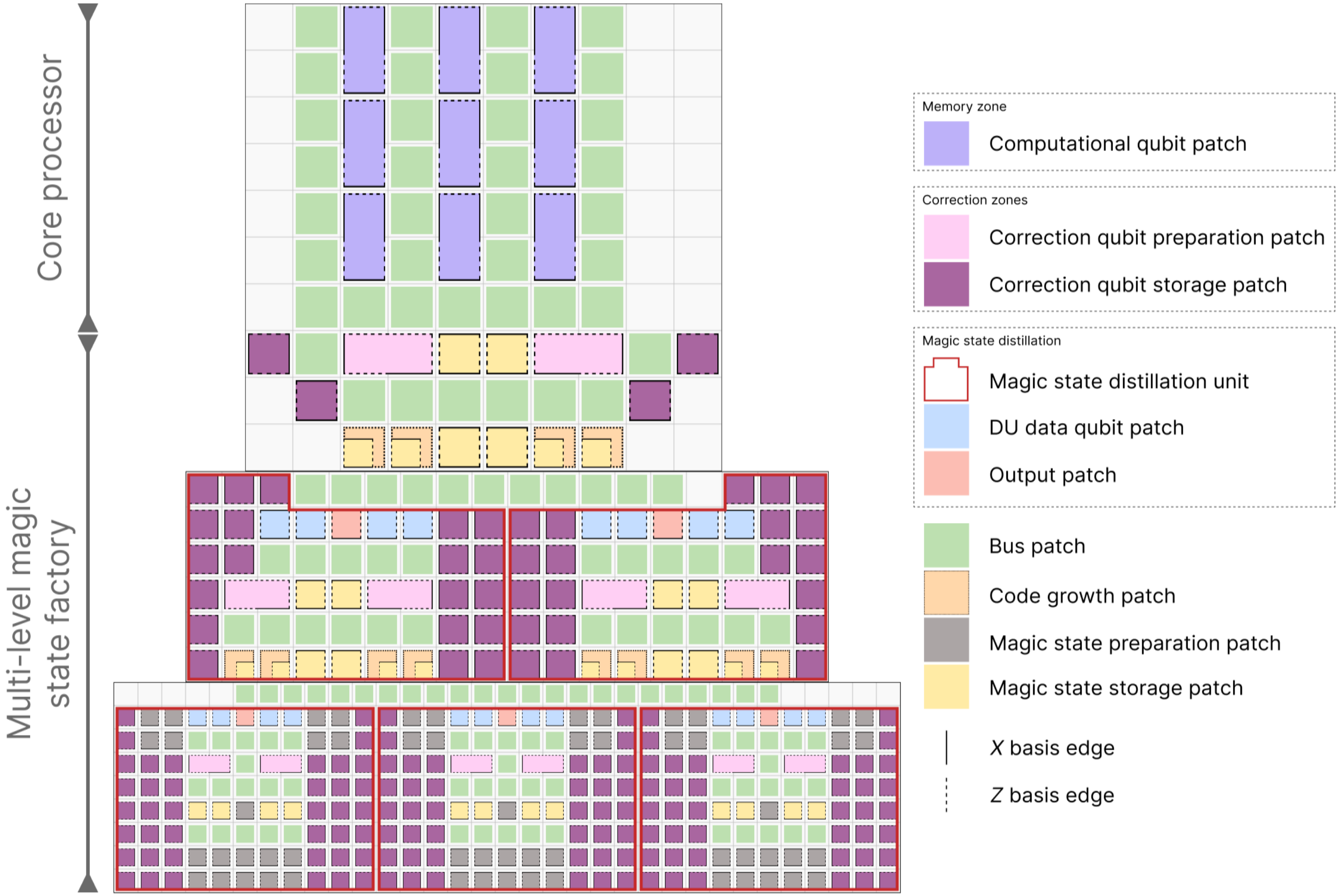
1QBit

Design and optimization of an FTQC architecture

Allyson Silva (*25 min*)

Senior Scientist, 1QBit

FTQC Microarchitecture Design



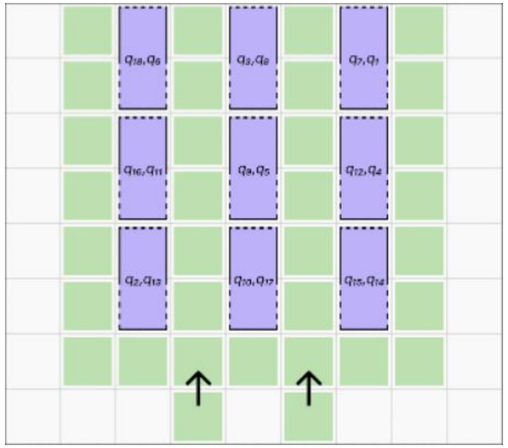
1QBit

Quantum Resource Estimation Service Steps

QRE Execution Steps

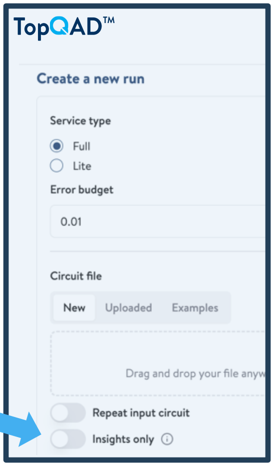
- **Synthesizing**
  - **Decomposing into Clifford+T gates**
  - **Removing Clifford gates and converting to Pauli product rotation gates**
- **Scheduling**
  - **Parsing rotations**
  - **Scheduling operations**
- **Noise profiling**
- **Assembling**
- **Generating report**

→ Core processor layout



Magic state factory

→ Scheduling data



To output detailed scheduling data (e.g., qubits used per step), toggle this

Time step	Num. operations scheduled	Lattice surgery size (B)
1	1	[8]
2	1	[6]
3	2	[14,6]
4	1	[17]
5	2	[12,12]
…	…	…

1QBit

Quantum Resource Estimation Service Steps

QRE Execution Steps

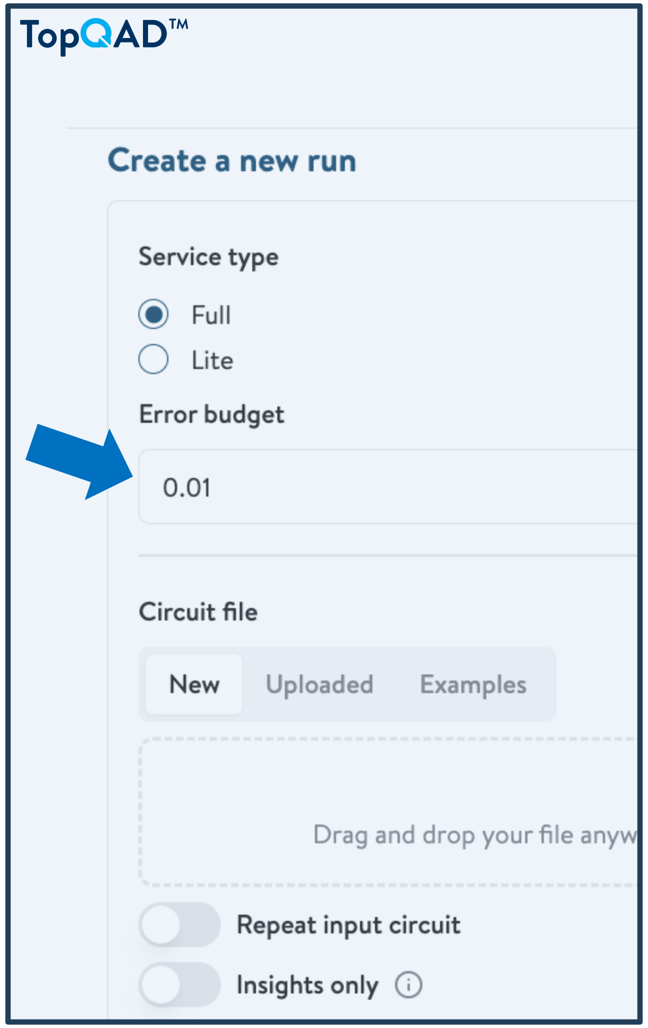
- **Synthesizing**
  - **Decomposing into Clifford+*T* gates**
  - **Removing Clifford gates and converting to Pauli product rotation gates**
- **Scheduling**
  - **Parsing rotations**
  - **Scheduling operations**
- **Noise profiling**
- **Assembling**
- **Generating report**

Problem:
Design a quantum microarchitecture

Biobjective minimization:
- Execution time
- Physical resource usage

Subject to:
- User-defined error budget
- Compiled logical schedule and core layout

Error Budget and Sources of Noise



FTQC requires an error budget, i.e., a user-defined threshold on the total acceptable probability of failure

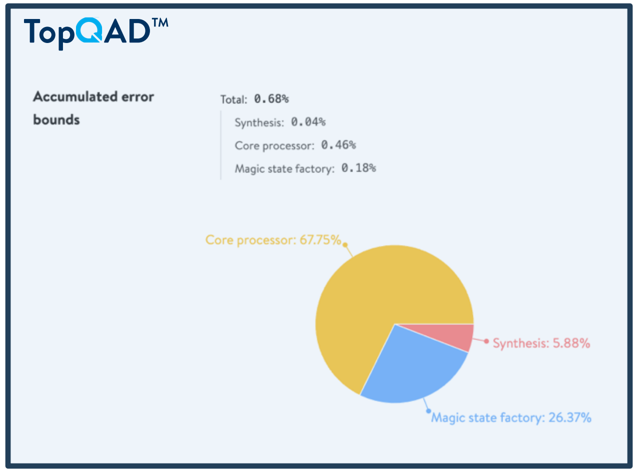
$$\boldsymbol{E \geq E_{\mathrm{synth}} + \sum_i E_{\mathrm{mod},i}}$$

E: error budget

E_{synth}: accumulated error with circuit synthesis (from compilation)

$E_{\mathrm{mod},i}$: accumulated error from FTQC operations of the invoked modules (e.g., core processor, magic state factory hierarchy, QROM)

Available modules	Noise sources	Model
Core processor	Idling computational qubits	e_{mem}
	Lattice surgeries	e_{surg}
Magic state factory	Magic state distillation	e_{msf}
	Idling correction qubits	e_{mem}
	Lattice surgeries	e_{surg}



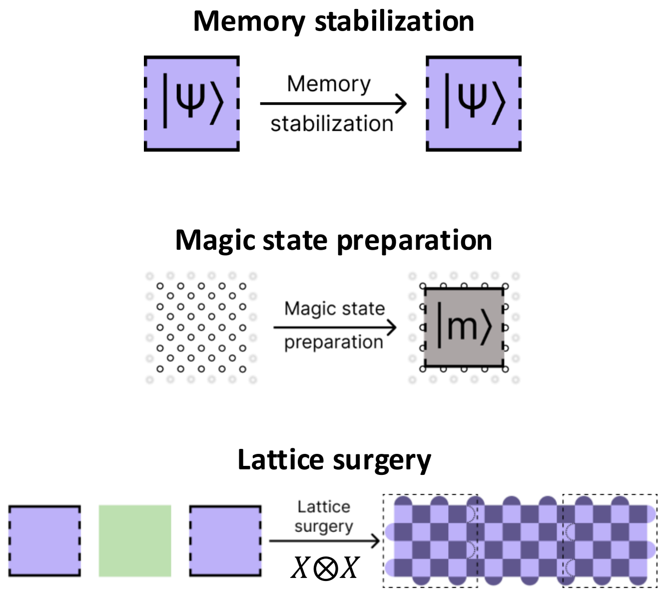
1QBit

Quantum Resource Estimation Service Steps

QRE Execution Steps

- ✓ **Synthesizing**
  - ✓ **Decomposing into Clifford+*T* gates**
  - ✓ **Removing Clifford gates and converting to Pauli product rotation gates**
- ✓ **Scheduling**
  - ✓ **Parsing rotations**
  - ✓ **Scheduling operations**
- ✓ **Noise profiling**
- ✓ **Assembling**
- ✓ **Generating report**

Projects the performance of fault-tolerant protocols
(more details in Session 2)



Protocol	Metric	Performance model
Memory stabilization	Logical error rate	$e_{\text{mem}} = f(d)$
	Reaction time	$\gamma_{\text{mem}} = f(d)$
Magic state preparation	Logical error rate	$e_{\text{prep}} = f(d)$
	Discard rate	$A_{\text{prep}} = f(d)$
Lattice surgery	Logical error rate	$e_{\text{ls}} = f(d, B)$
	Reaction time	$\gamma_{\text{ls}} = f(d)$

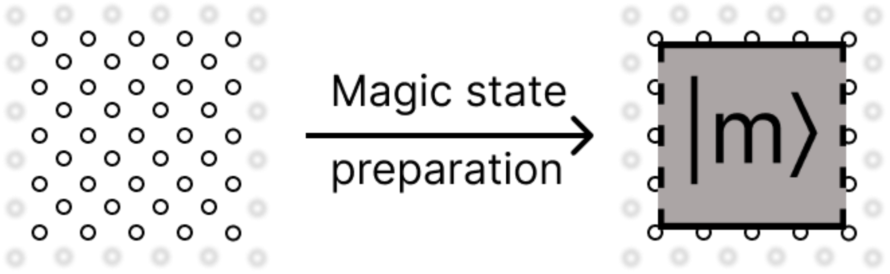
1QBit

Magic State Factory

What if we prepare logical magic states and feed directly to the core processor?

(see “baseline” data in Fig. 22 of
Mohseni, M. et al.,,arXiv:2411.10406 (2025))

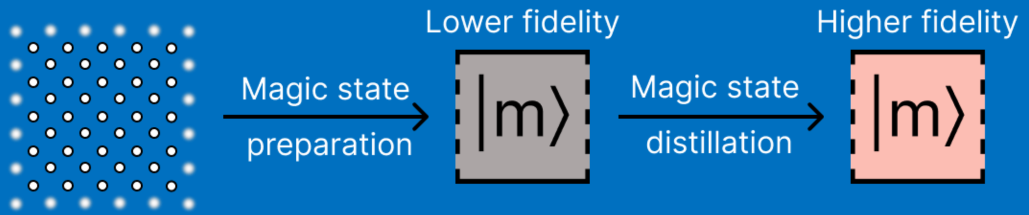
In a **state-of-the-art** magic state preparation protocol and hardware $e_{\mathrm{msf}} = e_{\mathrm{prep}} = 5 \times 10^{-4} = \mathbf{0.05}\%$:



Magic states prepared	Lower bound to accumulated error rate (E_{msf})
10^0	0.05%
10^1	0.5%
10^2	4.9%
10^3	39.3%
10^4	99.3%
10^5	~100%
10^6	~100%

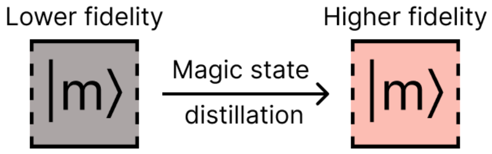
Smallest utility-scale algorithms are currently around here

Solution: magic state distillation



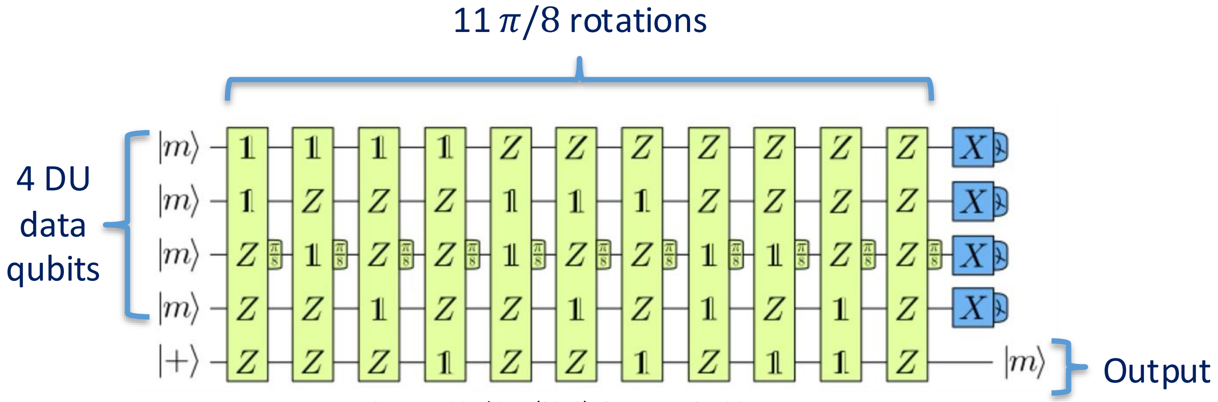
1QBit

Magic State Distillation



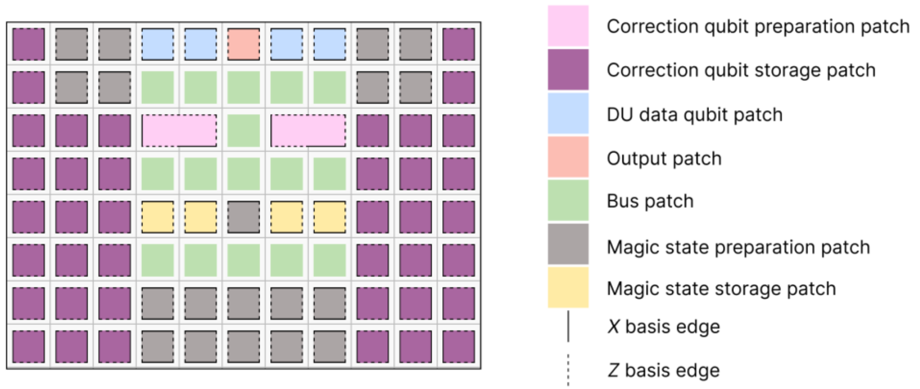
Execute a **distillation protocol** using a **distillation unit** (DU)

Example: 15-to-1 distillation protocol for T states



Source: Litinski, D. (2019), Quantum, 3, 128.

Time-optimal 15-to-1 DU layout



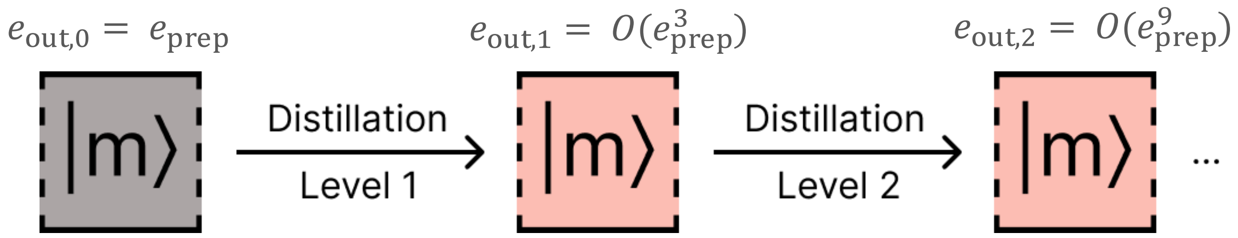
- There is a probability of success A_{prep} for magic state distillation (acceptance probability in TopQAD™
- Output magic state error rate (in 15-to-1 protocol): $e_{\text{out}} = O(e_{\text{in}}^3, e_{\text{mem}}, e_{\text{ls}})$

Note: The errors rates $\boldsymbol{e}_{\text{mem}} + \boldsymbol{e}_{\text{ls}} \to \mathbf{0}$, when $\boldsymbol{d} \to \infty$

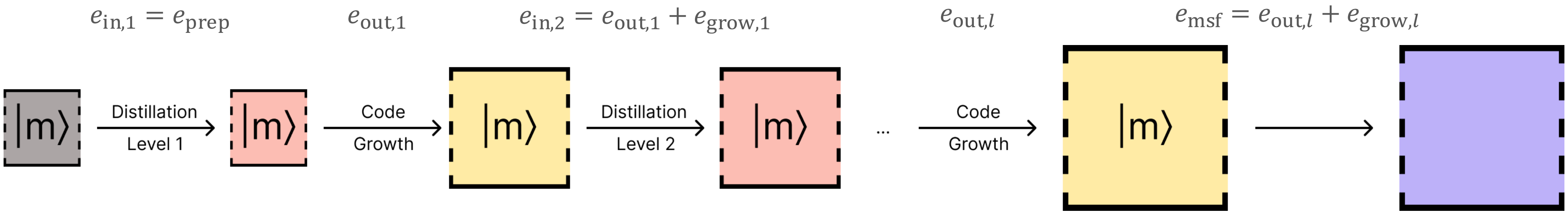
1QBit

Multi-level Magic State Factory

- Since fidelity improvement is bounded in a distillation round, perform multiple distillation rounds to keep improving fidelities



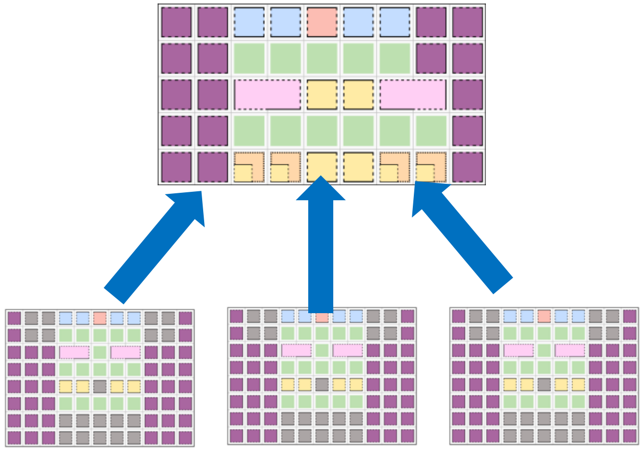
- To save space, different code distances may be used for distillation levels and the code processor



1QBit

Multi-level MSF with Parallel Distillation Units

- Running multiple DUs in parallel maintains throughput balance between distillation levels



Magic state factory

Distillation levels: 2

	Distillation protocol	Number of distillation units	Distillation runtime	Acceptance probability	Output magic state error rate
Distillation level 1	15:1	86	37.8 μs	82.58%	5.15×10^{-04}
Distillation level 2	15:1	14	105 μs	99.23%	1.55×10^{-08}

TopQAD™

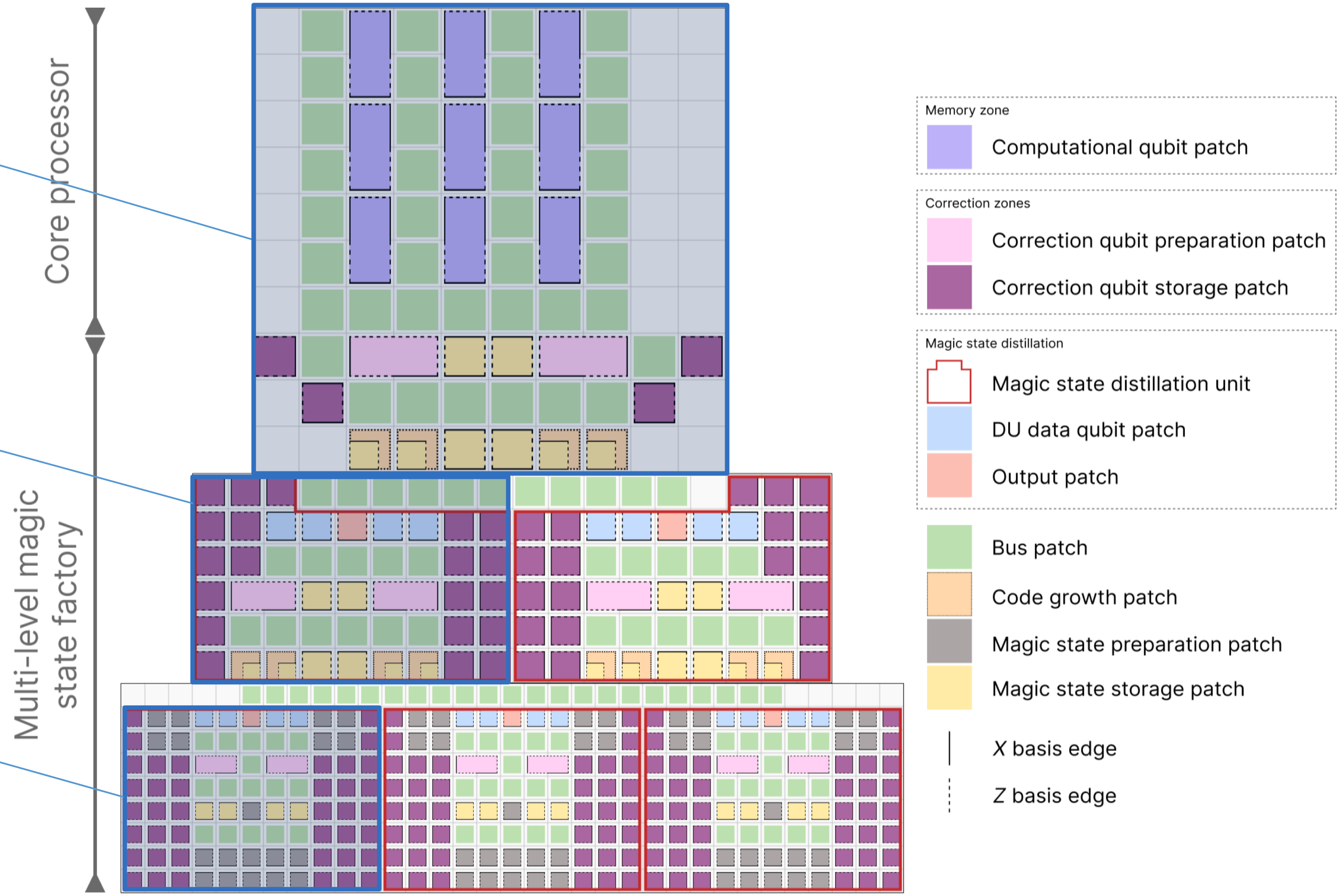
1QBit

Full FTQC Microarchitecture Design

Area designed to:
- Execute any circuit composed of $\pi/8$ rotations
- Prepare and store correction states
- Grow code patches

Area designed to:
- Execute a distillation protocol composed of $\pi/8$ rotations
- Prepare and store correction states
- Grow code patches

Area designed to:
- Execute a distillation protocol composed of $\pi/8$ rotations
- Prepare and store correction states
- Prepare magic states

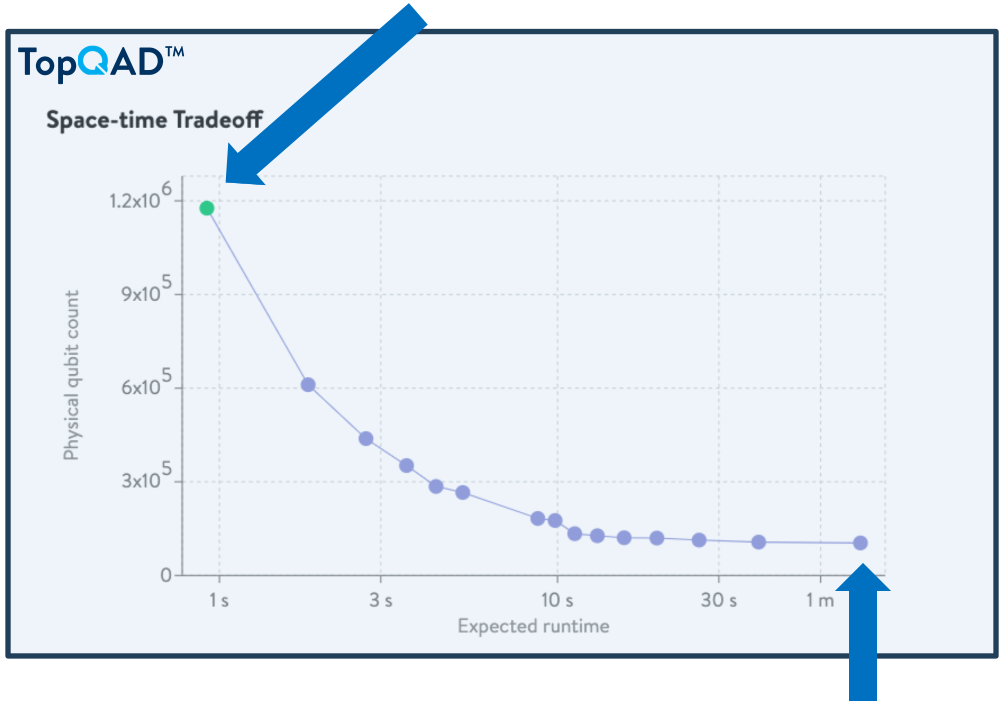


Optimizing the Microarchitecture

TopQAD™ uses optimization models to make several decisions for the architecture, including:

- Number of **preparation units** feeding the magic state factory
- Number of **distillation levels**
- Number and layout of **distillation units per level**, including the correction area
- Number and layout of **correction units** feeding the core processor
- Layout of the **core processor** (e.g., compact, fast, parallelizable)
- **Code distances** at each distillation level and the core processor

Time-optimal scenario:
Magic state distillation rate = magic state consumption rate

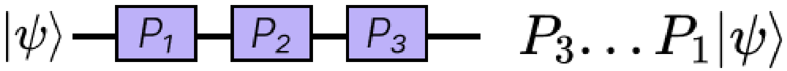


Space-optimal scenario:
Minimum viable MSF

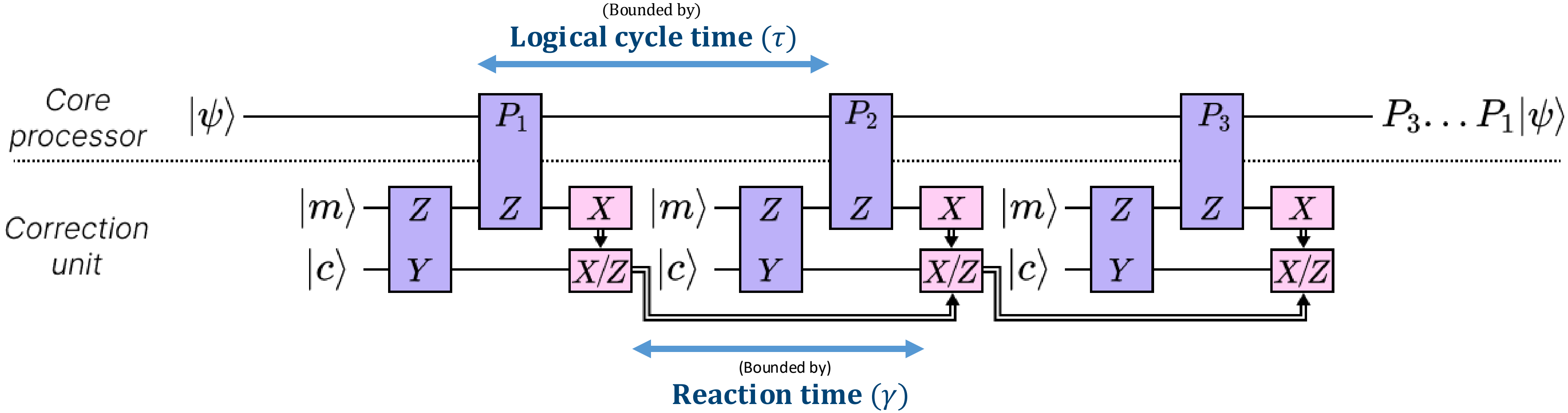
1QBit

Magic State Consumption Rate

Given a serial schedule of π/8 rotations:



Full circuit to execute it:



When $\gamma < \tau$:
1 magic state consumed per logical cycle

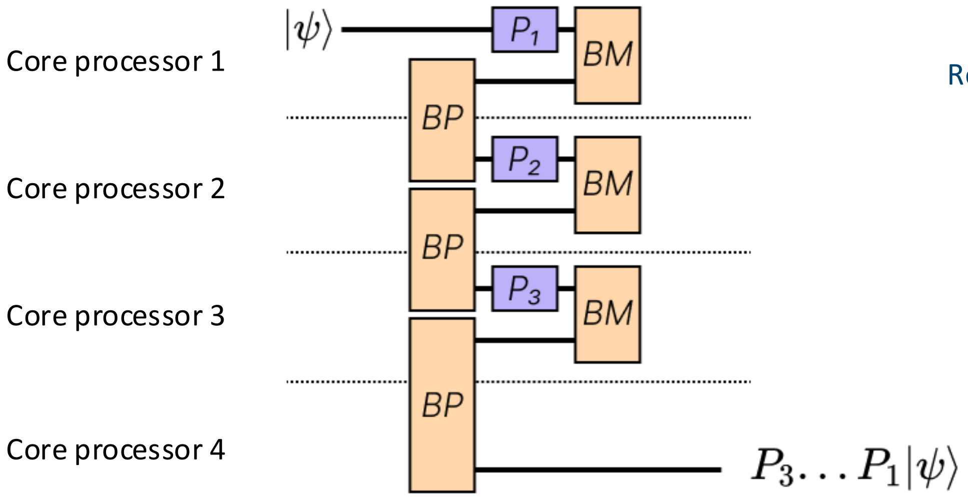
When $\gamma > \tau$:
1 magic state consumed per reaction time

1QBit

Implications of Magic State Consumption Rate

Slow reaction time	Fast reaction time
Fewer space–time trade-offs	More space–time trade-offs
Expected runtime $\approx \gamma \times$ number T gates	Expected runtime $\approx$ ~~$dW \times$ number T gates~~
	$\gamma \times$ number T gates

Special case: if reaction time is "fast enough", there is a trick to speed up the computation below the 1 magic state per logical cycle rate using quantum teleportation, i.e., through Bell pairs creation (BP) and measurements (BM), between multiple core processors



Reaction-time-limited computation (RTL)

Space-Time Optimal Architectures

Architecture	Expected Runtime	Physical Qubit Count	Core Code Distance	MSF Code Distance	# Distillation Units
RTL	14.81 h	17,225,064	33	[11, 29]	[141, 29]
2	1.26 d	2,222,014	31	[11, 29]	[70, 15]
3	2.49 d	1,278,974	31	[11, 29]	[35, 8]
4	3.62 d	928,798	31	[11, 29]	[24, 5]
5	4.83 d	780,582	31	[11, 29]	[18, 4]
6	6.21 d	659,358	31	[11, 29]	[14, 3]
7	7.91 d	646,470	31	[11, 31]	[11, 3]
8	9.67 d	544,958	33	[11, 27]	[9, 2]
9	10.87 d	531,462	33	[11, 27]	[8, 2]
10	12.43 d	517,966	33	[11, 27]	[7, 2]
11	14.5 d	504,470	33	[11, 27]	[6, 2]
12	16.41 d	444,733	33	[11, 27]	[6, 1]
13	17.4 d	431,237	33	[11, 27]	[5, 1]
14	21.75 d	417,741	33	[11, 27]	[4, 1]
15	29.0 d	404,245	33	[11, 27]	[3, 1]
16	43.5 d	390,749	33	[11, 27]	[2, 1]
17	87.0 d	384,628	33	[11, 29]	[1, 1]

Interaction with the TopQAD portal

Allyson Silva and Katie Olfert (*15 min*)

1QBit

1QBit

Redefine Intractable

TopQAD™ Tutorial

IEEE Quantum Week 2025

Evaluate and Design Quantum Computers: Automated FTQC Architecture Design and Resource Estimation Using TopQAD

Session 1 recap

Allyson Silva and Katie Olfert (*15 min*)

TopQAD Portal: The Browser Experience

https://topqad.1qbit.com

Trial Code: **********

Sign-up Checklist

- ✅ Visit topqad.1qbit.com and follow the "Portal" link
- ☑ Sign up -> enter desired account details
- ☑ Verify email
- ☑ Enter trial code
- ☑ Accept terms and conditions
- ☑ Authenticate with MFA

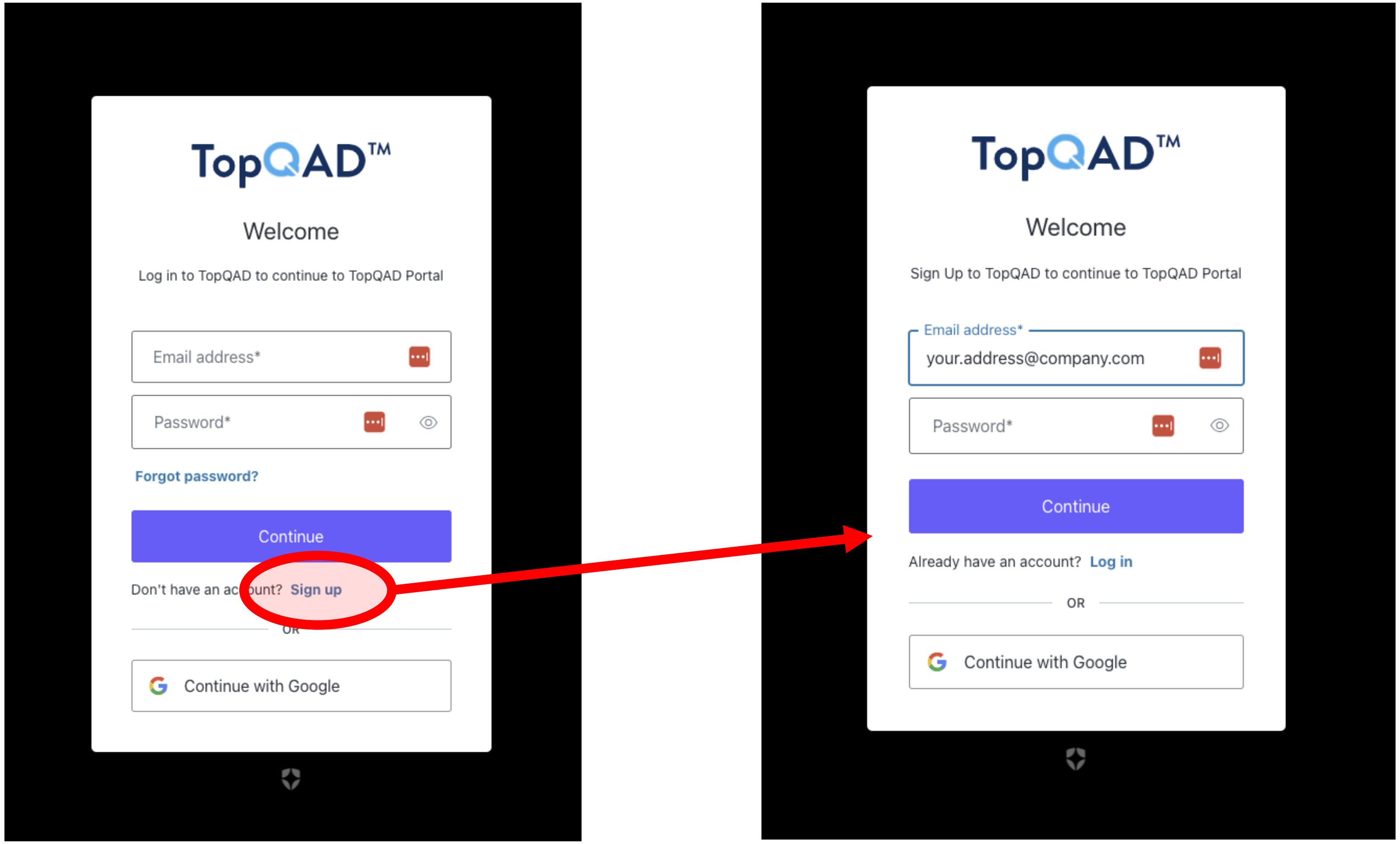
Create a TopQAD Account

https://topqad.1qbit.com

Trial Code: **********

Sign-up Checklist

- ✅ Visit topqad.1qbit.com and follow the “Portal” link
- ✅ Sign up -> enter desired account details
- ☑ Verify email
- ☑ Enter trial code
- ☑ Accept terms and conditions
- ☑ Authenticate with MFA



1QBit

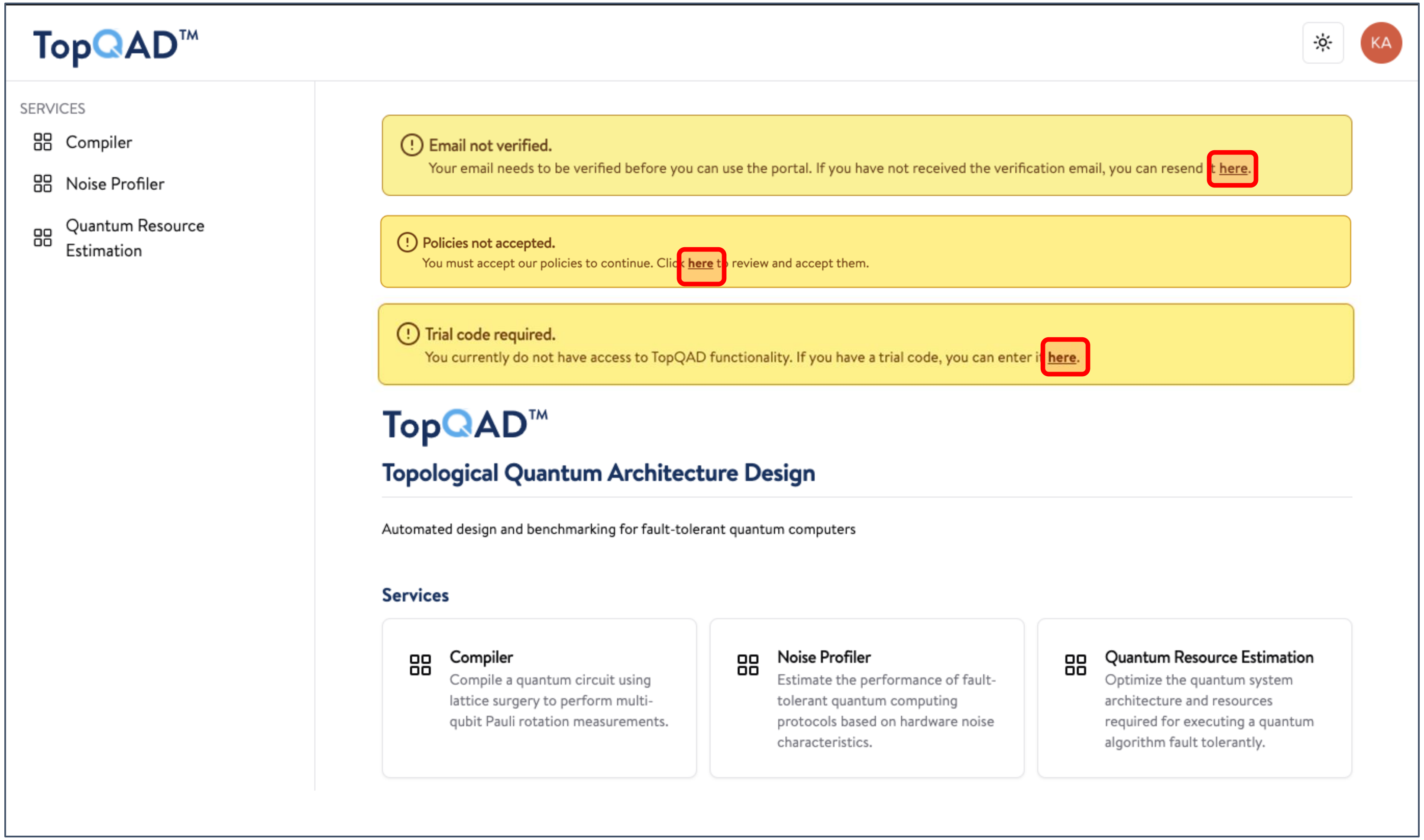
Verify Email

https://topqad.1qbit.com

Trial Code: **********

Sign-up Checklist

- ✅ Visit topqad.1qbit.com and follow the “Portal” link
- ✅ Sign up -> enter desired account details
- ✅ Verify email
- ☑ Enter trial code
- ☑ Accept terms and conditions
- ☑ Authenticate with MFA



1QBit

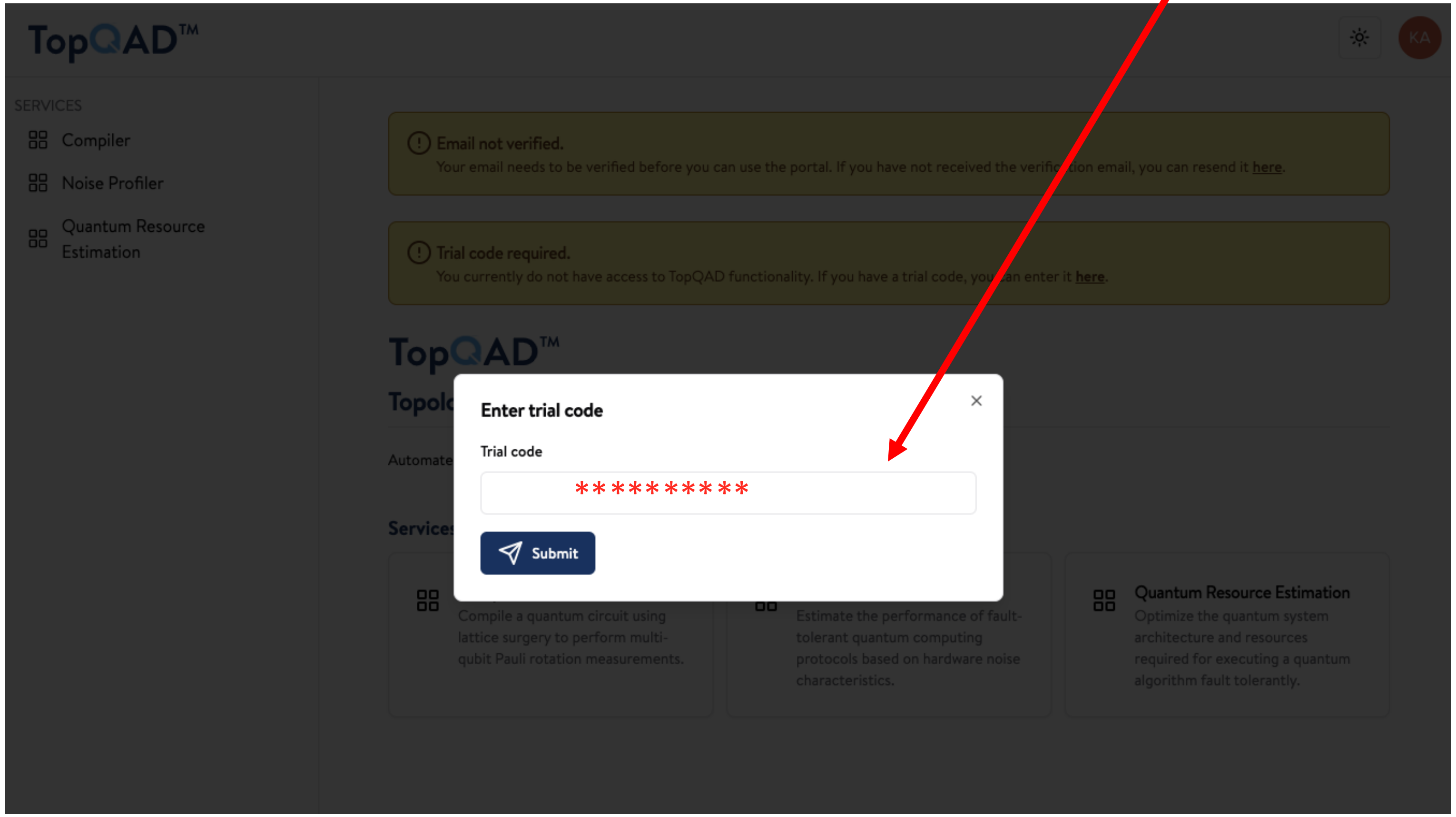
Trial Code

https://topqad.1qbit.com

Trial Code: **********

Sign-up Checklist

- ✅ Visit topqad.1qbit.com and follow the "Portal" link
- ✅ Sign up -> enter desired account details
- ✅ Verify email
- ✅ Enter trial code
- ☑ Accept terms and conditions
- ☑ Authenticate with MFA



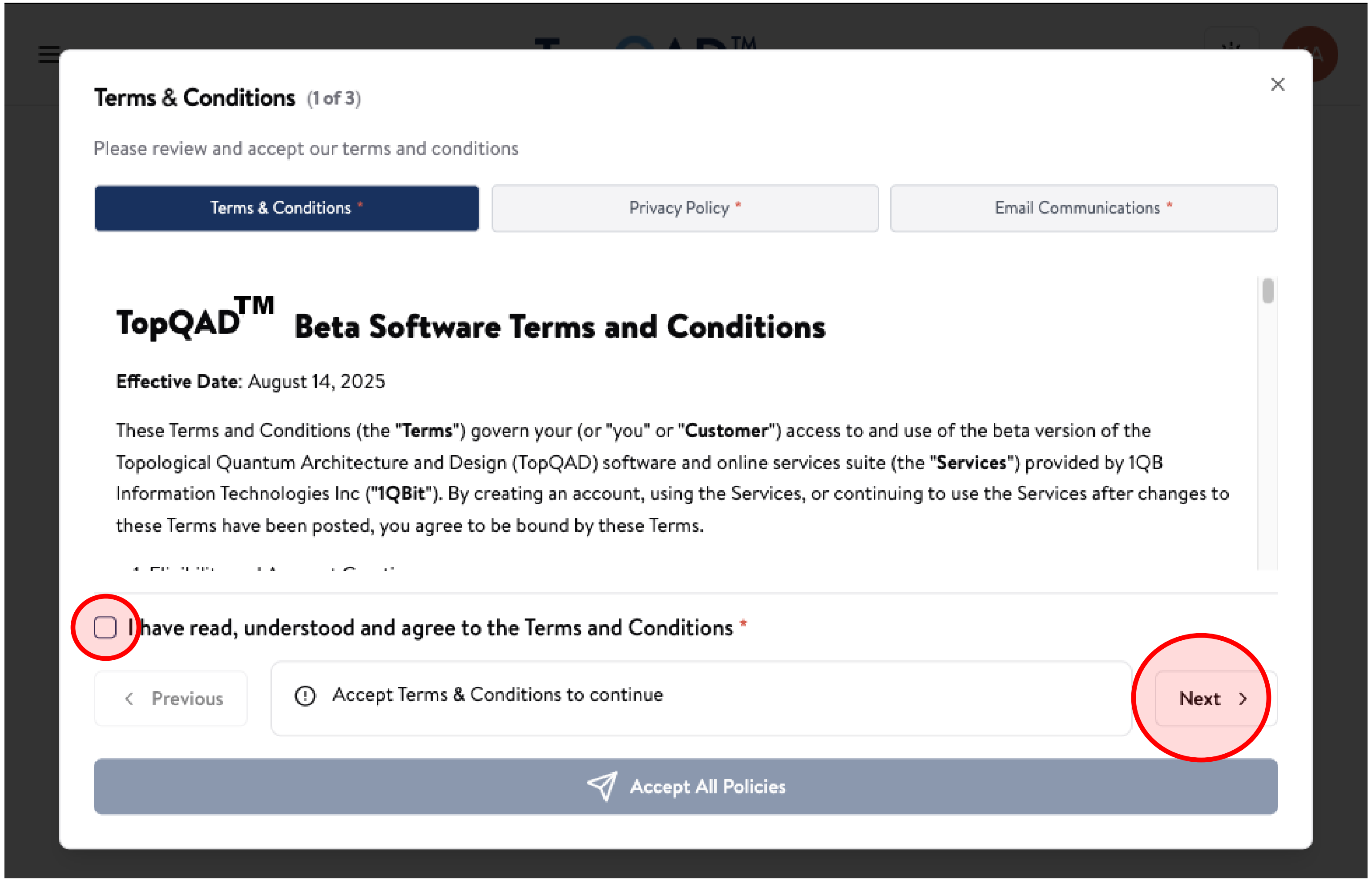
Terms and Conditions

https://topqad.1qbit.com

Trial Code: **********

Sign-up Checklist

- ✅ Visit topqad.1qbit.com and follow the "Portal" link
- ✅ Sign up -> enter desired account details
- ✅ Verify email
- ✅ Enter trial code
- ✅ Accept terms and conditions
- ☑️ Authenticate with MFA



1QBit

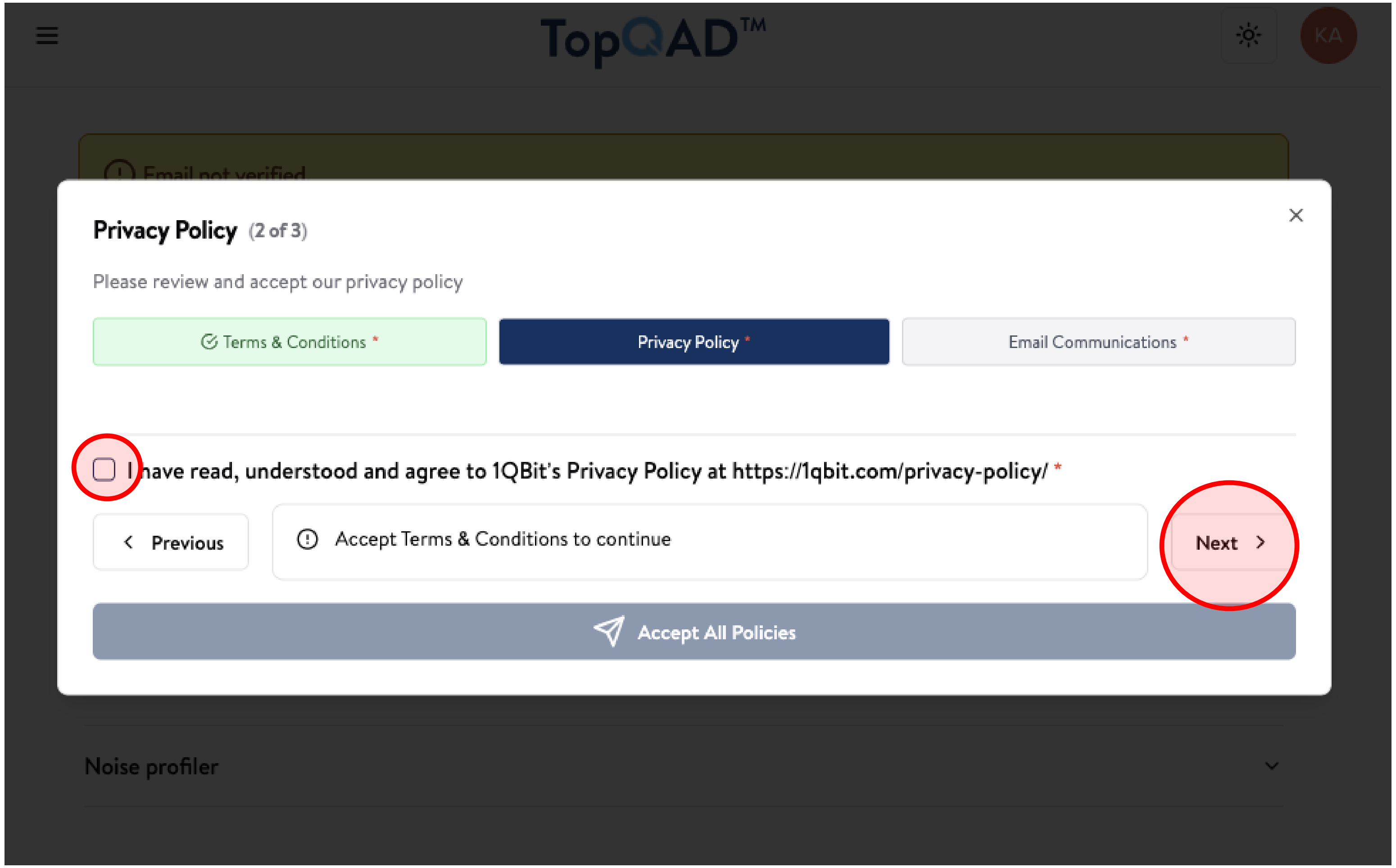
Terms and Conditions

https://topqad.1qbit.com

Trial Code: **********

Sign-up Checklist

- ✅ Visit topqad.1qbit.com and follow the “Portal” link
- ✅ Sign up -> enter desired account details
- ✅ Verify email
- ✅ Enter trial code
- ✅ Accept terms and conditions
- ☑ Authenticate with MFA



1QBit

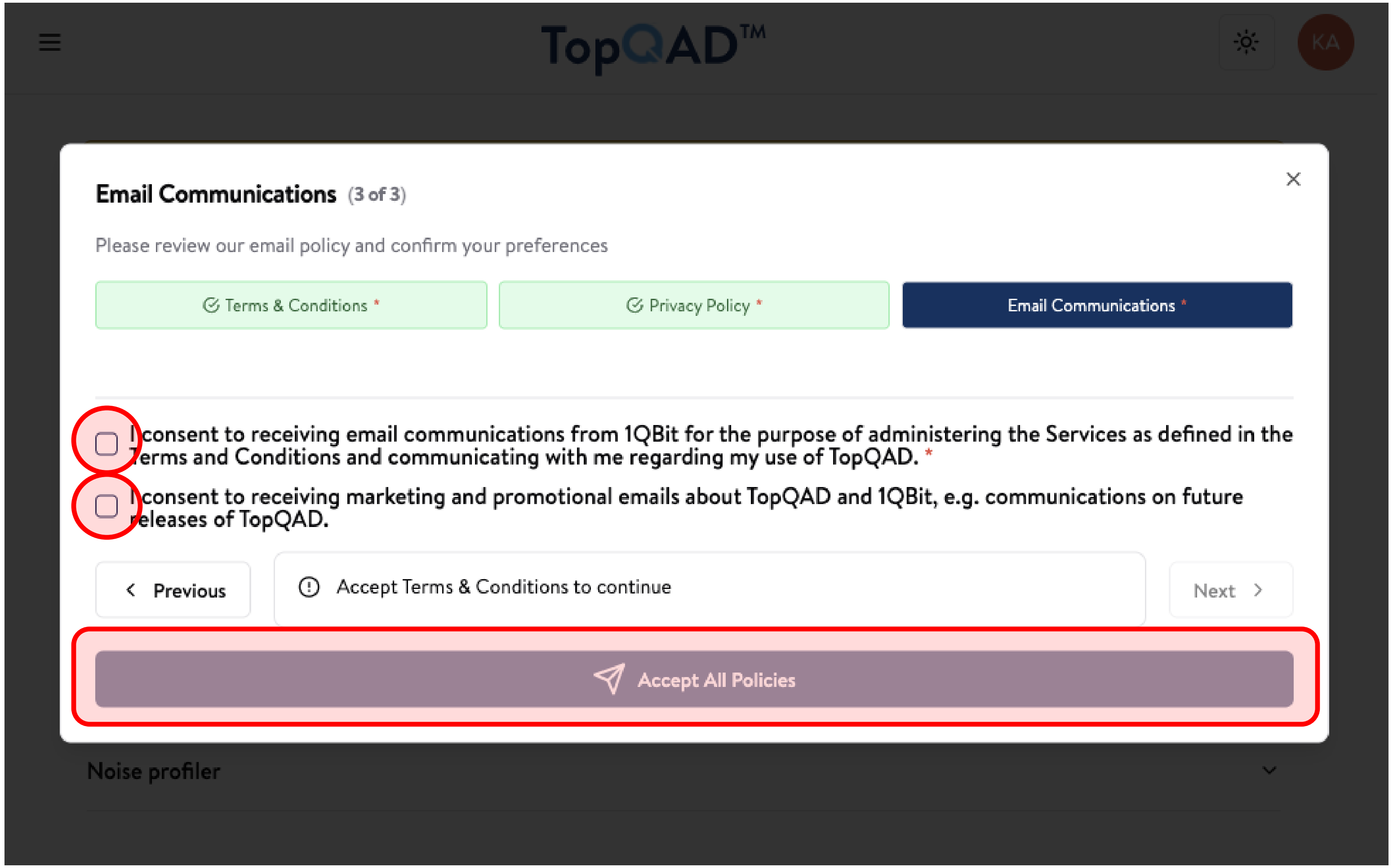
Terms and Conditions

https://topqad.1qbit.com

Trial Code: **********

Sign-up Checklist

- ✅ Visit topqad.1qbit.com and follow the “Portal” link
- ✅ Sign up -> enter desired account details
- ✅ Verify email
- ✅ Enter trial code
- ✅ Accept terms and conditions
- ☑ Authenticate with MFA



1QBit

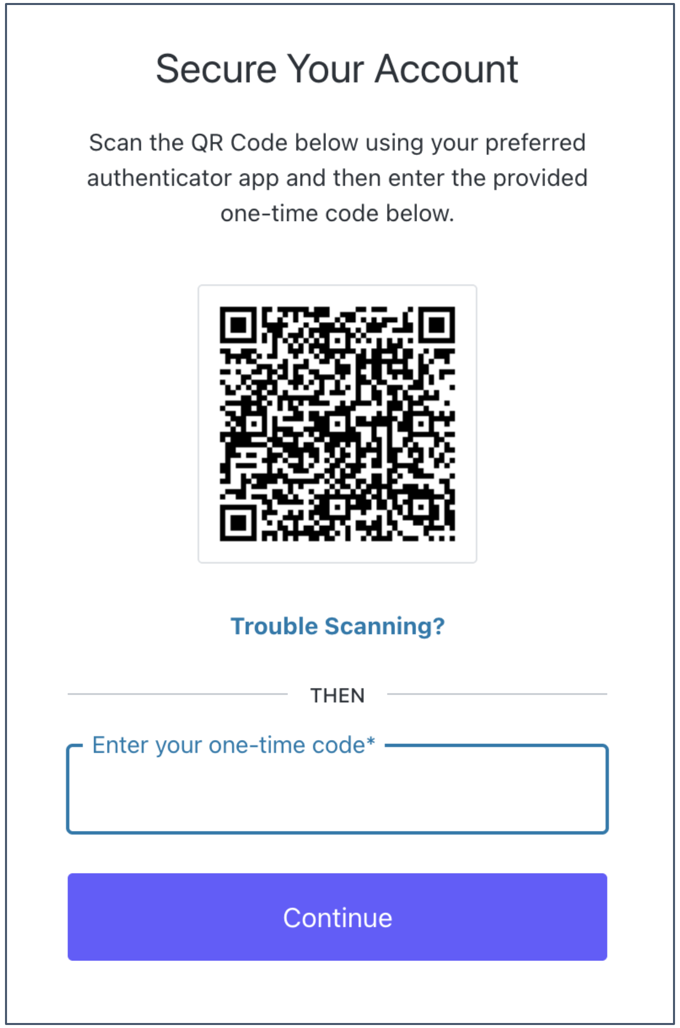
Multi-factor Authentication (MFA)

https://topqad.1qbit.com

Trial Code: **********

Sign-up Checklist

- ✅ Visit topqad.1qbit.com and follow the "Portal" link
- ✅ Sign up -> enter desired account details
- ✅ Verify email
- ✅ Enter trial code
- ✅ Accept terms and conditions
- ✅ Authenticate with MFA

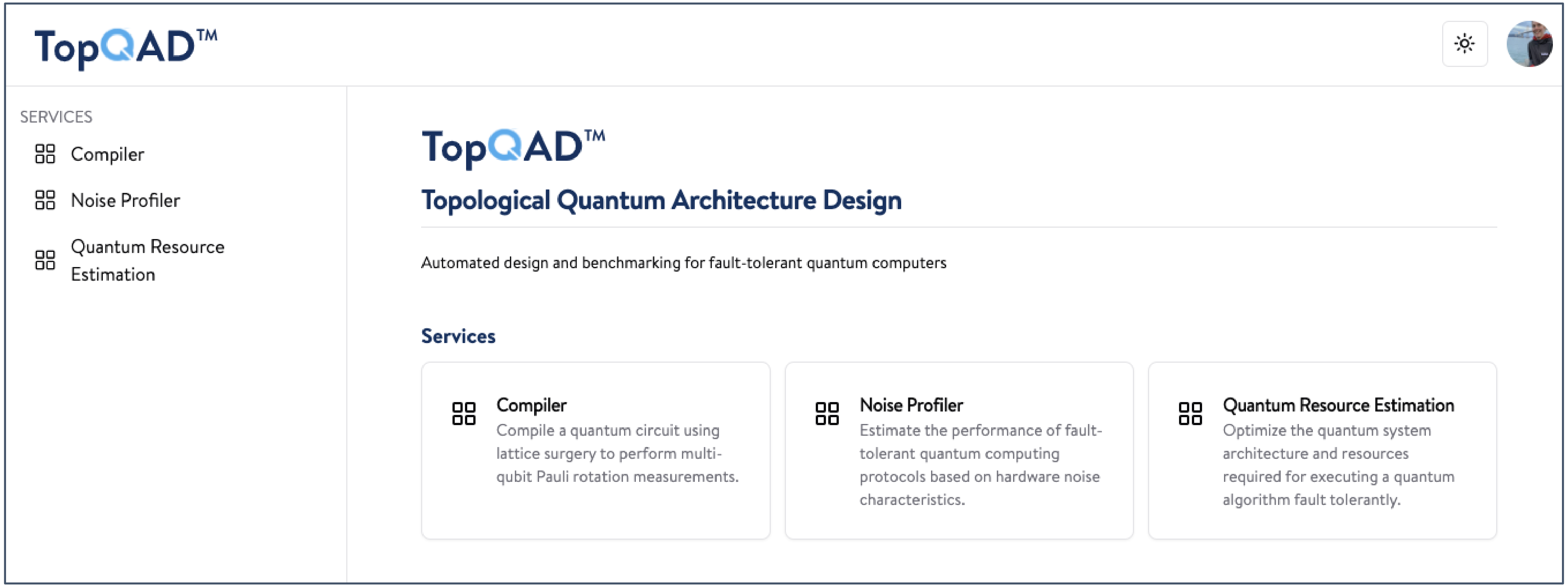


1QBit

Ready to Explore

https://topqad.1qbit.com

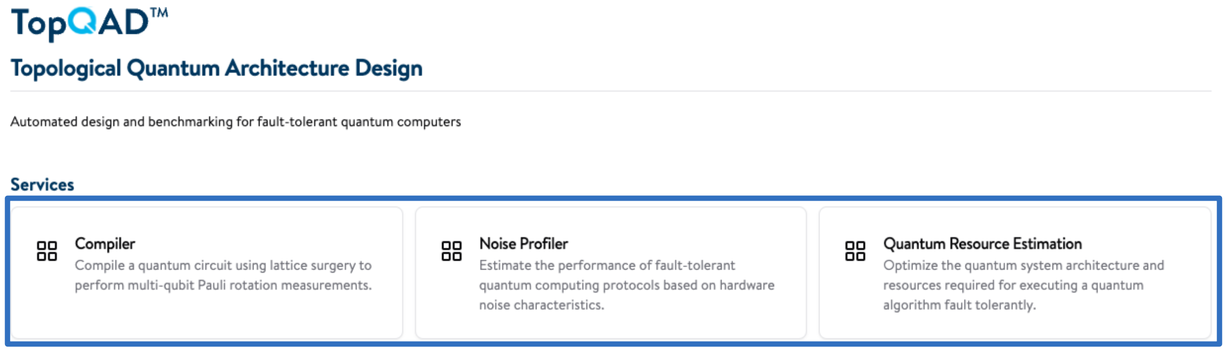
Trial Code: **********



Session 1 Recap

https://topqad.1qbit.com

Trial Code: **********


TopQAD™

Topological Quantum Architecture Design

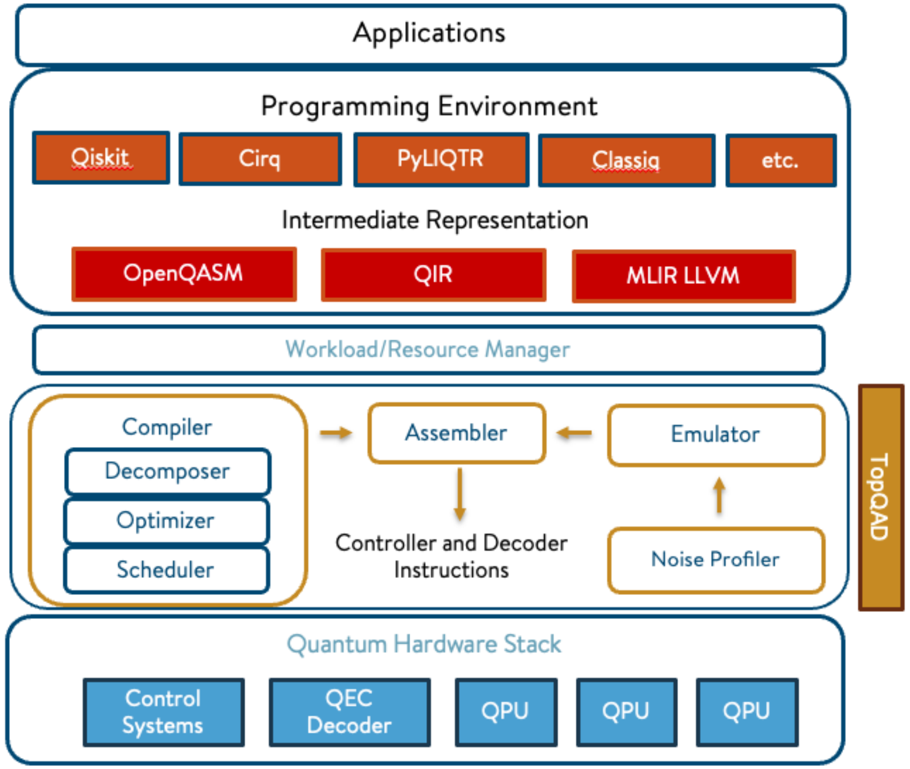
Automated design and benchmarking for fault-tolerant quantum computers

Services

Compiler
Compile a quantum circuit using lattice surgery to perform multi-qubit Pauli rotation measurements.

Noise Profiler
Estimate the performance of fault-tolerant quantum computing protocols based on hardware noise characteristics.

Quantum Resource Estimation
Optimize the quantum system architecture and resources required for executing a quantum algorithm fault tolerantly.



- TopQAD™ is a software suite that provides tools and services for automated design and benchmarking for fault-tolerant quantum computers.
- Behind the scenes, runs components of the quantum operating system of tomorrow. TopQAD™

1QBit

Session 1 Recap

https://topqad.1qbit.com

Trial Code: **********

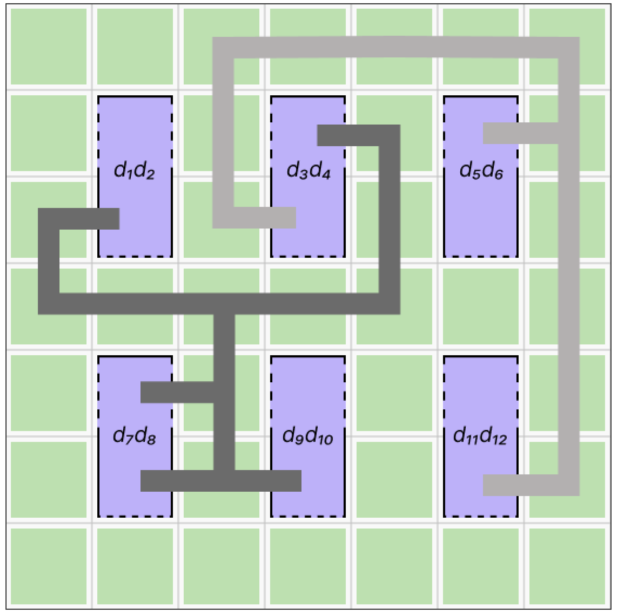
TopQAD™

Topological Quantum Architecture Design

Automated design and benchmarking for fault-tolerant quantum computers

Services

- **Compiler** — Compile a quantum circuit using lattice surgery to perform multi-qubit Pauli rotation measurements.
- **Noise Profiler** — Estimate the performance of fault-tolerant quantum computing protocols based on hardware noise characteristics.
- **Quantum Resource Estimation** — Optimize the quantum system architecture and resources required for executing a quantum algorithm fault tolerantly.



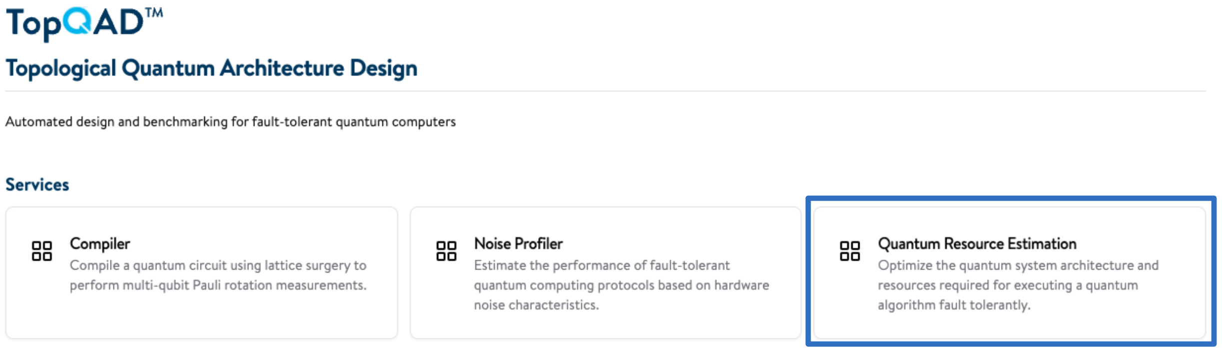
- Optimize the conversion of gates from a universal gate set into an instruction set architecture (ISA) for a given quantum error correction scheme (e.g., Pauli rotations for surface codes).
- Design a layout where the ISA will be executed in the quantum processor.
- Schedule the operations (e.g., lattice surgeries required to implement long-range qubit measurements) to produce an FTQC compiled program.

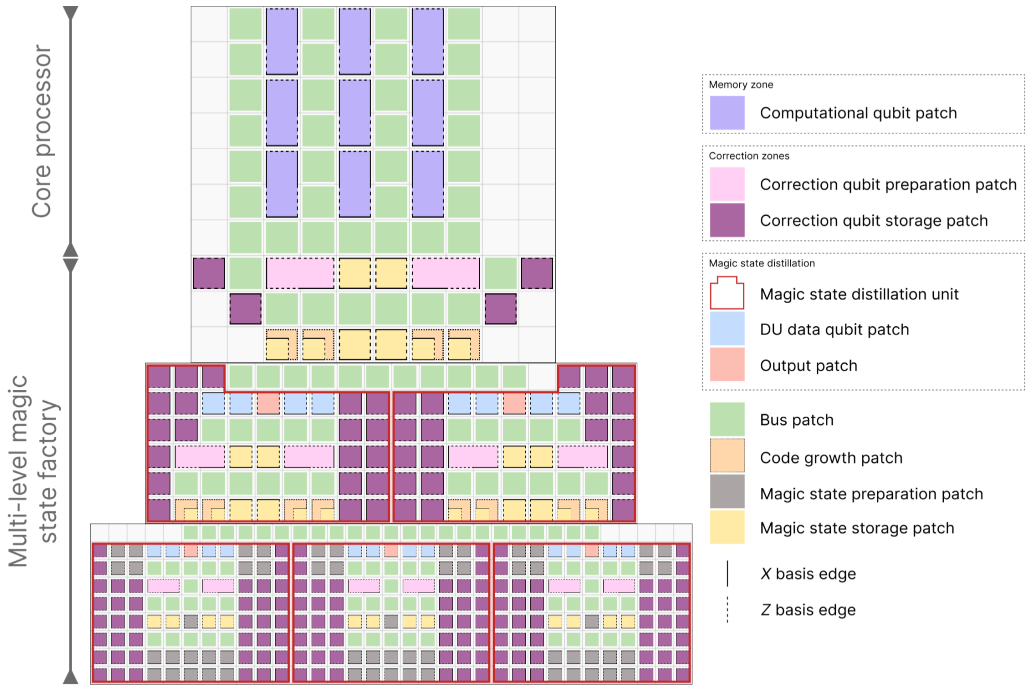
1QBit

Session 1 Recap

https://topqad.1qbit.com

Trial Code: **********





- Uses the FTQC compiled program and the noise models derived from emulations of several FTQC protocols to design a space–time efficient FTQC microarchitecture that meets a user-provided error budget.

Session 1 Recap

https://topqad.1qbit.com

Trial Code: **********

TopQAD™

Topological Quantum Architecture Design

Automated design and benchmarking for fault-tolerant quantum computers

Services

Compiler
Compile a quantum circuit using lattice surgery to perform multi-qubit Pauli rotation measurements.

Noise Profiler
Estimate the performance of fault-tolerant quantum computing protocols based on hardware noise characteristics.

Quantum Resource Estimation
Optimize the quantum system architecture and resources required for executing a quantum algorithm fault tolerantly.

Coming next

What are the FTQC protocols I should run in my applications?

How to emulate the FTQC protocols' performance for my target hardware

1QBit

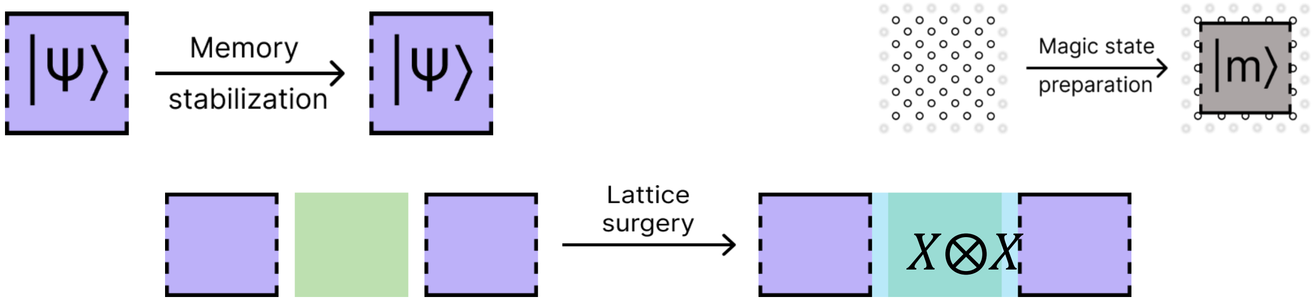
FTQC protocol performance

Abdullah Khalid (*30 min*)
Scientist, 1QBit

FTQC Protocol Simulations with TopQAD's Noise Profiler

FTQC Protocols

- Logical operations on logical qubits
- Leverage QEC techniques to minimize probability of logical errors



- Description given some parameters:

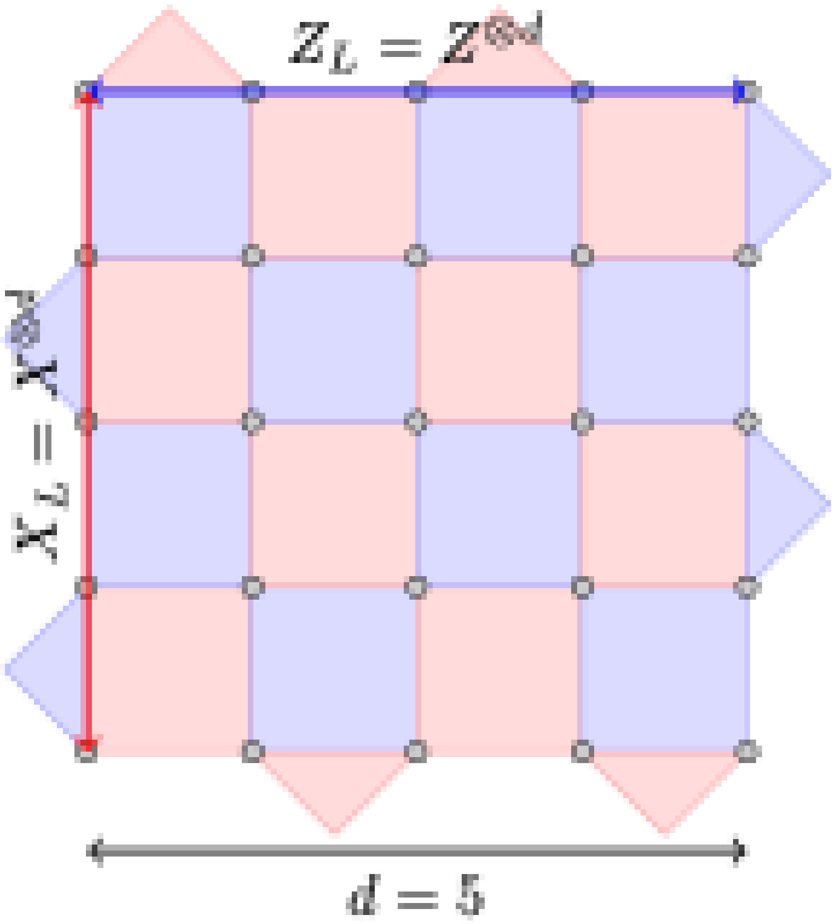
1. A quantum circuit operating on physical qubits
2. A decoder
3. Classical conditional logic based on measurement/decoder outcomes (optional)

Noise models

- describe how errors occur on a quantum chip

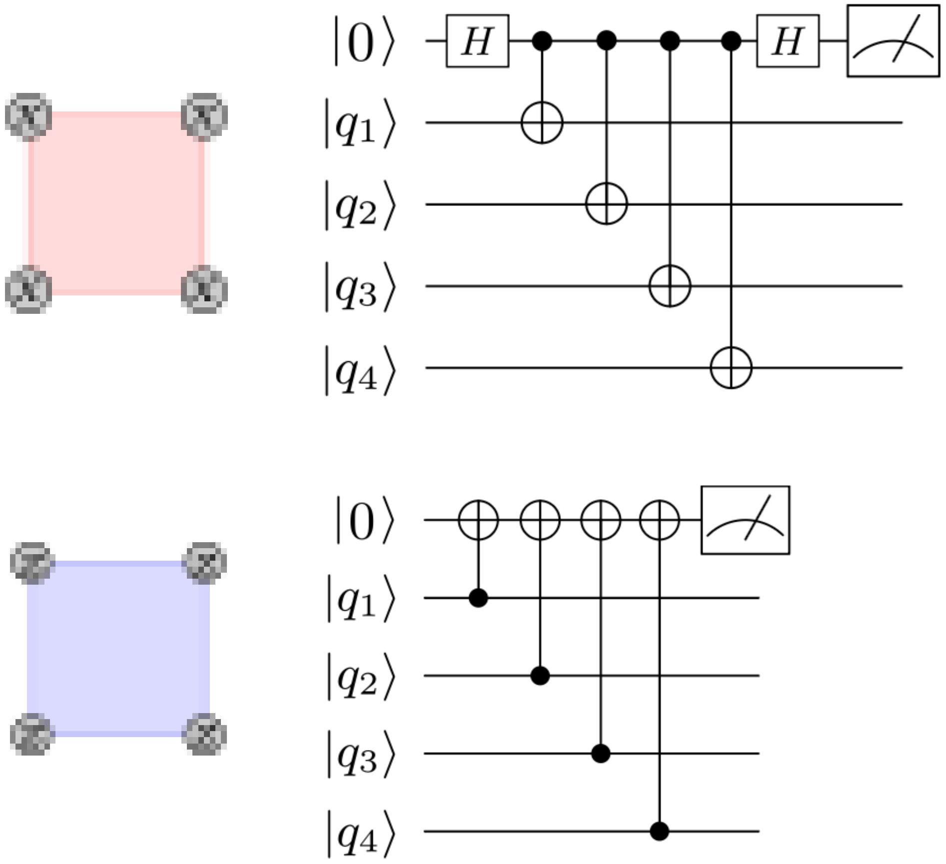
1QBit

Rotated Surface Codes



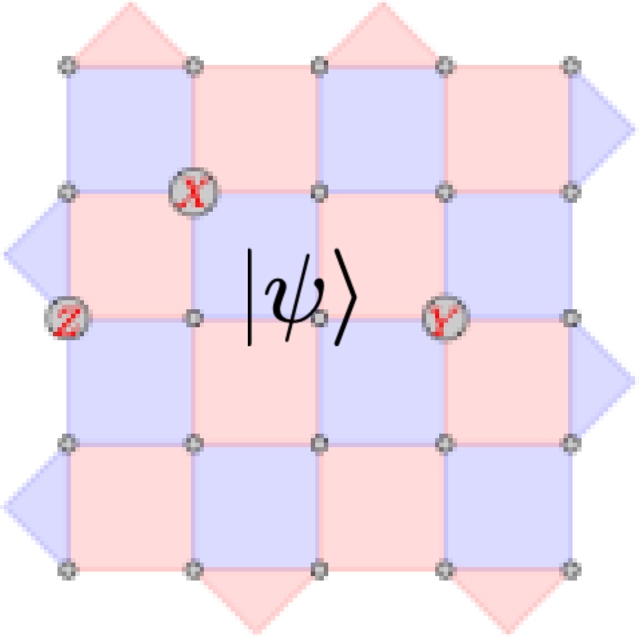
Distance of the code

Syndrome extraction circuits



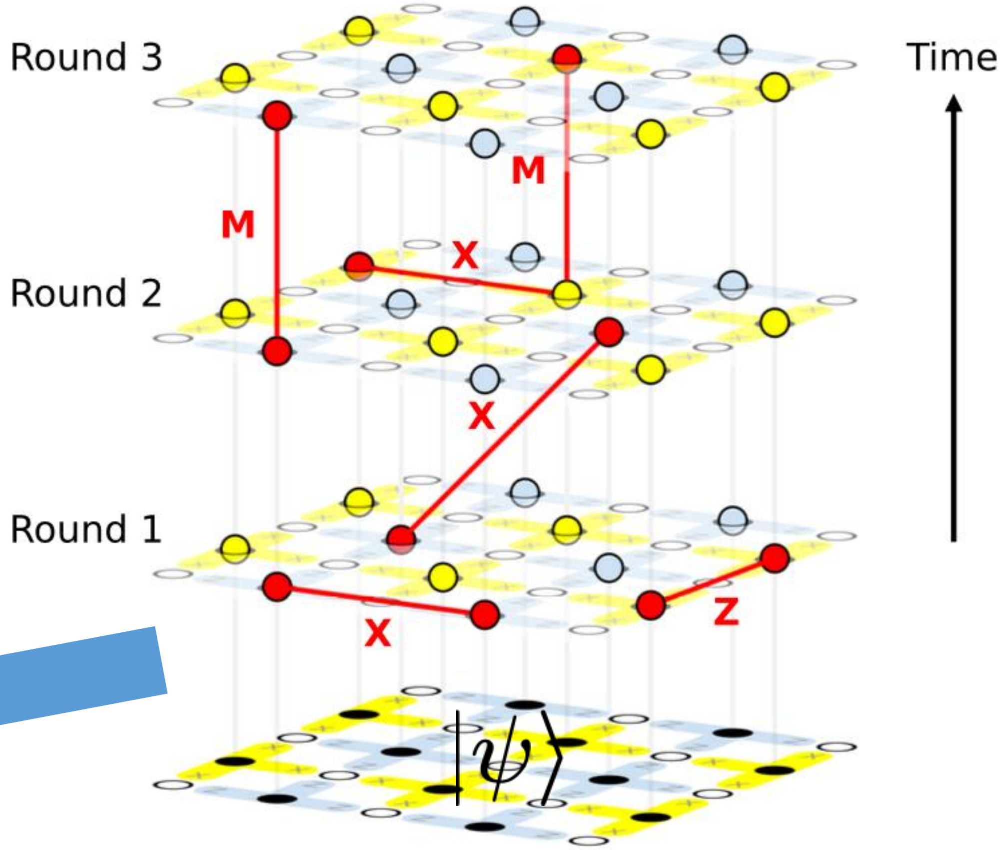
Idling Errors and Error Correction for Protection

Example data qubit errors



Iterative stabilizations to protect against data qubit errors

Now, also have syndrome qubit (measurement) errors



Wu, Y. et al., arXiv:2211.03288 (2022)

Sending syndromes to decoder

Decoder

Classical algorithm executed on classical coprocessor

Determines which errors occurred

1QBit

Quantum Memory Protocol

Purpose: protect idling logical qubits against noise

FTQC protocol description as a parameter set:

1. A quantum circuit operating on physical qubits
2. A decoder
3. ~~Classical conditional logic based on measurement/decoder outcomes (optional)~~
4. Validation in simulation: additional physical operations to determine if a logical error has occurred

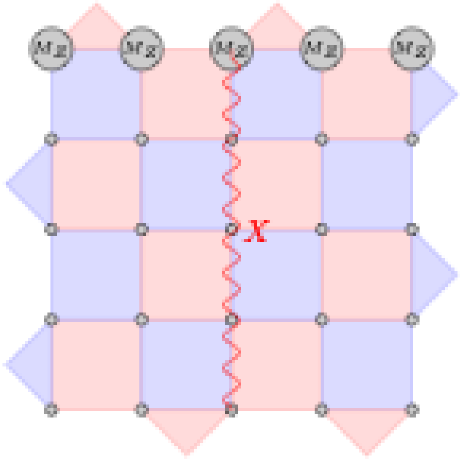
Parameter	Description
Distance (d)	Distance of the code
Rounds (r)	Number of stabilization rounds
basis	Which logical basis state to preserve: X or Z

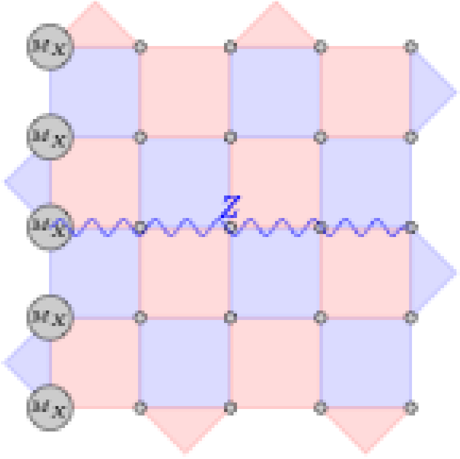
Decoder:

- TopQAD currently uses PyMatching
- More decoders will be added in the future

Circuit:

1. Start with a surface code patch of distance d
2. if basis == Z:
3. Prepare data qubits in $|0\rangle$
4. elif basis == X:
5. Prepare data qubits in $|+\rangle$
6. Do one stabilization round to prepare logical $|0\rangle$ or $|+\rangle$
7. Do $r - 1$ stabilization rounds
8. Measure data qubits
9. if basis == Z:
10. Compute parity of top row (–1 if X logical error)
11. elif basis == X:
12. Compute parity of left col (–1 if Z logical error)



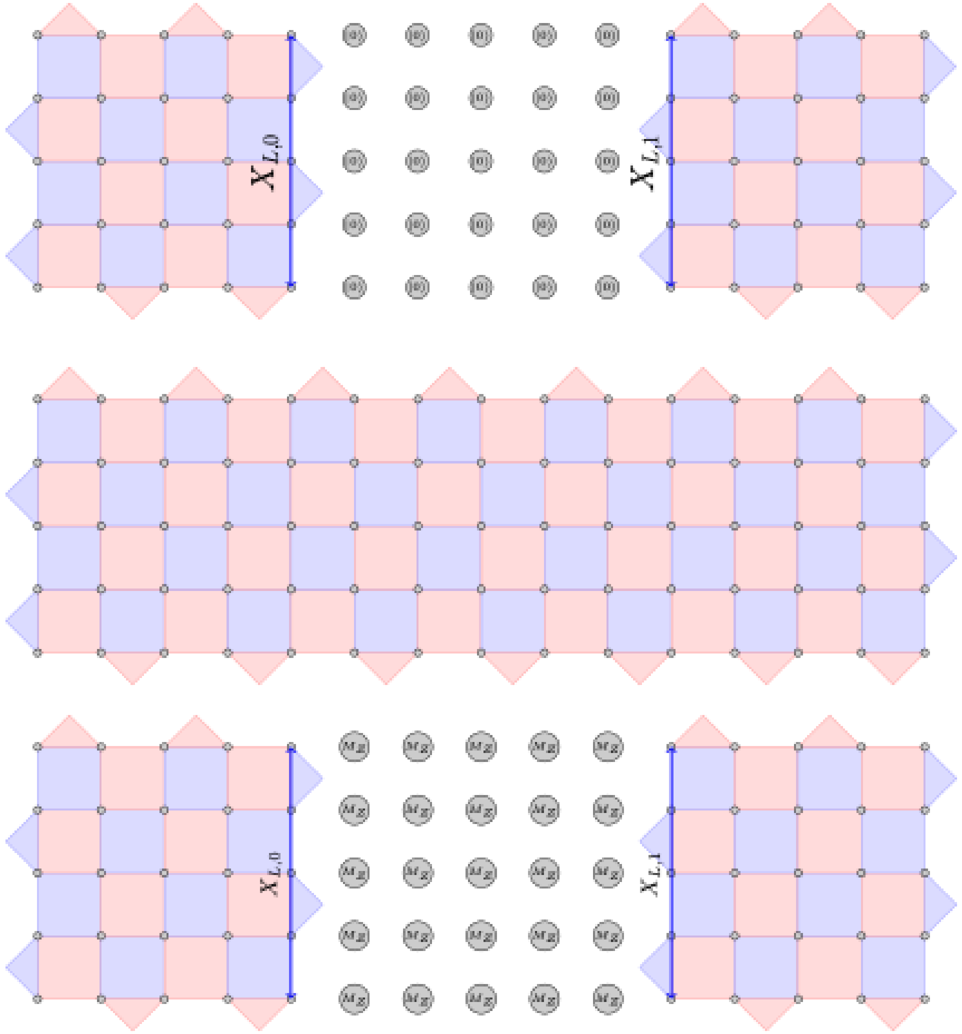


Lattice Surgery Protocol

<u>Purpose</u>:
1. Perform multi-qubit entangling operations
2. Move qubits

Parameter	Description
Distance (d)	Distance of the two codes
Bus width (b)	Number of data qubits cols. in bus
Rounds (r)	Number of stabilization rounds
Surgery type	Merging operators. XX or ZZ More coming to TopQAD
Preparation basis	Initial logical state of two qubits, e.g., (X, Z)
Measurement basis	Logical basis in which qubits are measured, e.g., (Z, X)
Logical observable	Which physical qubit parity to check?

XX surgery



1. Start with two surface code patches ($d = 3$) with a bus connecting them

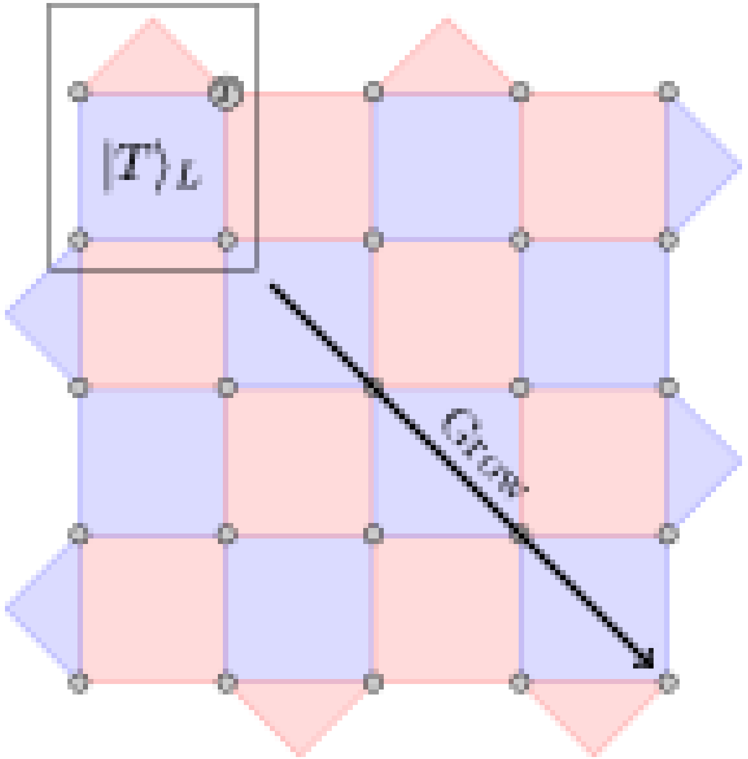
2. Perform r stabilization rounds

3. Measure bus qubits in Z (because XX surgery)

4. Measure other qubits for validation

1QBit

Magic State Preparation Protocols



Purpose: to create a logical magic state $$|T\rangle = \frac{|0\rangle + e^{\frac{i\pi}{4}}|1\rangle}{\sqrt{2}}$$

Circuit:

1. Prepare a few nearby qubits into the target state, constrained by nearest neighbour connectivity;
2. If errors are detected during this process, restart (classical conditional logic);
3. Otherwise, grow the state to the final desired distance.

TopQAD's current library of magic state preparation protocols:

1. Singh, S. et al., Phys. Rev. A 105, 052410 (2022),
2. Gidney, C., arXiv:2302.12292 (2023),
3. Gidney, C. et al., arXiv:2409.17595 (2024) *(coming soon).*

Parameter	Description
Distance 1 (d_1)	Initial patch distance
Distance 2 (d_2)	Final patch distance
Inject state	Which basis state to create, e.g., T, X, or Y basis states

1QBit

Simulators for FTQC Protocols

Less Accurate Characterization → More Accurate Characterization

Clifford simulators	Intermediate simulators	Statevector/density matrix simulators
Stim, Pennylane, Cirq Only Clifford operations $\{H, S, CX\}$	Some non-Clifford operations	Qiskit, Cirq, … Any quantum operation

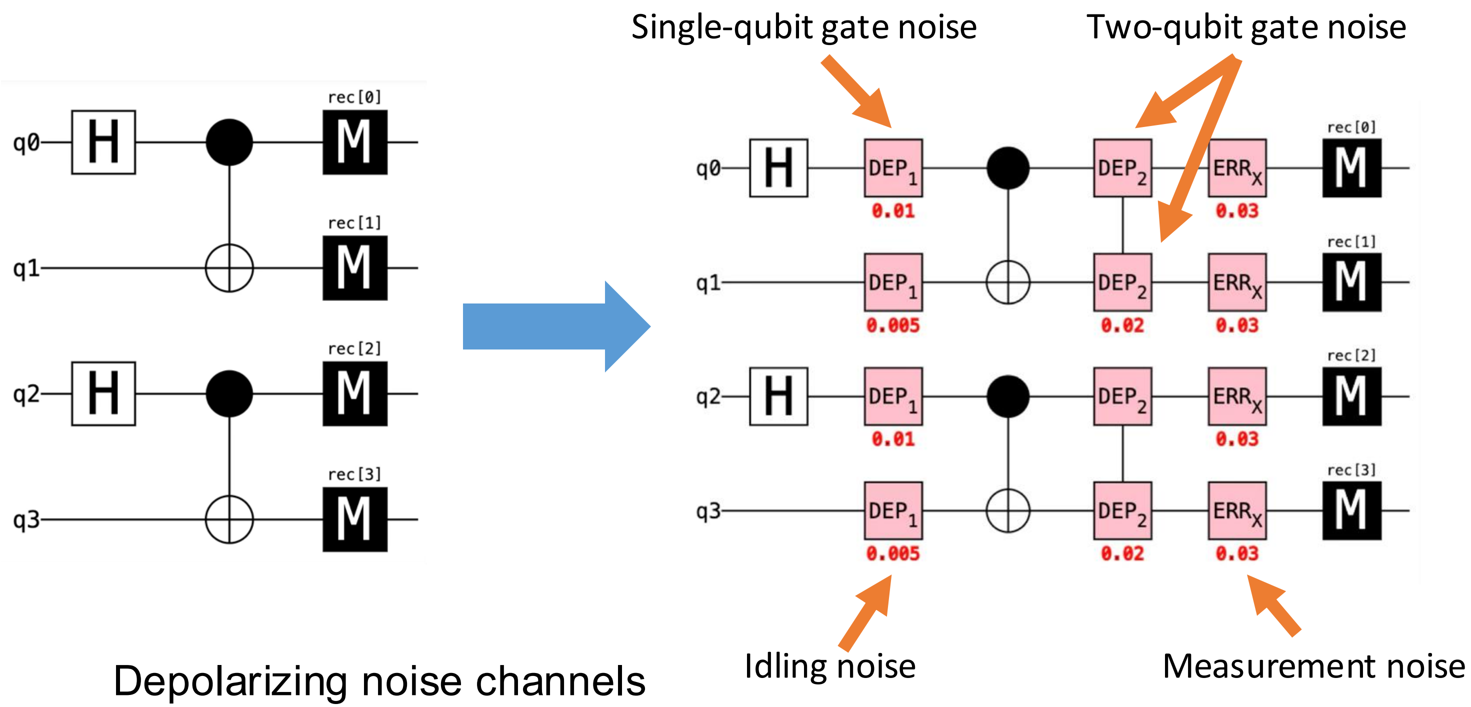
Fast, low memory ← Slow, high memory

1QBit

Noise Models

Mathematically describe the noise that occurs on a real or hypothetical quantum chip.

Simplest noise modes: add quantum channels (representing noise) before/after each operation.



Depolarizing noise channels

Name	Channel
X_ERROR(p)	$\rho \to (1-p)\rho + pX\rho X$
Depolarizing1(p)	$\rho \to (1-p)\rho + \frac{p}{3}(X\rho X + Y\rho Y + Z\rho Z)$
Depolarizing2(p)	$\rho \to (1-p)\rho + \frac{p}{15} \sum_{P_0 \otimes P_1 \in \{I,X,Y,Z\} \otimes \{I,X,Y,Z\} \setminus II} P_0 \otimes P_1 \rho P_0 \otimes P_1$

1QBit

Simulators for FTQC Protocols with Noise

Less Accurate Characterization → More Accurate Characterization

Clifford simulators	Intermediate simulators	Statevector/density matrix simulators
Stim, Pennylane, Cirq		Qiskit, Cirq, …
Only Clifford operations $\{H, S, CX\}$	Some non-Clifford operations	Any quantum operation
Depolarizing noise and erasure channels	1 . Leakage 2. Coherent errors (cross-talk) 3. Non-Markovian noise	Arbitrary quantum channels for noise
Good results at this stage of hardware technology **TopQAD™**	**TopQAD™** Coming soon (1, 2) **TopQAD™** Planned (3)	Useful largely to validate faster simulators

Fast, low memory ← Slow, high memory

1QBit

Uniform Depolarizing Noise Model

Gate	Noise channel map
Preparation/Reset in Z basis (R)	R → X_ERROR(p)
Measurement (M)	M → Classical_Flip(p) → Depolarizing1(p)
Single-qubit gate (G)	G → Depolarizing1(p)
Two-qubit gate (G)	G → Depolarizing2(p)
Idle qubit (I)	Depolarizing1(p)

Simple one-parameter (p) noise model

1QBit

Physical Depolarizing Noise Model

Step 1. Experimental benchmarking

Hardware Parameter	Baseline	Target	Desired
T_1, T_2 times	100 µs	200 µs	340 µs
T_1 tailedness	71 µs	23 µs	23 µs
Single-qubit gate error	0.0004	0.0002	0.00012
Two-qubit gate error	0.003	0.0005	0.00029
State preparation error	0.02	0.01	0.00588
Measurement error	0.01	0.005	0.00294
Reset error	0.01	0.005	0.00294
Single-qubit gate time	25 ns	25 ns	25 ns
Two-qubit gate time	25 ns	25 ns	25 ns
State preparation time	1 µs	1 µs	1 µs
Measurement time	200 ns	100 ns	100 ns
Reset time	200 ns	100 ns	100 ns

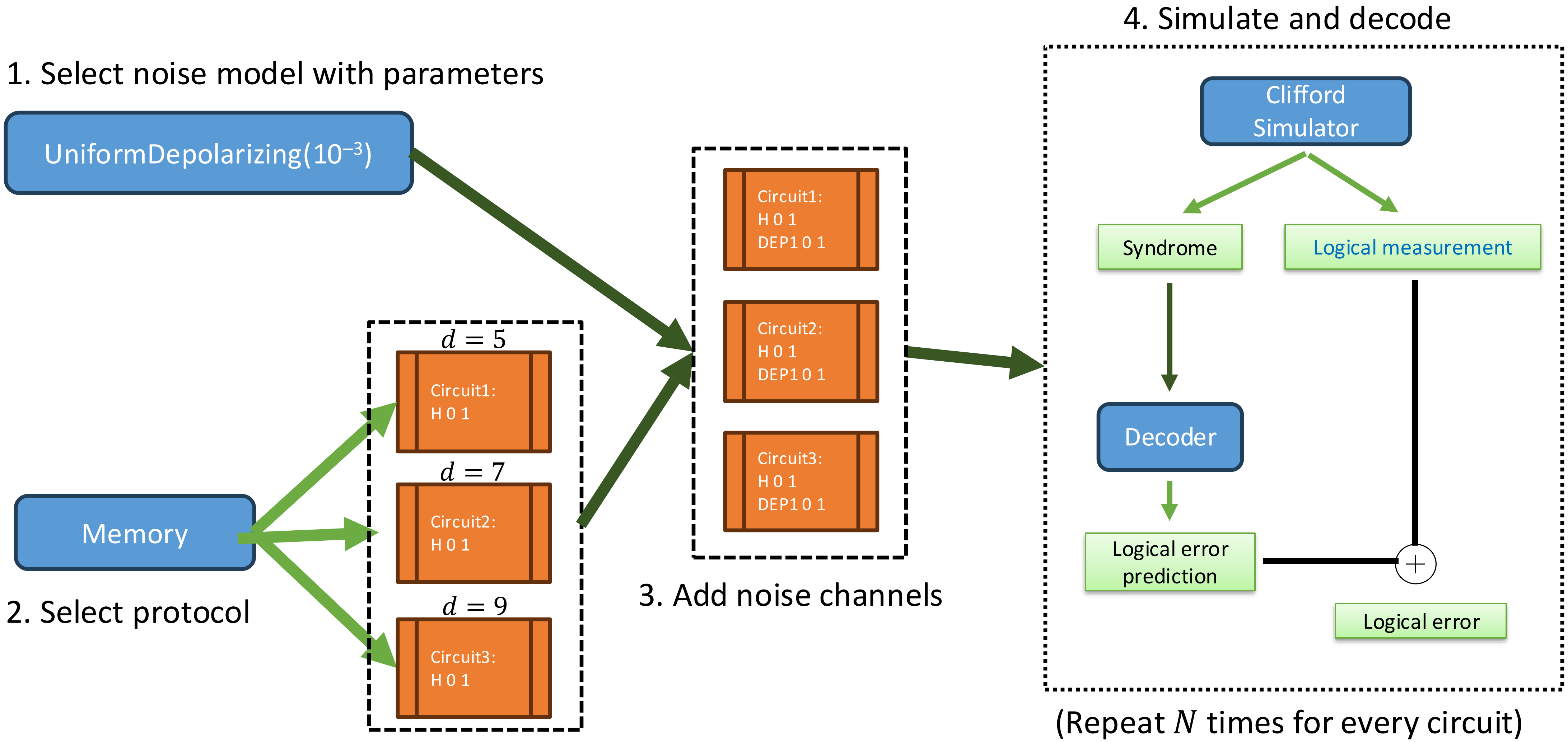
Step 2. Use quantum information theory to infer channel strengths

Gate	Noise channel map
Prep/Reset in Z basis (R)	R → X_ERROR(p_{reset})
Measurement (M)	X_ERROR($p_{\text{measurement}}$) → M
Single-qubit gate (G)	G → Depolarizing1($p_{\text{gate,1}}$)
Two-qubit gate (G)	G → Depolarizing2($p_{\text{gate,2}}$)
Idle qubit (I)	Depolarizing1(p_{idle})

A physical realistic Clifford noise model

1QBit

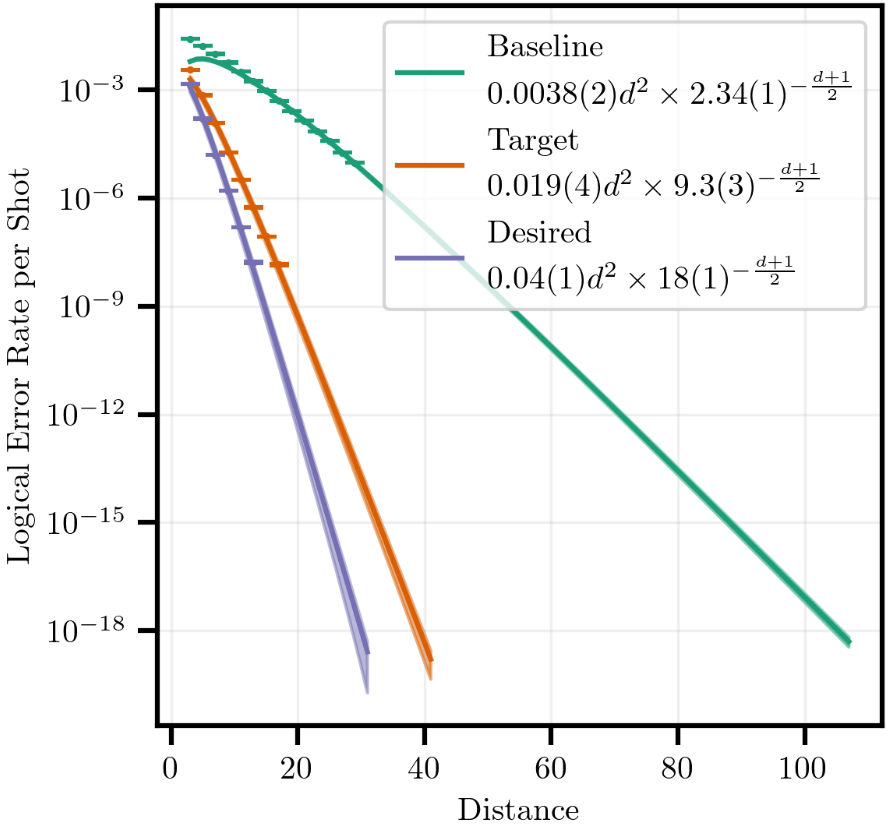
Simulation Flow



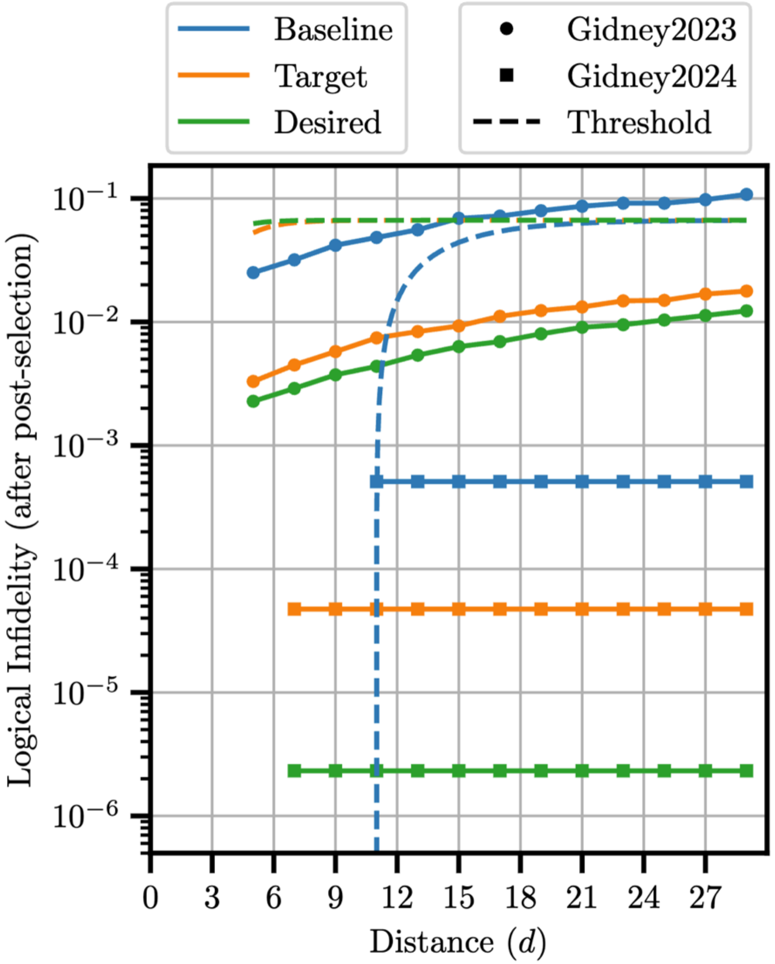
1QBit

Simulation Results for QRE

1. Simulate protocols, assuming each logical step has d stabilization rounds.
2. Regress the results to an appropriate fitting function.
3. Extrapolate to needed distances.



Quantum Memory



Magic State Preparation

Mohseni, M. et al., arXiv:2411.10406 (2025).

FTQC emulation using the TopQAD SDK

Abdullah Khalid and Katie Olfert (*25 min*)

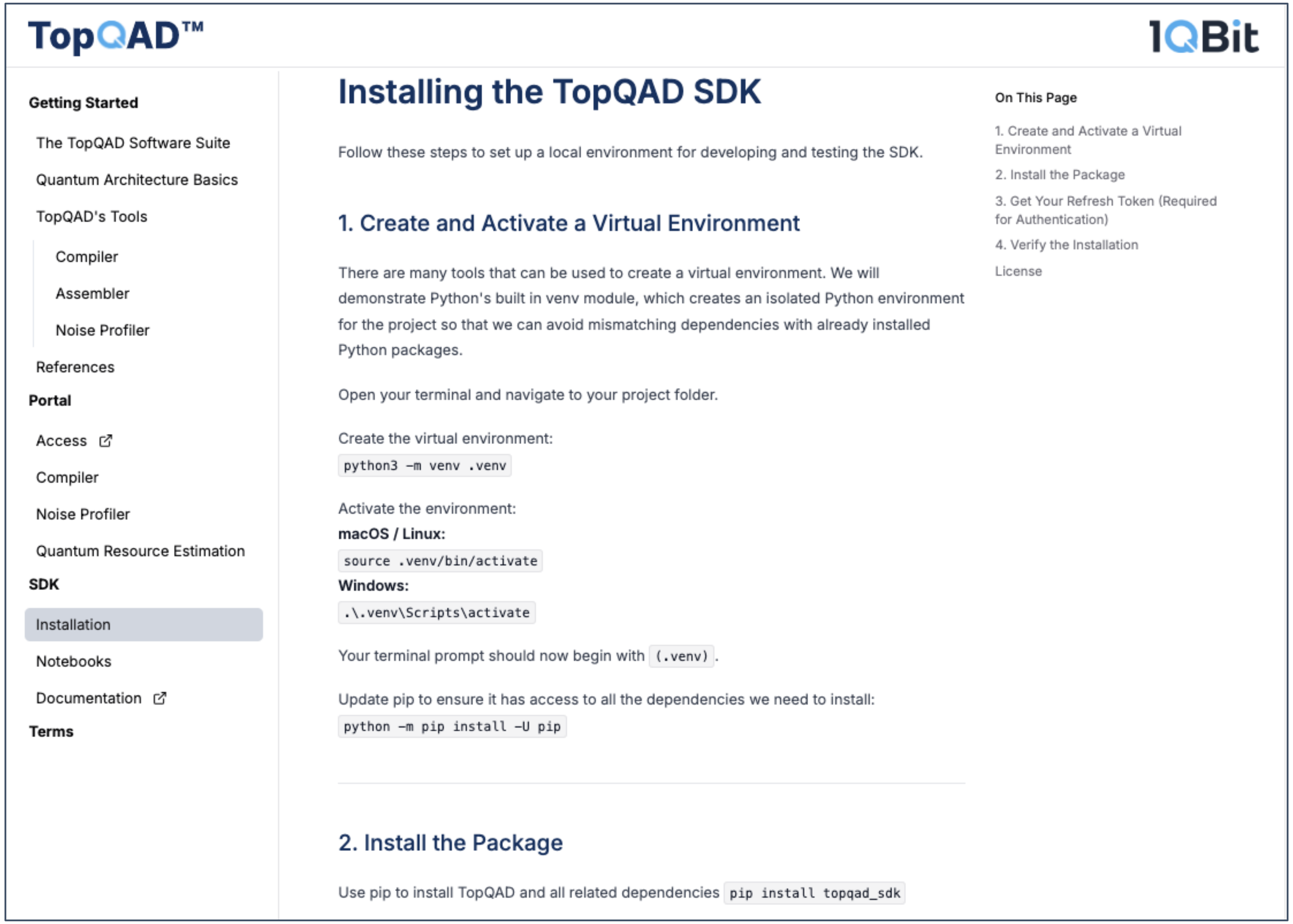
Installation Instructions

https://topqad.1qbit.com

Trial Code: **********

How to Run Notebooks

- ☑ Sign up on TopQAD portal
- ☑ Install TopQAD SDK
- ☑ Download notebooks
- ☑ Get refresh token
- ☑ Make .env file with refresh token inside
- ☑ Run notebooks



TopQAD™ 1QBit

Getting Started
The TopQAD Software Suite
Quantum Architecture Basics
TopQAD's Tools
Compiler
Assembler
Noise Profiler
References
Portal
Access
Compiler
Noise Profiler
Quantum Resource Estimation
SDK
Installation
Notebooks
Documentation
Terms

Installing the TopQAD SDK

Follow these steps to set up a local environment for developing and testing the SDK.

1. Create and Activate a Virtual Environment

There are many tools that can be used to create a virtual environment. We will demonstrate Python's built in venv module, which creates an isolated Python environment for the project so that we can avoid mismatching dependencies with already installed Python packages.

Open your terminal and navigate to your project folder.

Create the virtual environment:

```
python3 -m venv .venv
```

Activate the environment:

macOS / Linux:

```
source .venv/bin/activate
```

Windows:

```
.\.venv\Scripts\activate
```

Your terminal prompt should now begin with `(.venv)`.

Update pip to ensure it has access to all the dependencies we need to install:

```
python -m pip install -U pip
```

2. Install the Package

Use pip to install TopQAD and all related dependencies `pip install topqad_sdk`

On This Page
1. Create and Activate a Virtual Environment
2. Install the Package
3. Get Your Refresh Token (Required for Authentication)
4. Verify the Installation
License

Installation Instructions

https://topqad.1qbit.com

Trial Code: **********

How to Run Notebooks

- ✅ Sign up on TopQAD portal
- ✅ Install TopQAD SDK
- ☑ Download notebooks
- ☑ Get refresh token
- ☑ Make .env file with refresh token inside
- ☑ Run notebooks

TopQAD™ 1QBit

Getting Started
- The TopQAD Software Suite
- Quantum Architecture Basics
- TopQAD's Tools
  - Compiler
  - Assembler
  - Noise Profiler
- References

Portal
- Access
- Compiler
- Noise Profiler
- Quantum Resource Estimation

SDK
- Installation
- Notebooks
- Documentation

Terms

Installing the TopQAD SDK

Follow these steps to set up a local environment for developing and testing the SDK.

1. Create and Activate a Virtual Environment

There are many tools that can be used to create a virtual environment. We will demonstrate Python's built in venv module, which creates an isolated Python environment for the project so that we can avoid mismatching dependencies with already installed Python packages.

Open your terminal and navigate to your project folder.

Create the virtual environment:

```
python3 -m venv .venv
```

Activate the environment:

macOS / Linux:

```
source .venv/bin/activate
```

Windows:

```
.\.venv\Scripts\activate
```

Your termina

Update pip

```
python -m
```

2. Instal

Use pip to install TopQAD and all related dependencies `pip install topqad_sdk`

On This Page
1. Create and Activate a Virtual Environment
2. Install the Package
3. Get Your Refresh Token (Required for Authentication)
4. Verify the Installation
License

Note:
Running the SDK will lock TopQAD beta access to the device you are using.

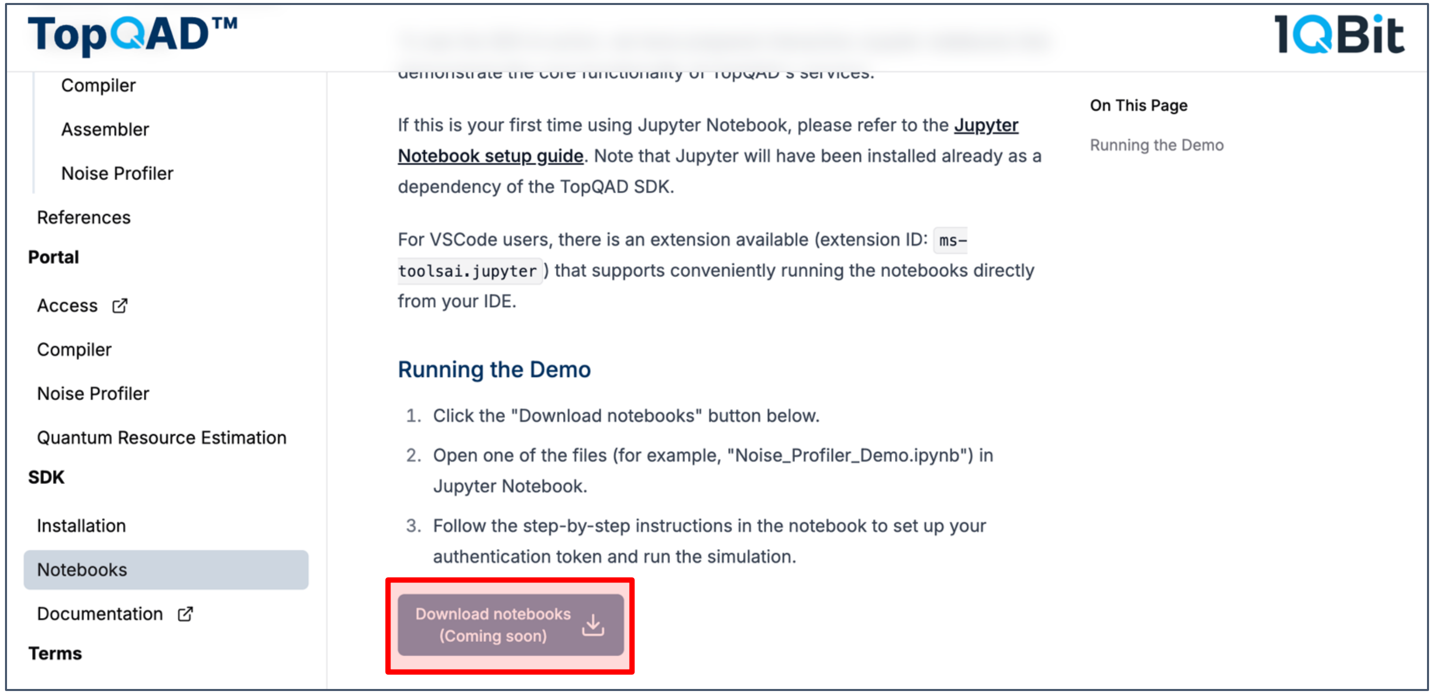
Download the Notebooks

https://topqad.1qbit.com

Trial Code: **********

How to Run Notebooks

- ☑ Sign up on TopQAD portal
- ☑ Install TopQAD SDK
- ☑ Download notebooks
- ☑ Get refresh token
- ☑ Make .env file with refresh token inside
- ☑ Run notebooks



1QBit

Introducing Refresh Tokens

https://topqad.1qbit.com

Trial Code: **********

How to Run Notebooks

- ✅ Sign up on TopQAD portal
- ✅ Install TopQAD SDK
- ✅ Download notebooks
- ✅ Get refresh token
- ☑ Make .env file with refresh token inside
- ☑ Run notebooks

⚠ The refresh token is sensitive and should be ⚠

⚠ treated like a password ⚠

For your security:

- Keep it out of code (add .env to .gitignore, if applicable)
- Never paste tokens in GitHub issues or pull requests
- Do not log the token's value

1QBit

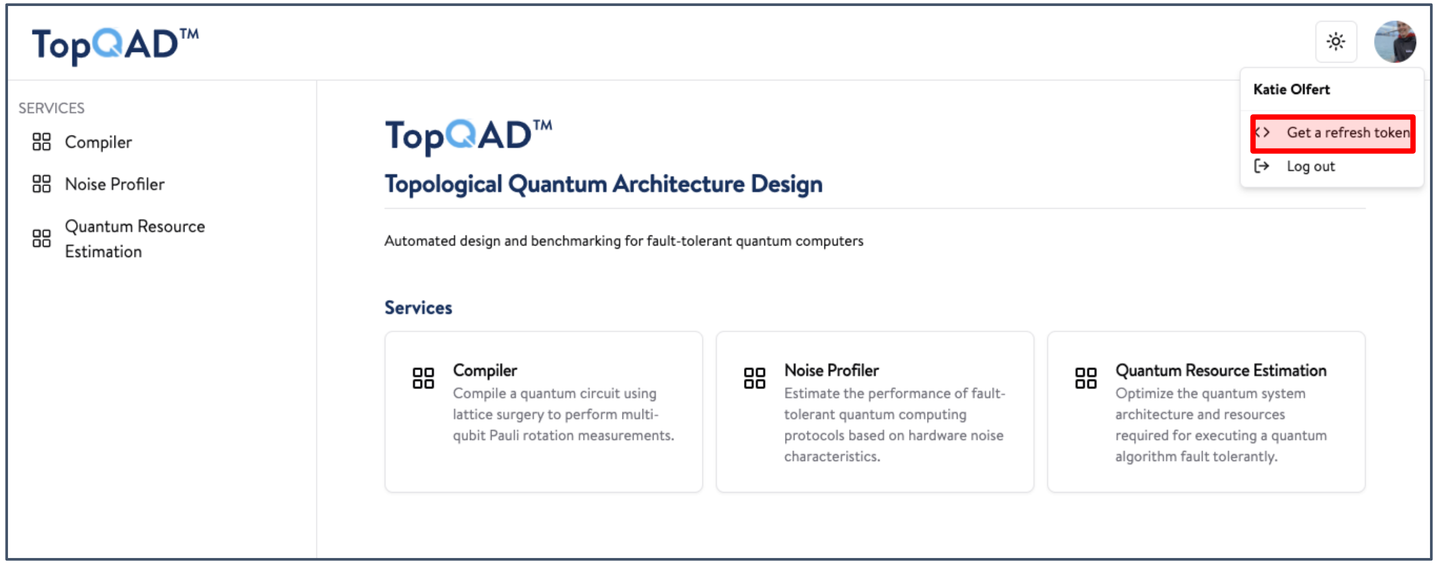
Obtain Your Refresh Token

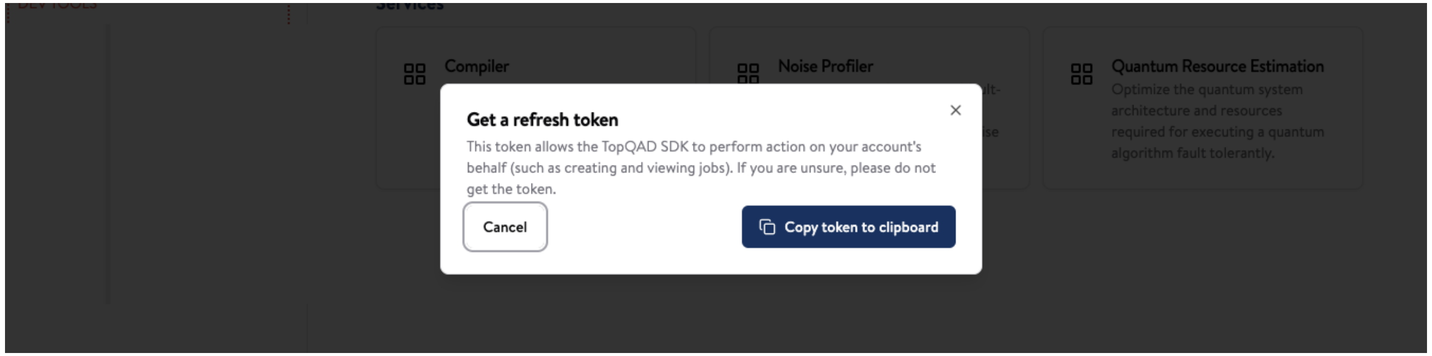
https://topqad.1qbit.com

Trial Code: **********

How to Run Notebooks

- ✅ Sign up on TopQAD portal
- ✅ Install TopQAD SDK
- ✅ Download notebooks
- ✅ Get refresh token
- ☑ Make .env file with refresh token inside
- ☑ Run notebooks





1QBit

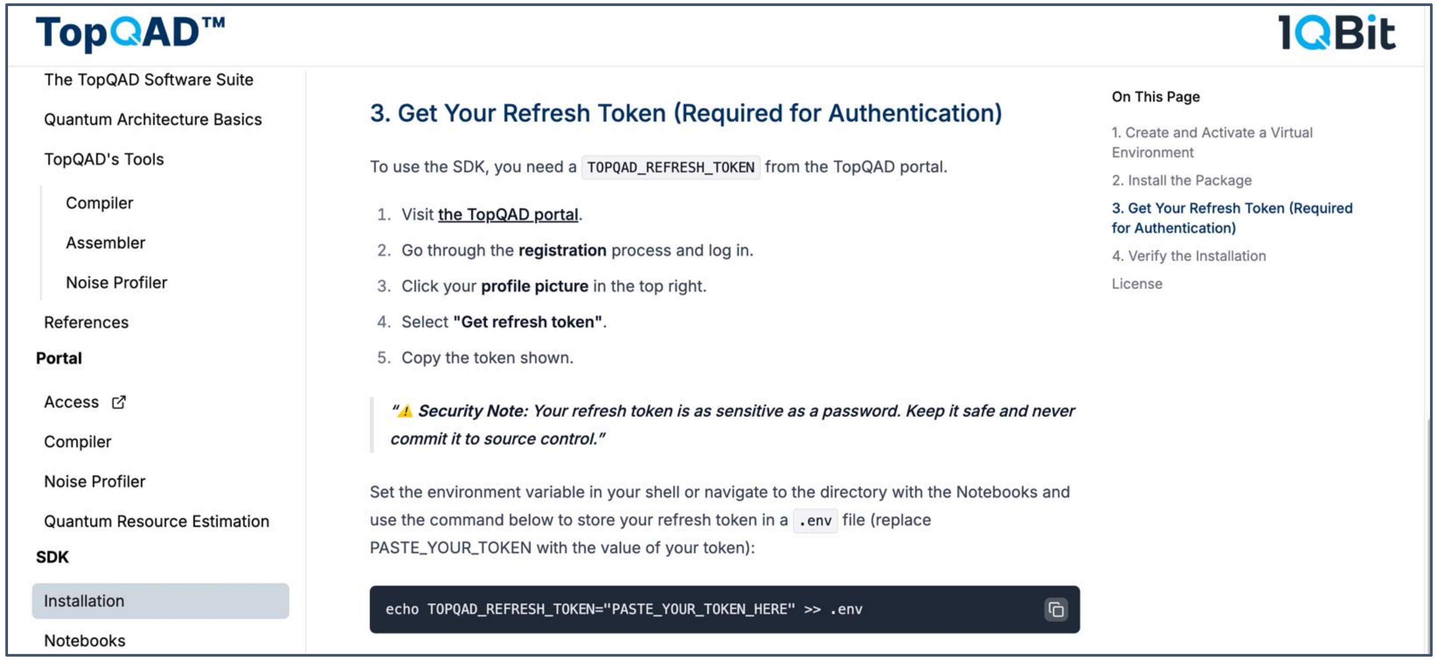
Place Refresh Token in .env File

https://topqad.1qbit.com

Trial Code: **********

How to Run Notebooks

- ✅ Sign up on TopQAD portal
- ✅ Install TopQAD SDK
- ✅ Download notebooks
- ✅ Get refresh token
- ✅ Make .env file with refresh token inside
- ☑ Run notebooks



TopQAD™

The TopQAD Software Suite
Quantum Architecture Basics
TopQAD's Tools
Compiler
Assembler
Noise Profiler
References
Portal
Access
Compiler
Noise Profiler
Quantum Resource Estimation
SDK
Installation
Notebooks

3. Get Your Refresh Token (Required for Authentication)

To use the SDK, you need a `TOPQAD_REFRESH_TOKEN` from the TopQAD portal.

1. Visit the TopQAD portal.
2. Go through the **registration** process and log in.
3. Click your **profile picture** in the top right.
4. Select **"Get refresh token"**.
5. Copy the token shown.

> *"⚠ **Security Note:** Your refresh token is as sensitive as a password. Keep it safe and never commit it to source control."*

Set the environment variable in your shell or navigate to the directory with the Notebooks and use the command below to store your refresh token in a `.env` file (replace PASTE_YOUR_TOKEN with the value of your token):

```
echo TOPQAD_REFRESH_TOKEN="PASTE_YOUR_TOKEN_HERE" >> .env
```

On This Page
1. Create and Activate a Virtual Environment
2. Install the Package
3. Get Your Refresh Token (Required for Authentication)
4. Verify the Installation
License

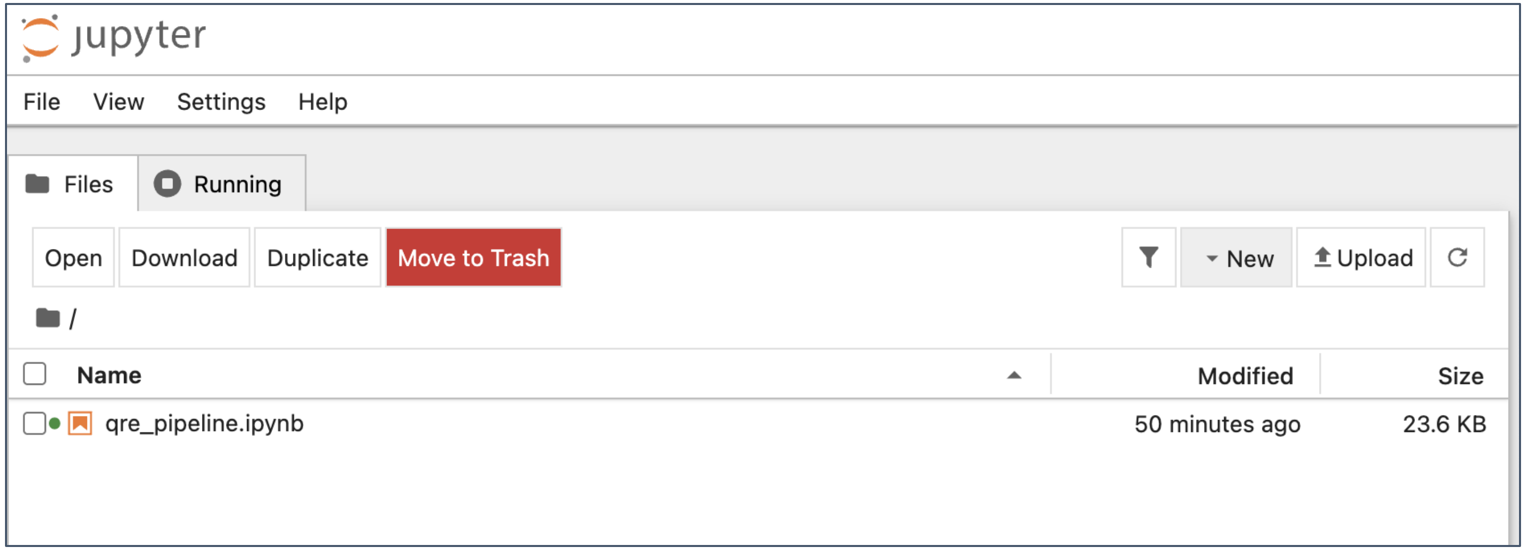
Running the Notebooks

https://topqad.1qbit.com

Trial Code: **********

How to Run Notebooks

- ✅ Sign up on TopQAD portal
- ✅ Install TopQAD SDK
- ✅ Download notebooks
- ✅ Get refresh token
- ✅ Make .env file with refresh token inside
- ✅ Run notebooks



1QBit

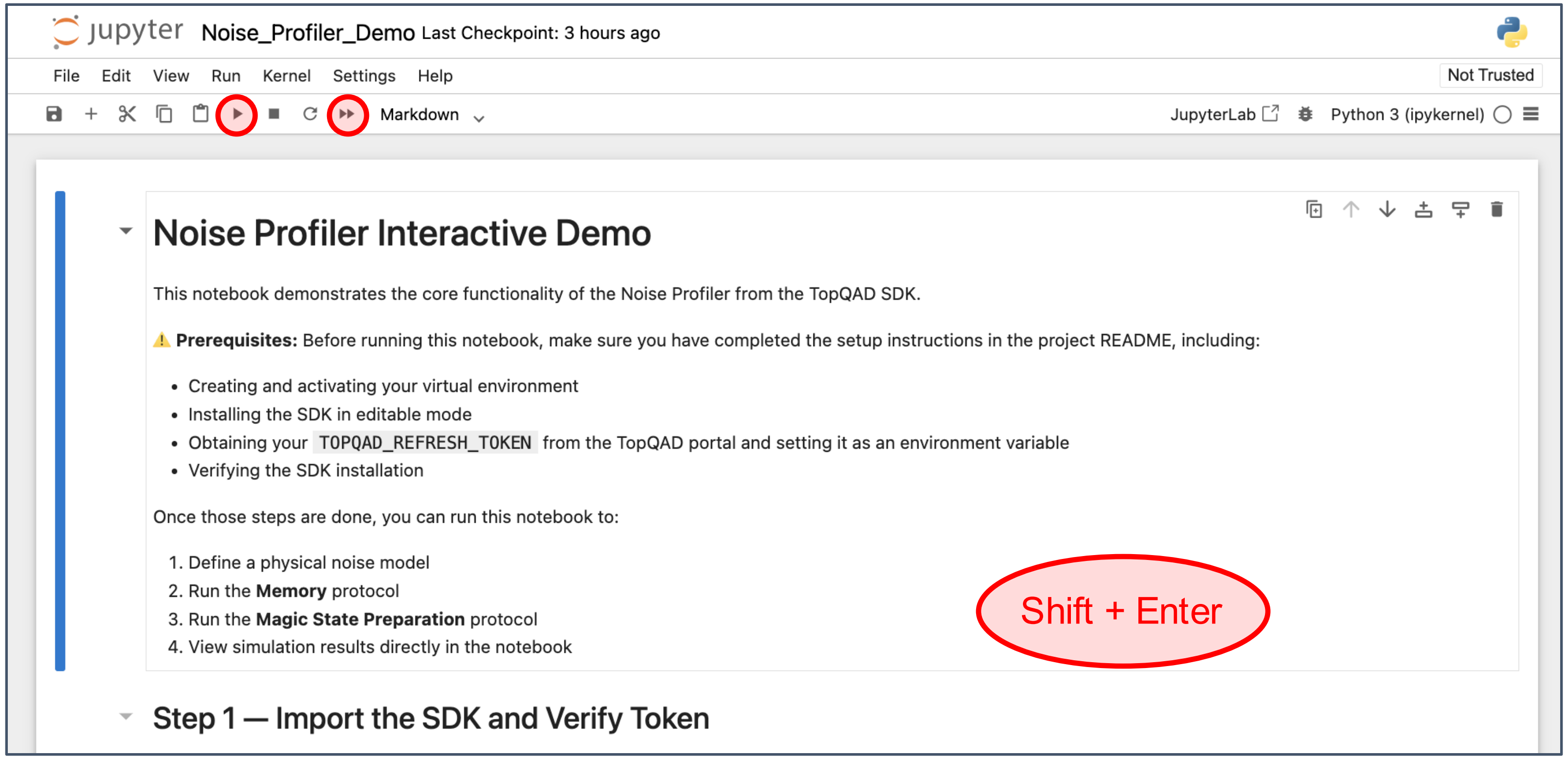
Running the Notebooks

https://topqad.1qbit.com

Trial Code: **********

How to Run Notebooks

- ✅ Sign up on TopQAD portal
- ✅ Install TopQAD SDK
- ✅ Download notebooks
- ✅ Get refresh token
- ✅ Make .env file with refresh token inside
- ✅ Run notebooks



Quantum resource estimation using the TopQAD SDK

Allyson Silva and Katie Olfert (*20 min*)

1QBit

Features and Releases

https://topqad.1qbit.com

Trial Code: **********

Beta access – special offer for QCE25 TUT21 attendees

- Free portal and SDK access (time limited)
- Noise Profiler, Compiler, Quantum Resource Estimation services
- Unlimited jobs, one job at a time
- One device

TopQAD™

September 2025

Today's Tutorial

Account activation

Portal interaction

SDK interaction

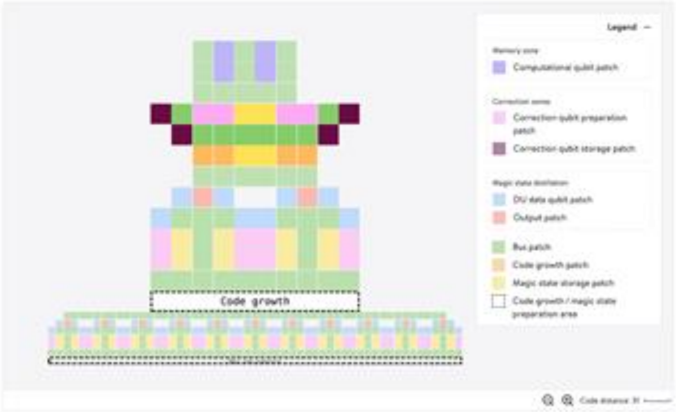
October 2025

Beta Update

Circuit file upload

QRE lite

Architecture visualizer



Q1 2026

Commercial Launch

Paid product

Multiple jobs at a time

Multiple devices

1QBit

Stay in touch!

topqad@1qbit.com