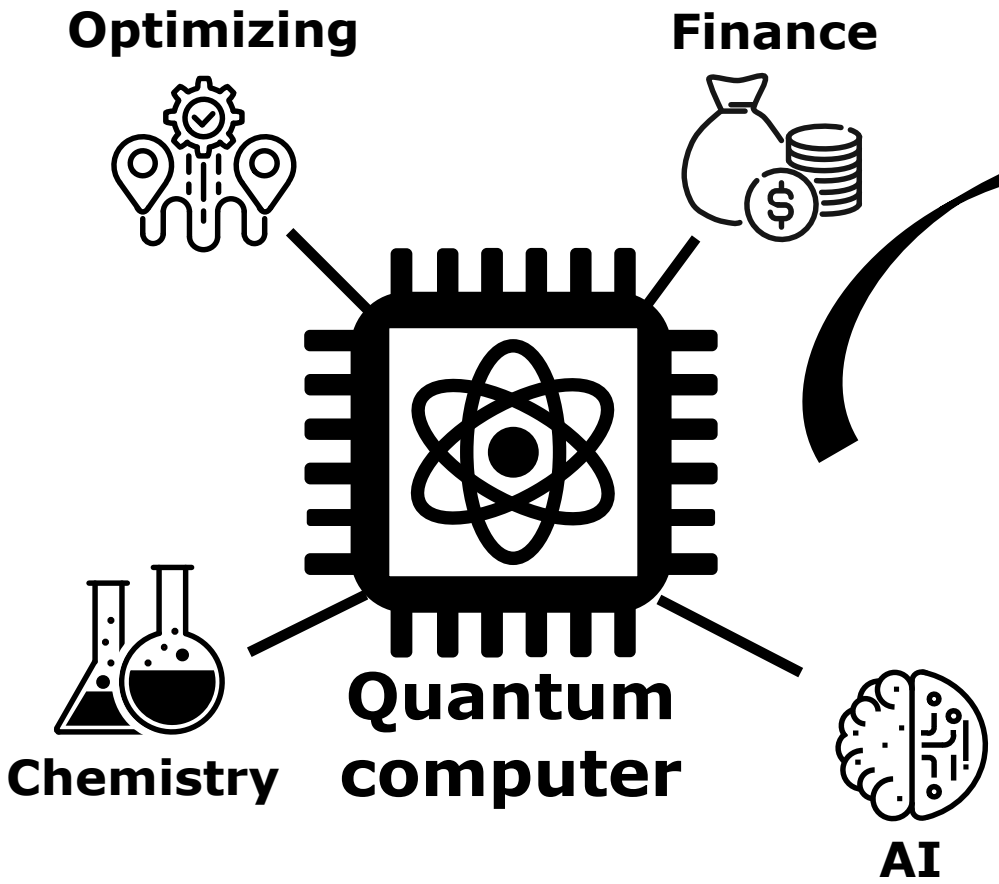
HPCS
High Performance Computer System Lab

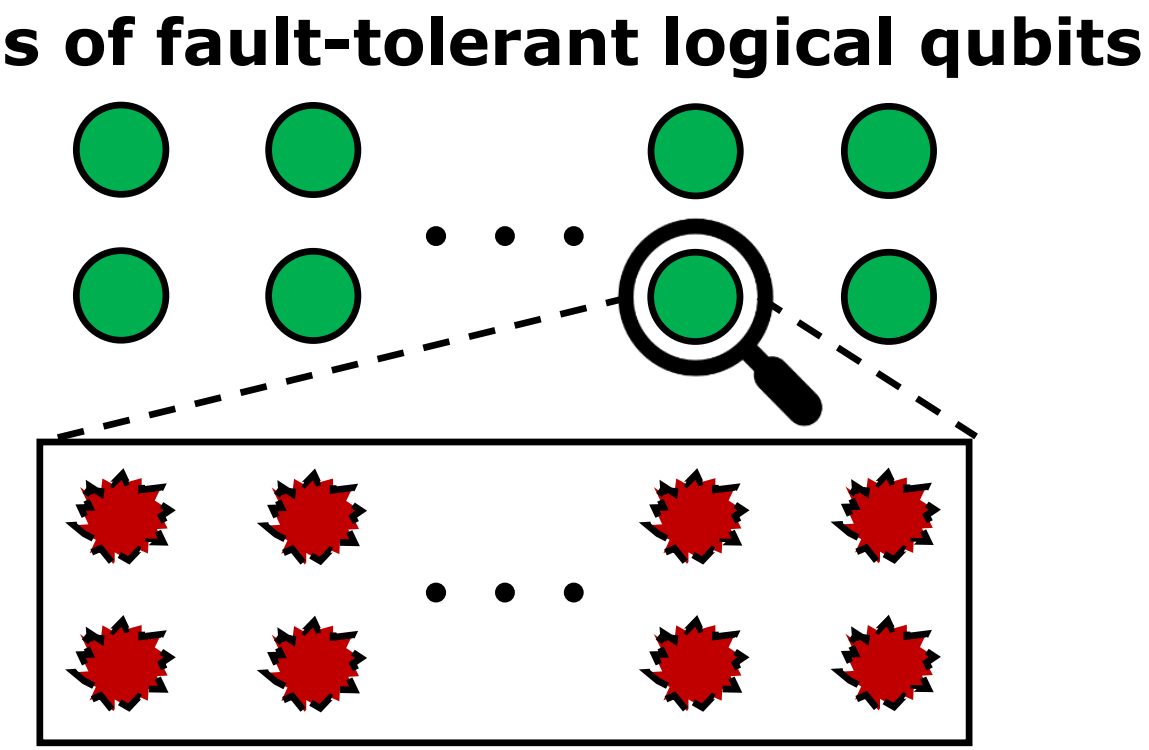
XQsim:
Scalability analysis tool for the fault-tolerant quantum control processor

Ilkwon Byun
E-mail: ik.byun@snu.ac.kr

High Performance Computer System (HPCS) Lab
Department of Electrical and Computer Engineering
Seoul National University

Toward large-scale quantum computer

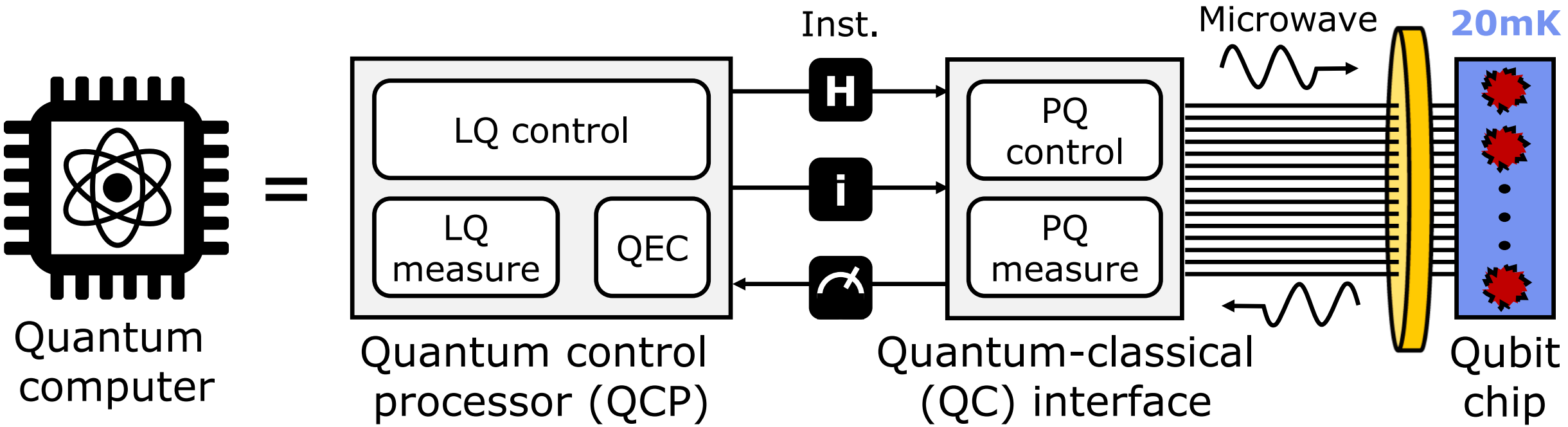




We need a fault-tolerant quantum computer with 10+K physical qubits!

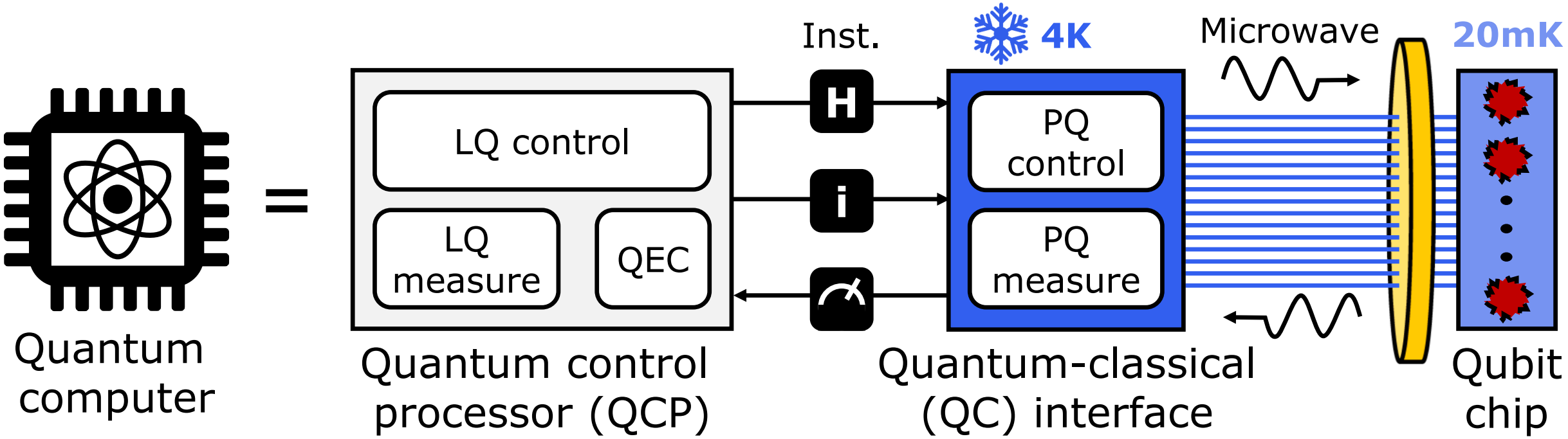
Scalable quantum control system

- **We need a scalable quantum-classical interface and quantum control processor**



Targeting "scalable control processor"

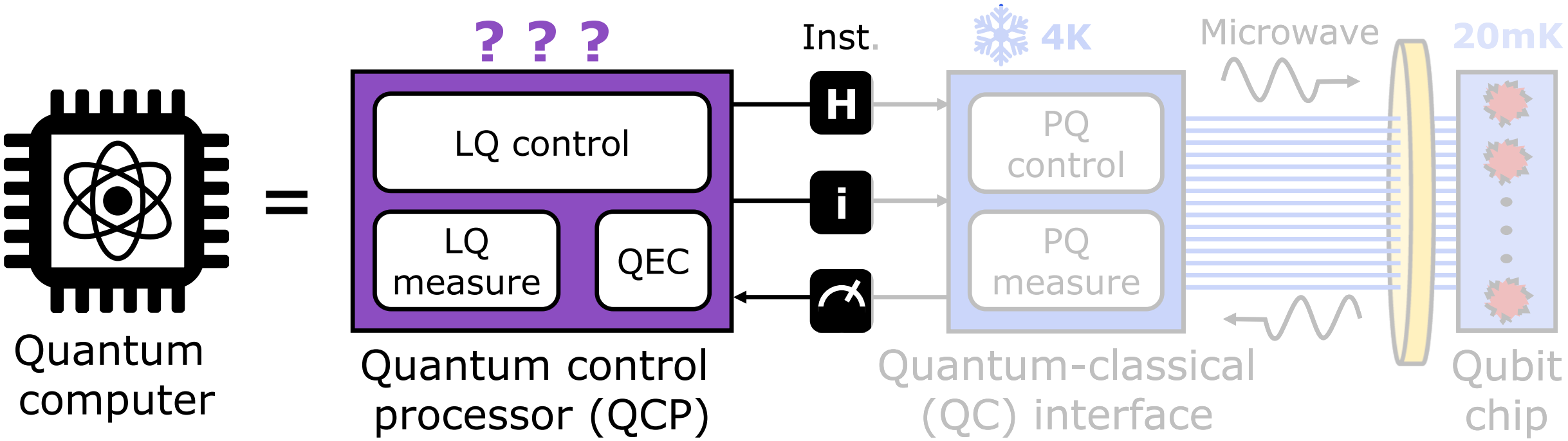
- **Scalable QCP has not been actively explored yet**



Actively explored scalable QC interface:
Run at 4K to utilize scalable interconnects

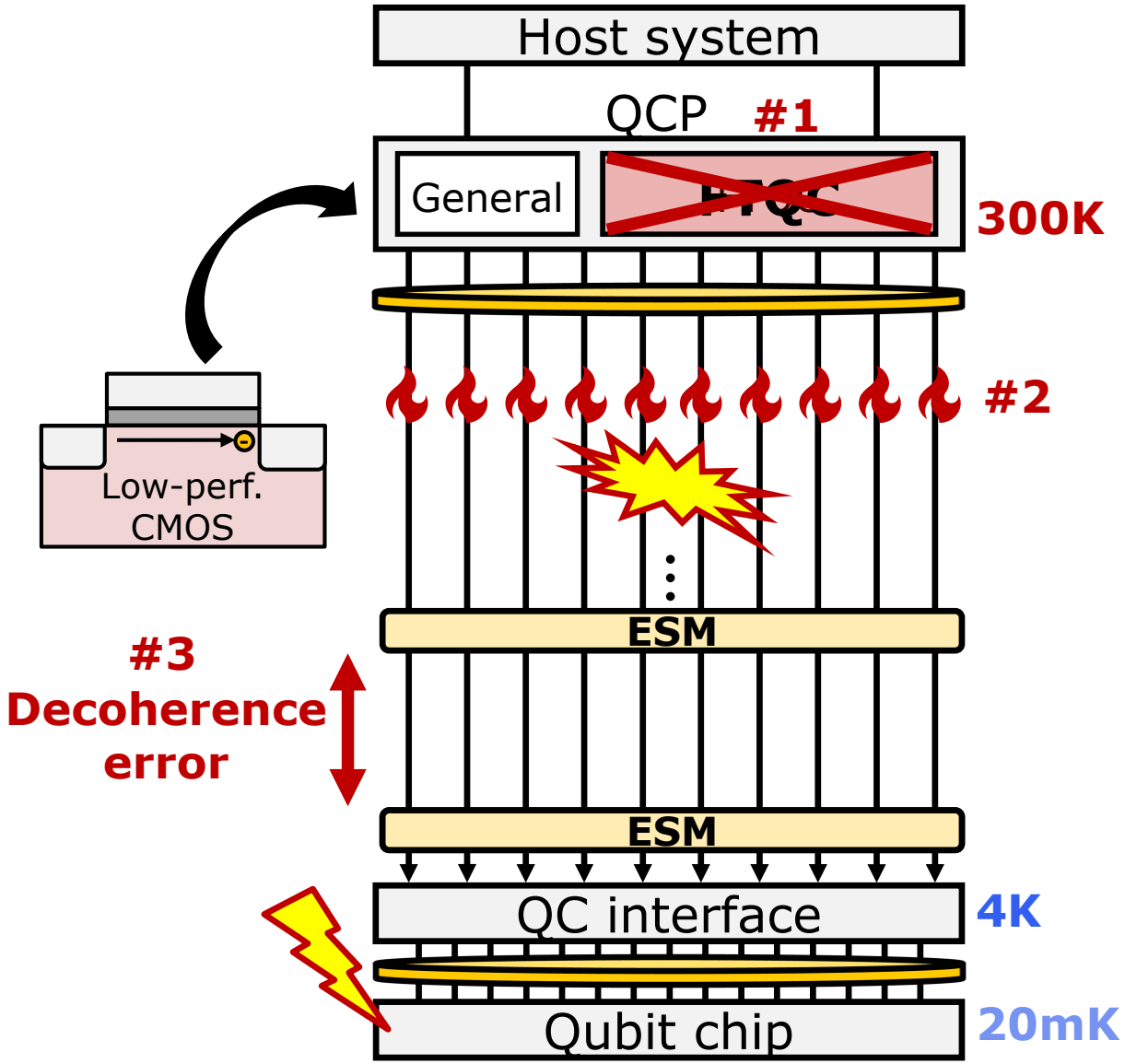
Targeting "scalable control processor"

- **Scalable QCP has not been actively explored yet**



We should explore a scalable QCP architecture

Limited scalability of today's QCP



#1. Microarchitecture

No scalable μarch unit for the fault-tolerant quantum computing

#2. Temperature

300K operation

→ Huge 300K-4K data transfer

→ Wire heat > 4K power budget

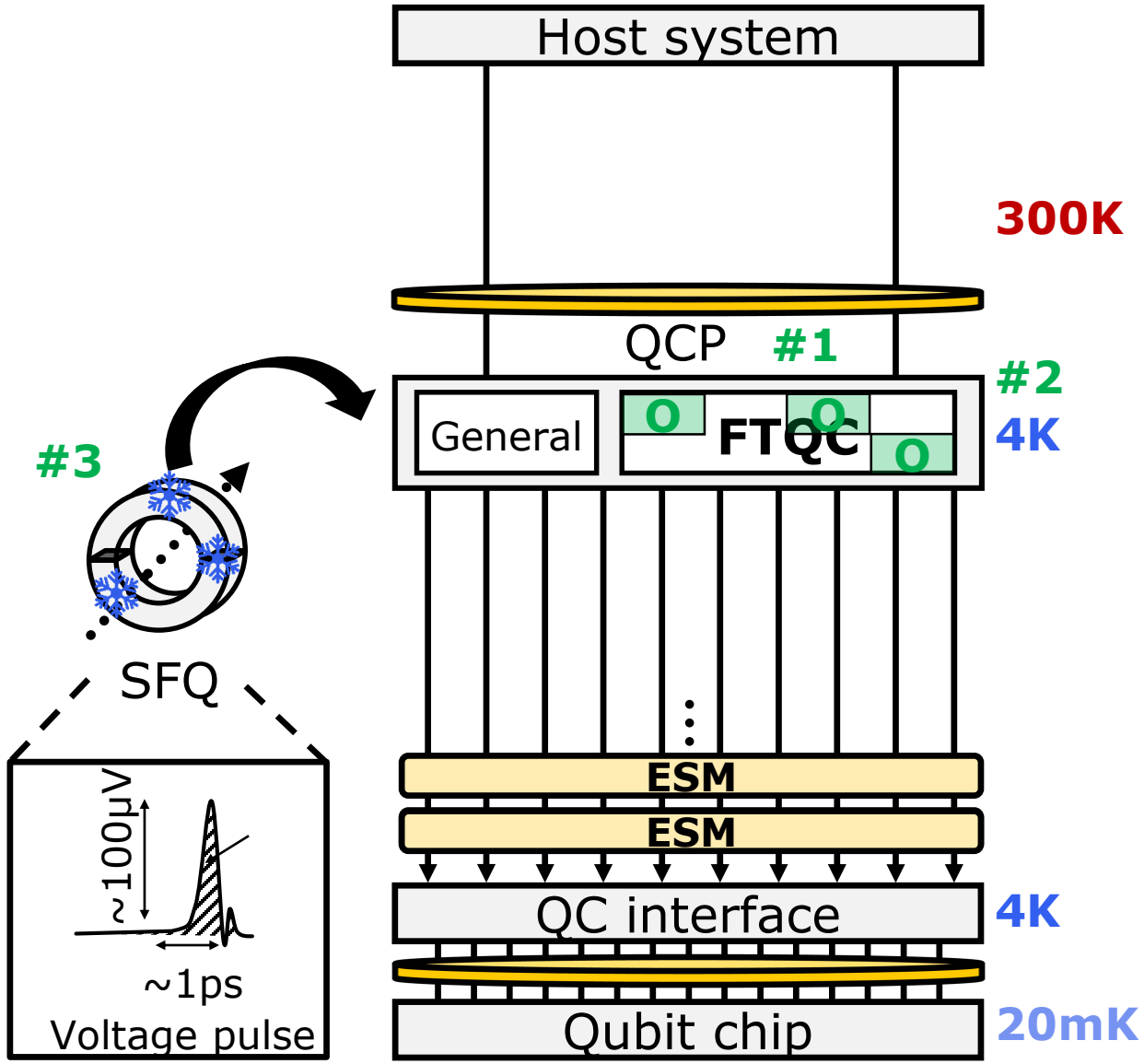
#3. Technology

Performance-limited CMOS

→ Slow QED or Low inst. BW

→ Decoherence error

Recent ideas for scalable QCP



#1. Microarchitecture

(+) Scalable FTQC unit research

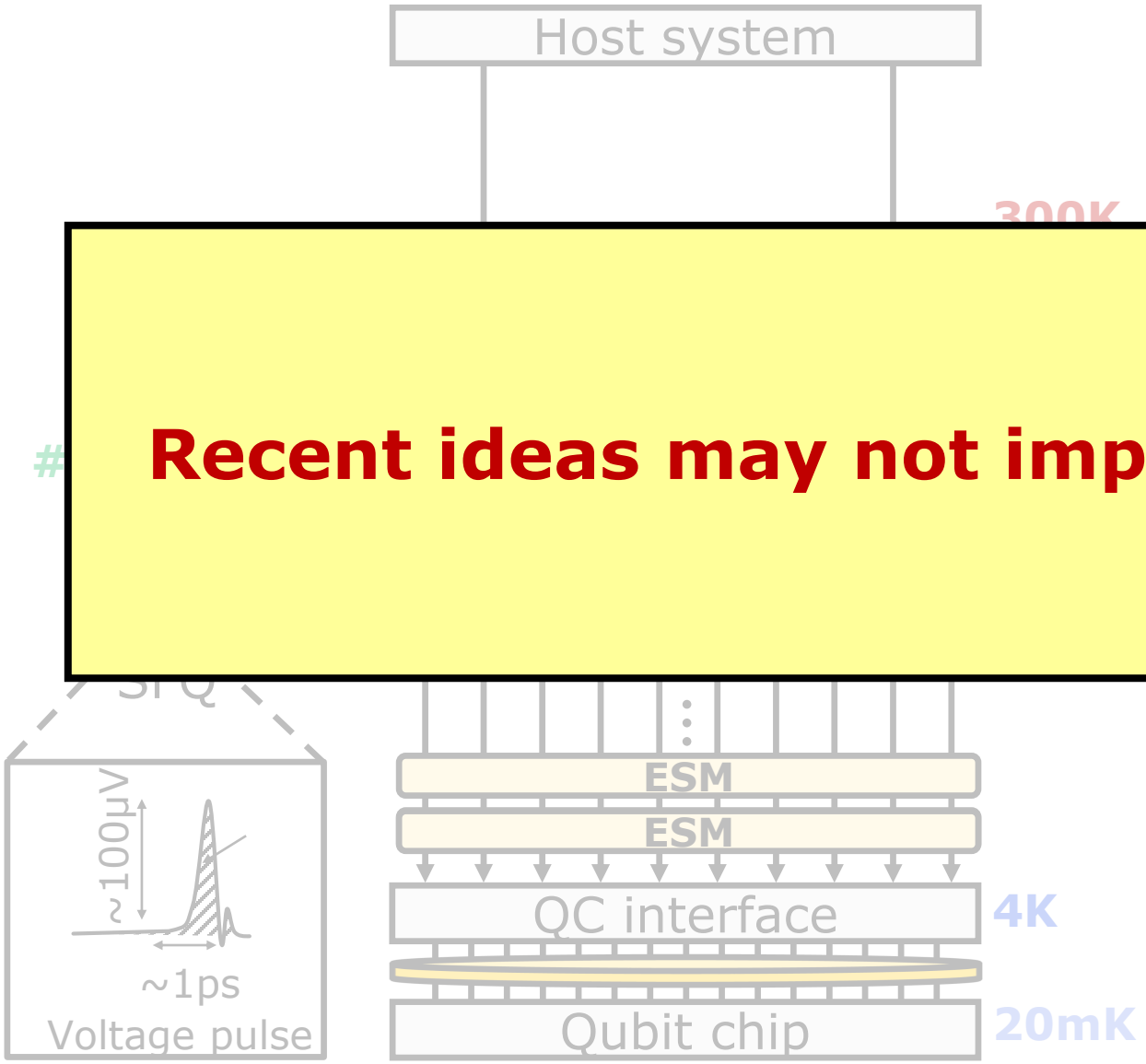
#2. Temperature

(+) 4K operation

#3. Technology

(+) Fast & Low-power SFQ

Recent ideas for scalable QCP



#1. Microarchitecture

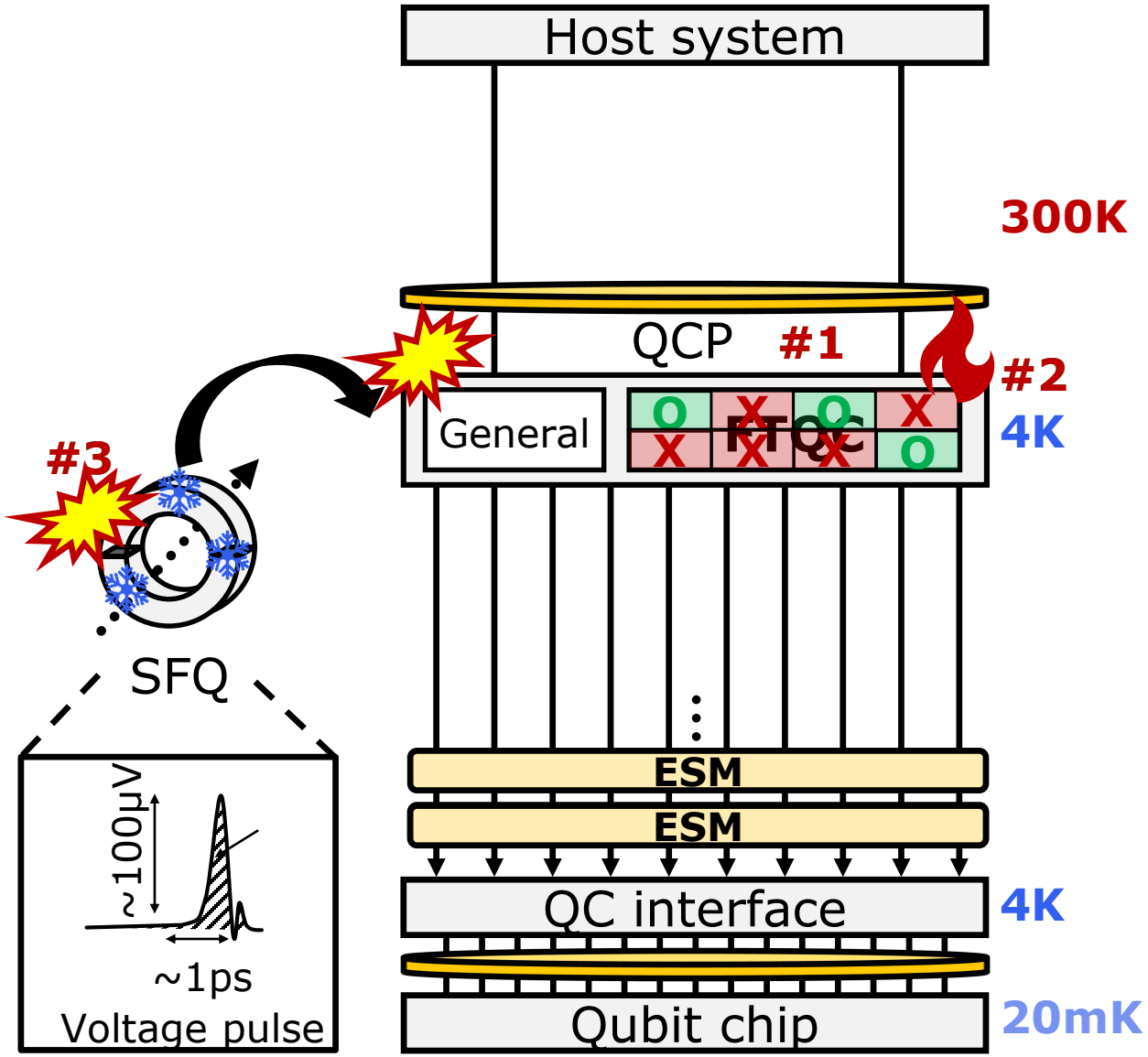
(+) Scalable FTQC unit research

Recent ideas may not improve the scalability of QCP!

#3. Technology

(+) Fast & Low-power SFQ

Recent ideas for scalable QCP



#1. Microarchitecture

(+) Scalable FTQC unit research

(−) Limited μarch coverage

#2. Temperature

(+) 4K operation

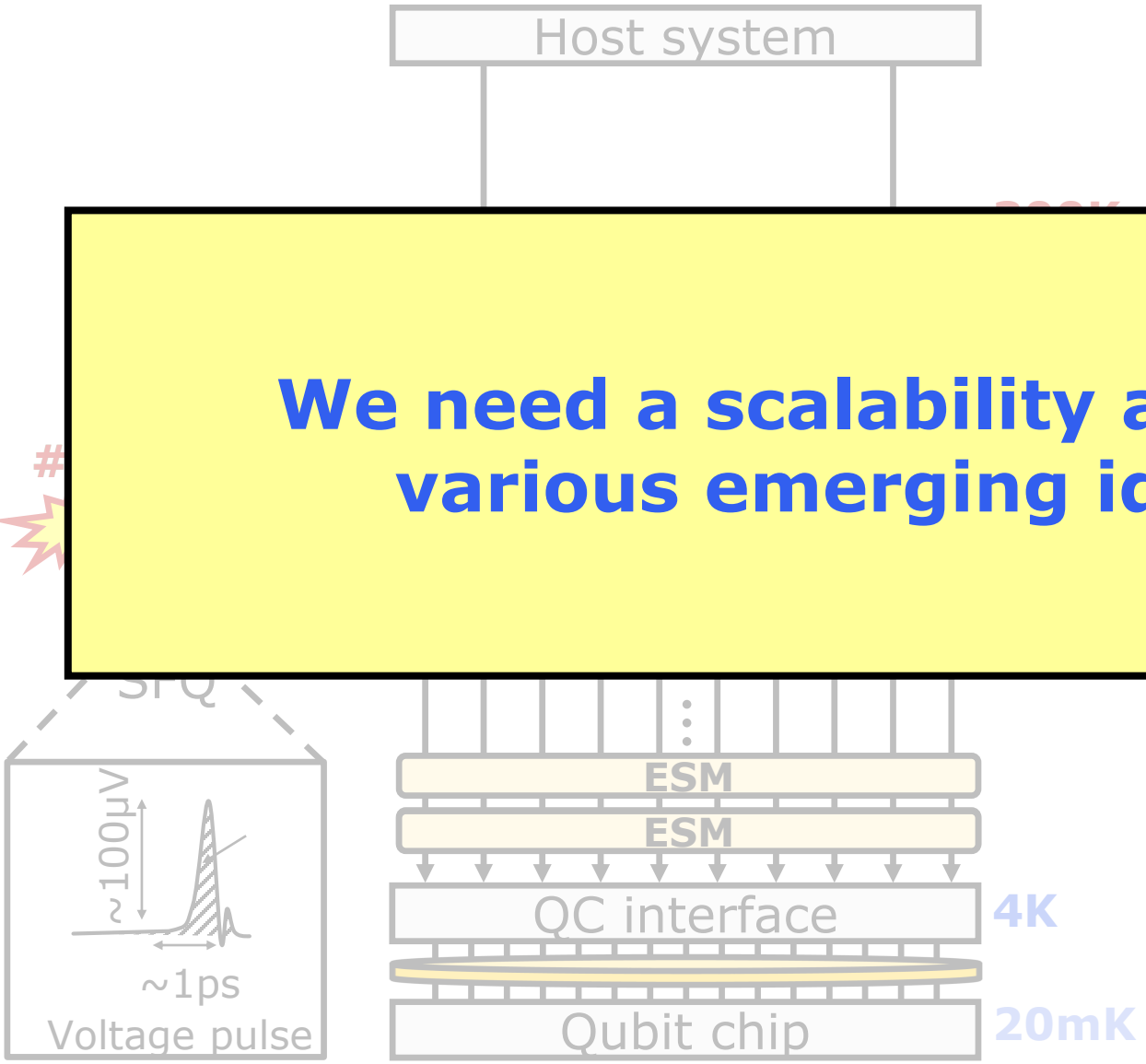
(−) 4K device power dissipation

#3. Technology

(+) Fast & Low-power SFQ

(−) Non-trivial scalable design

Recent ideas for scalable QCP



We need a scalability analysis tool to evaluate various emerging ideas in all directions!

#1. Microarchitecture

(+) Scalable FTQC unit research

#3. Technology

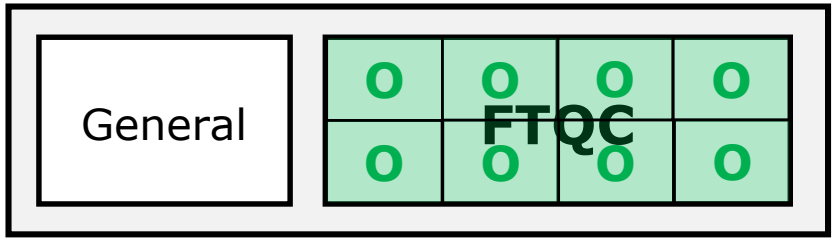
(+) Fast & Low-power SFQ

(–) Non-trivial scalable design

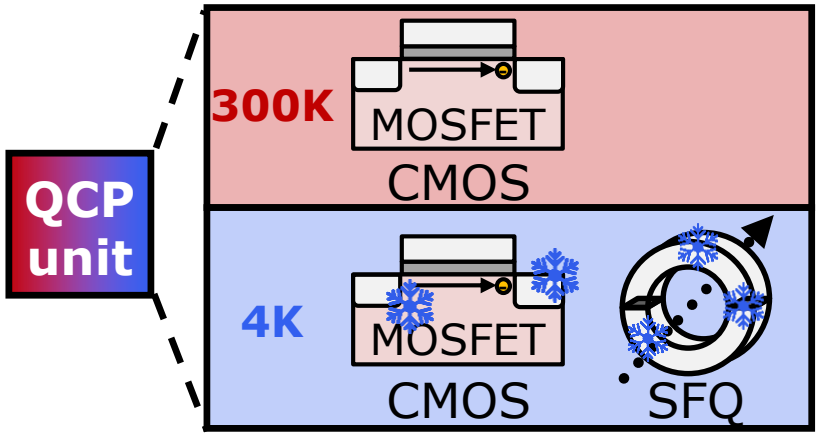
Goal: QCP scalability analysis tool

- **Evaluate the QCP's scalability for various μarch, temperature, and device technologies**

QCP microarchitecture



Temperature & Technology



QCP scalability analysis tool

Scalability metrics

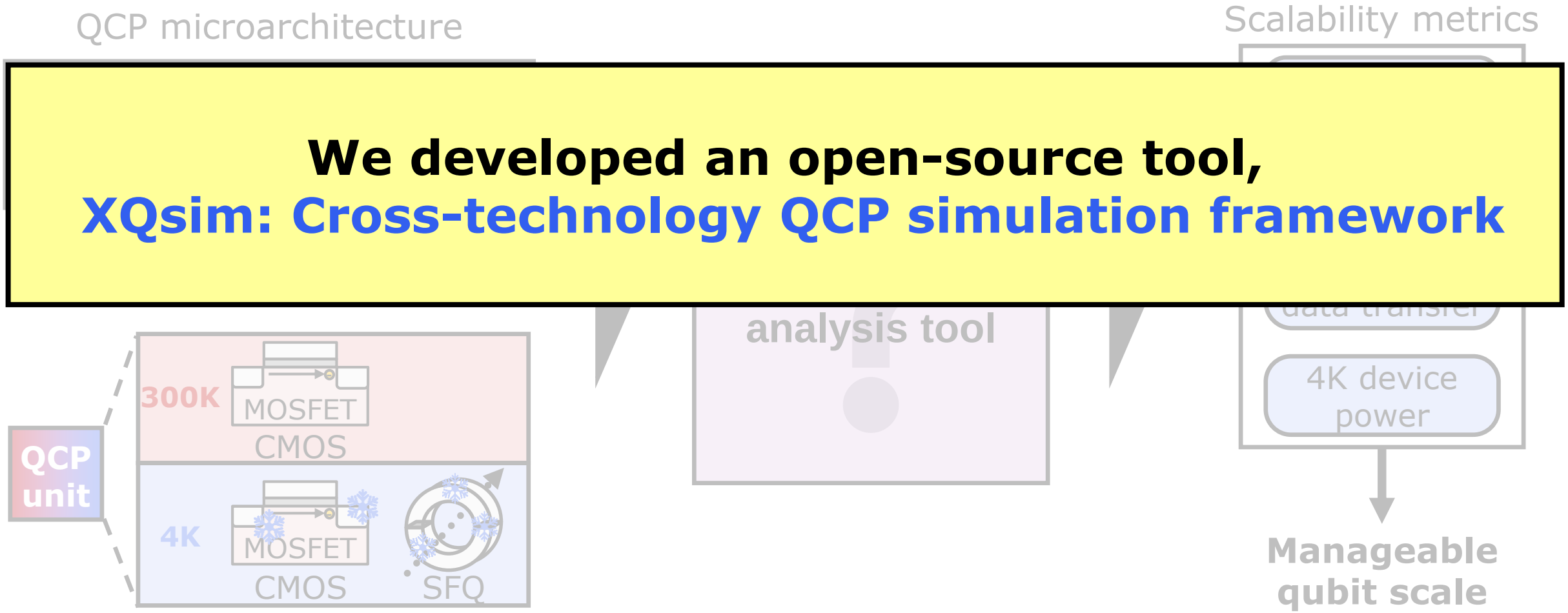
- Inst. BW
- EDU latency
- 300K-4K data transfer
- 4K device power

Manageable qubit scale

Goal: QCP scalability analysis tool

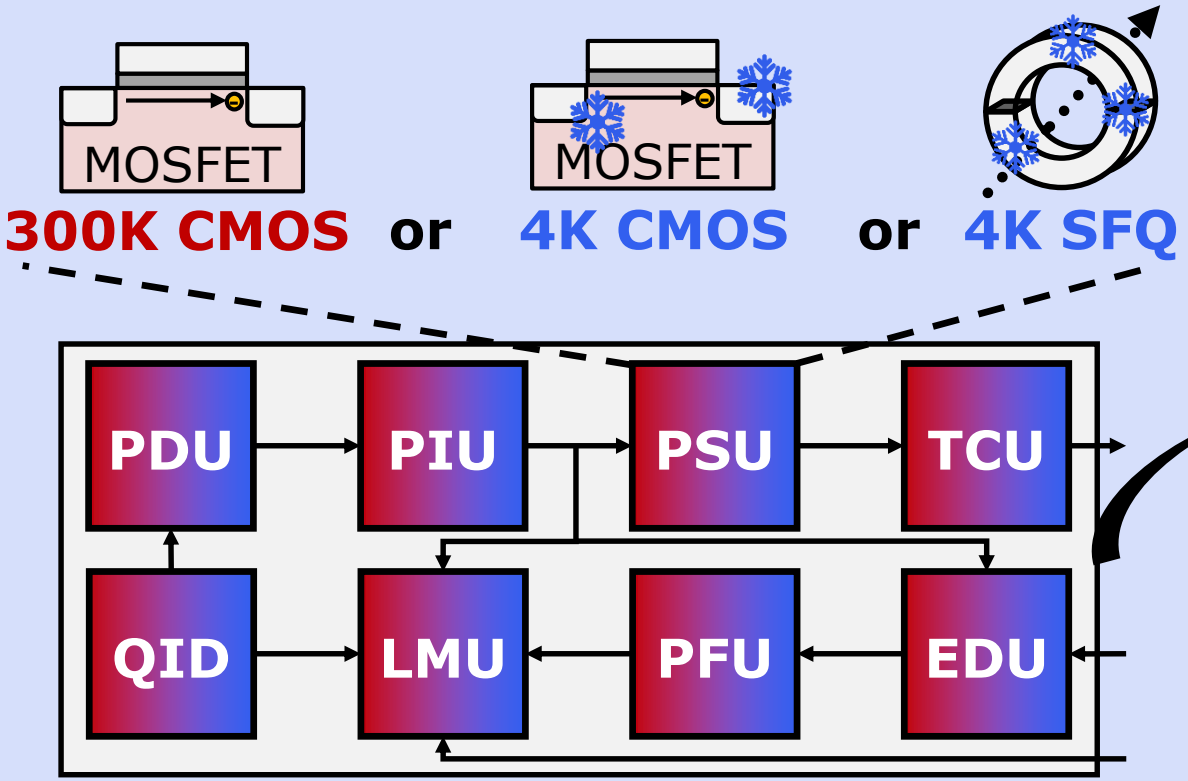
- Evaluate the QCP's scalability for various μarch, temperature, and device technologies

We developed an open-source tool,
XQsim: Cross-technology QCP simulation framework



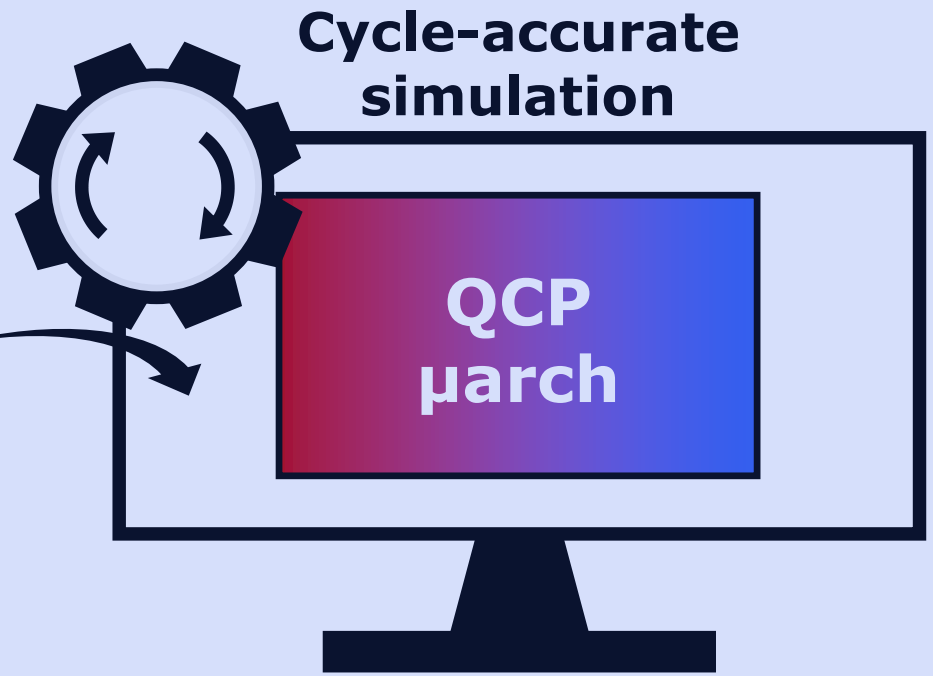
XQsim: Overview

1) XQ-estimator



Frequency and power of each µarch unit

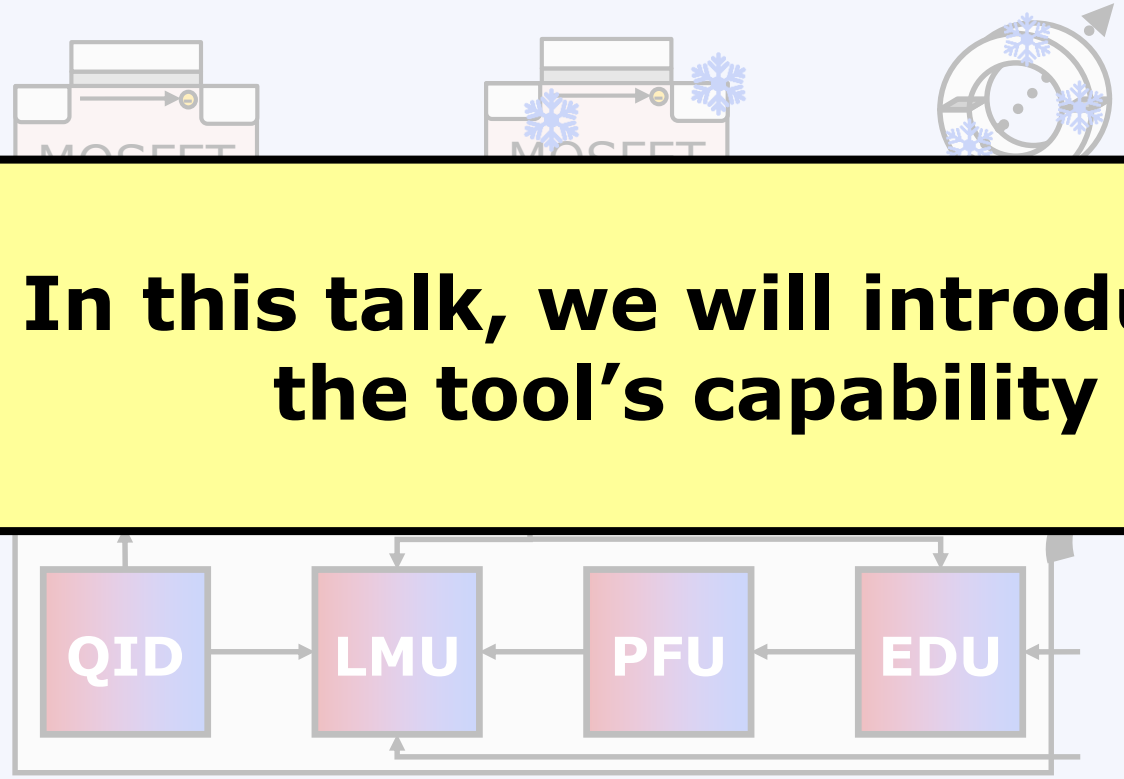
2) XQ-simulator



Manageable qubit scale and scalability bottlenecks

XQsim: Overview

1) XQ-estimator



Frequency and power
of each μarch unit

2) XQ-simulator

Cycle-accurate
simulation

Manageable qubit scale
and scalability bottlenecks

In this talk, we will introduce XQsim by demonstrating the tool's capability with various examples

Tutorial outline

1. Configuration file

- Example configuration filles

2. XQ-estimator

- CMOS model demonstration
- SFQ model demonstration

3. XQ-simulator

- Quantum compiler demonstration
- Single run demonstration
- Scalability analysis demonstration

We will provide all the demonstrations with the prepared notebook file

Index

Configuration

- **XQsim requires a configuration file with three fields:**

arch_unit
⋮
PSU: μarch, temp, tech
TCU: μarch, temp, tech
EDU: μarch, temp, tech
⋮

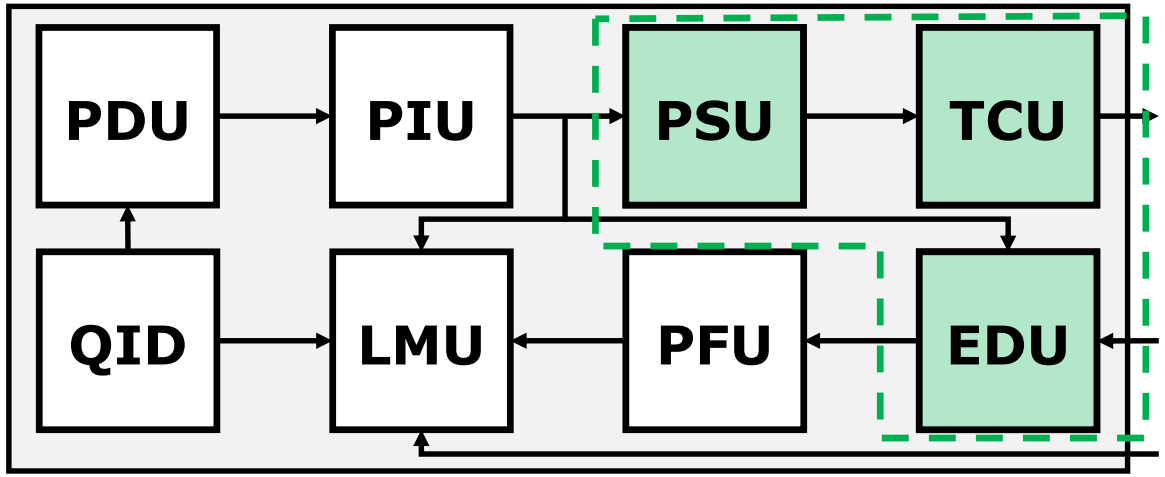
qubit_plane
code_dist
block_type
physical_error_rate

scale_constraint
gate_latency
4K_power_budget
digital_cable_heat

Configuration: arch_unit (1/2)

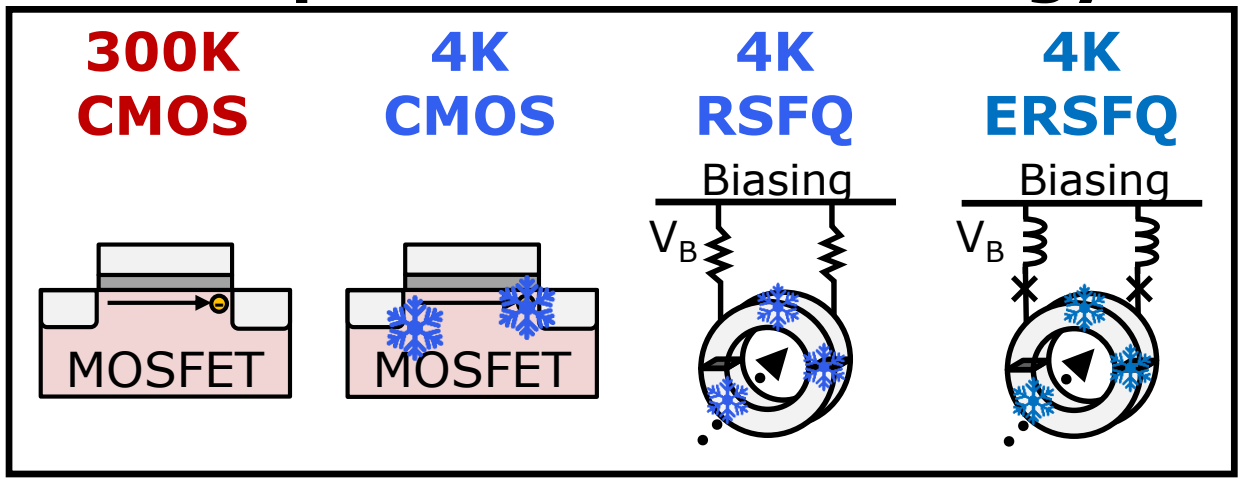
- **Define each architectural unit's configuration**
  - Microarchitecture (μarch)
    - "baseline" for every unit + opt. microarchitectures for PSU/TCU/EDU
  - Temperature & Technology (temp_tech)
    - 300K_CMOS_ / 4K_CMOS_(vopt) / 4K_RSFQ / 4K_ERSFQ

Microarchitecture



baseline **opt.**

Temperature & Technology



Configuration: arch_unit (2/2)

- **Define each architectural unit's configuration**
  - Microarchitecture (μarch)
    - "baseline" for every unit + opt. microarchitectures for PSU/TCU/EDU
  - Temperature & Technology (temp_tech)
    - 300K_CMOS_ / 4K_CMOS_(vopt) / 4K_RSFQ / 4K_ERSFQ

nearfuture_RSFQ_opt

arch_unit

QID: {μarch : baseline, temp_tech: **300K_CMOS**}

⋮

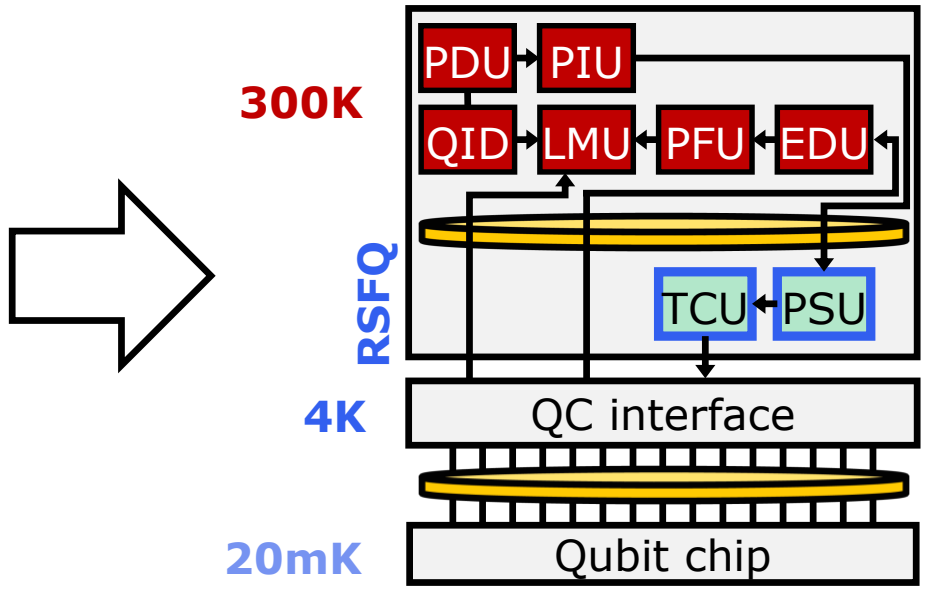
PSU: {μarch : **maskshare**, temp_tech: **4K_RSFQ**}

TCU: {μarch : **simplebuf**, temp_tech: **4K_RSFQ**}

EDU: {μarch : baseline, temp_tech: **300K_CMOS**}

⋮

LMU: {μarch : baseline, temp_tech: **300K_CMOS**}



Configuration: arch_unit (2/2)

- Define each architectural unit's configuration
  - Microarchitecture (μarch)
    - "baseline" for every unit + opt. microarchitectures for PSU/TCU/EDU
  - Temperature & Technology: (temp_tech)

We can apply different microarchitecture, temperature, and technology for each architectural unit

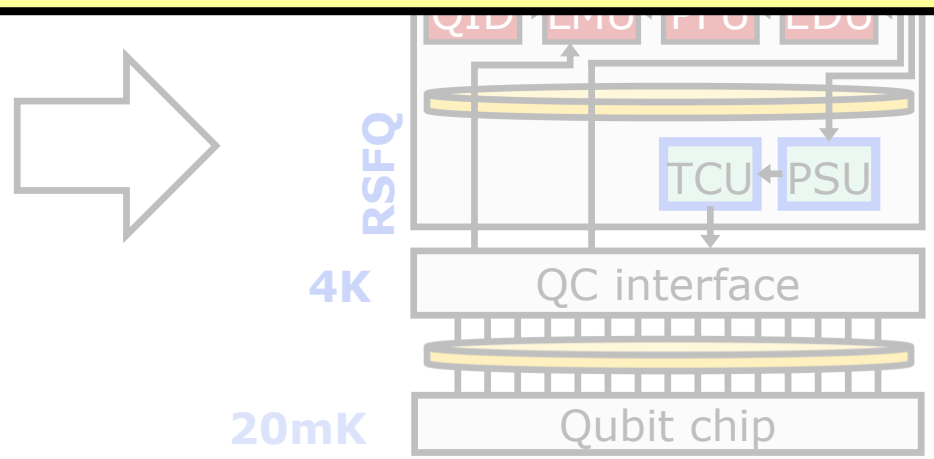
QID: {μarch : baseline, temp_tech: 300K_CMOS}

PSU: {μarch : **maskshare**, temp_tech: **4K_RSFQ**}

TCU: {μarch : **simplebuf**, temp_tech: **4K_RSFQ**}

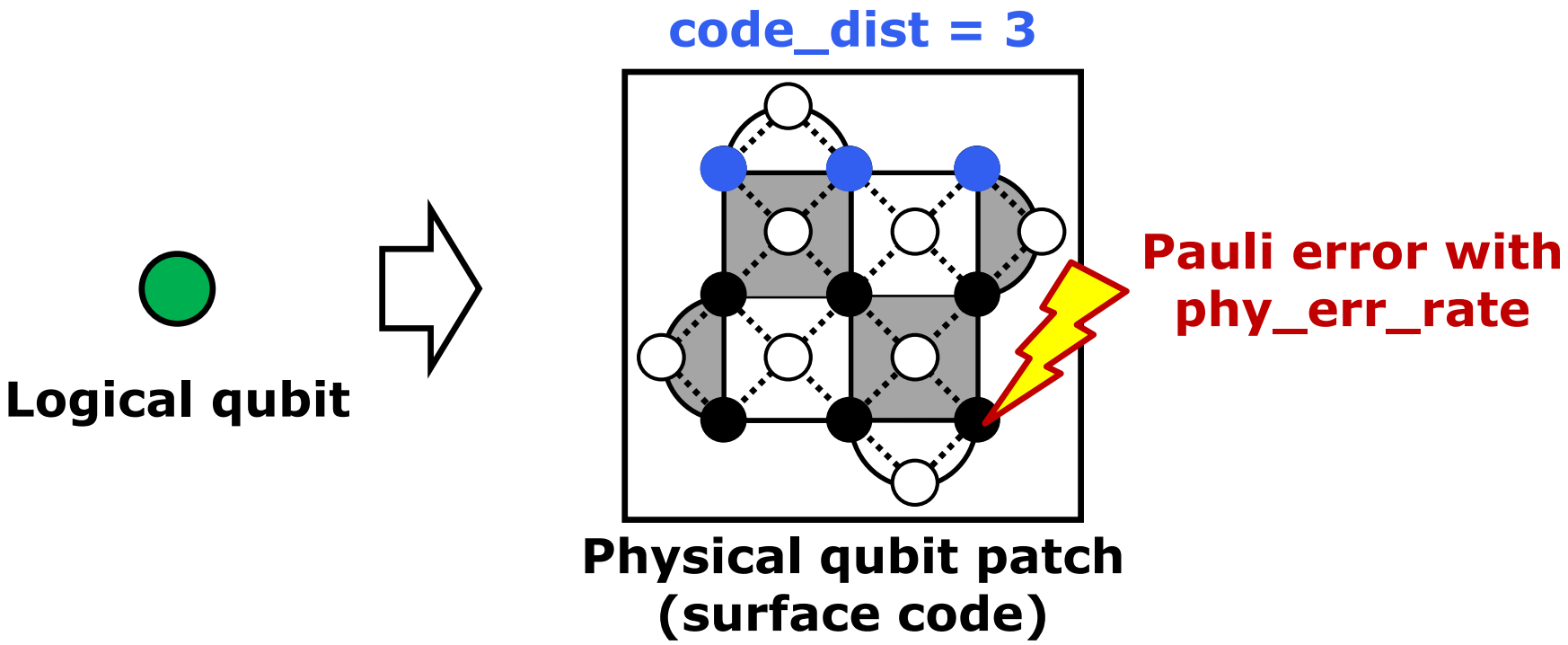
EDU: {μarch : baseline, temp_tech: **300K_CMOS**}

LMU: {μarch : baseline, temp_tech: **300K_CMOS**}



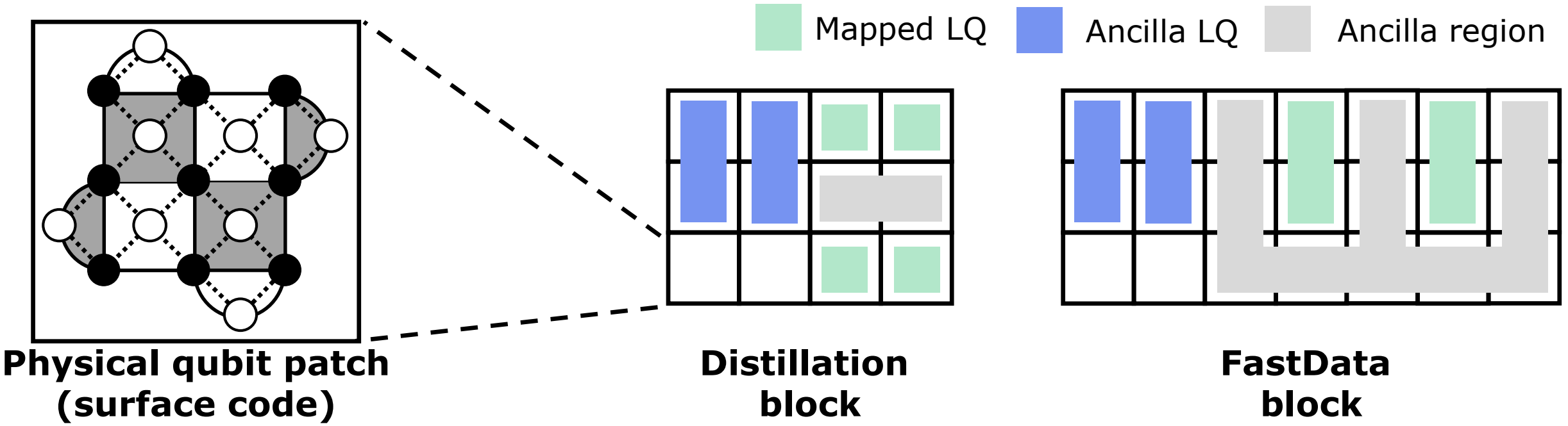
Configuration: qubit_plane (1/2)

- **Define the configurations for the physical qubit plane**
  - Surface code distance (code_dist)
  - Physical qubit error rate (phy_err_rate)



Configuration: qubit_plane (2/2)

- **Define the configurations for the physical qubit plane**
  - Code distance (code_dist)
  - Physical qubit error rate (phy_err_rate)
  - Distillation block or FastData block [1] (block_type)



[1] Litinski, Daniel. "A game of surface codes: Large-scale quantum computing with lattice surgery." Quantum 3 (2019): 128.

Configuration: scale_constraint

- **Define parameters related to the scalability constraints**
  - Quantum gate latency (gate_latency)
  - Power budget of refrigeration (4K_power_budget)
  - Heat dissipation of 300K-to-4K digital cable (digital_cable_heat)

gate_latency		
sqgate_ns	tqgate_ns	meas_ns
14ns [2]	26ns [2]	600ns [2]
4K_power_budget	**digital_cable_heat**	
1.5W [3]	3.1mW/Gbps [4]	

Reference scale_constraint

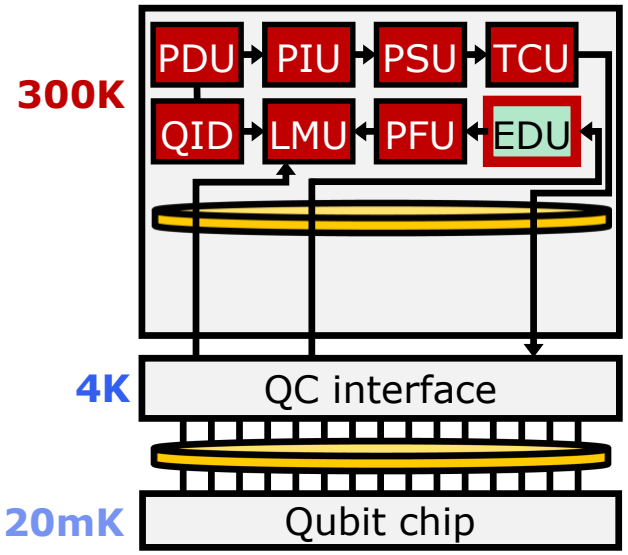
Inst. BW
Error decoding latency

4K device power
300K-to-4K data transfer

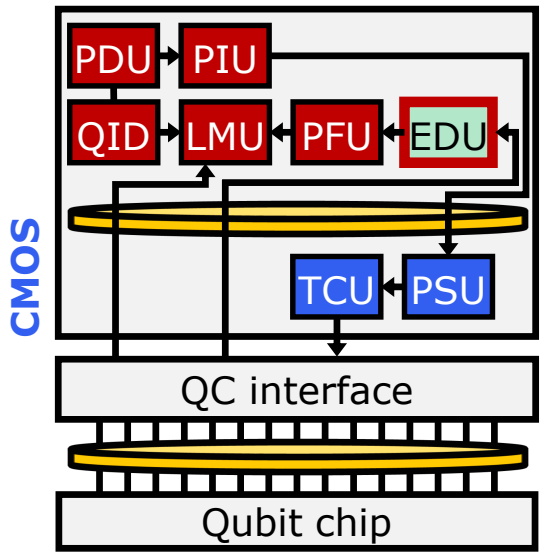
[2] Chen, Zijun, et al. "Exponential suppression of bit or phase flip errors with repetitive error correction." arXiv preprint arXiv:2102.06132 (2021).
[3] Krinner, Sebastian, et al. "Engineering cryogenic setups for 100-qubit scale superconducting circuit systems." EPJ Quantum Technology 6.1 (2019): 2.
[4] Hashimoto, Yoshihito, et al. "Implementation and experimental evaluation of a cryocooled system prototype for high-throughput SFQ digital applications." IEEE transactions on applied superconductivity 17.2 (2007): 546-551.

Configuration: Example

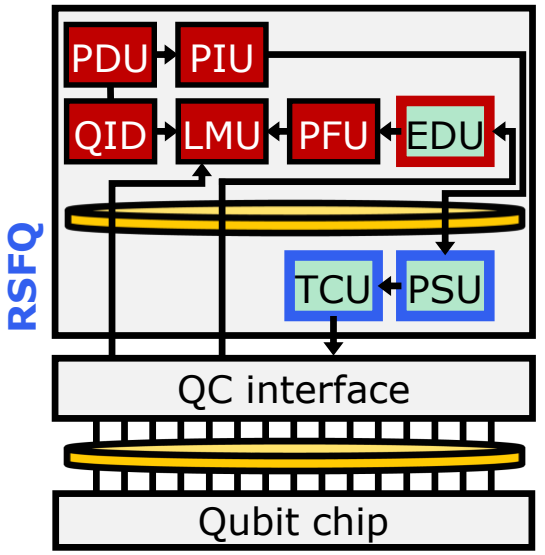
- **Let's check the configurations with the notebook**
  - We will demonstrate various architecture configurations:
    - e.g., current_300kCMOS_opt / nearfuture_RSFQ_opt / nearfuture_4kCMOS_opt / future_ERSFQ_opt
  - Each configuration corresponds to the architecture presented in our paper



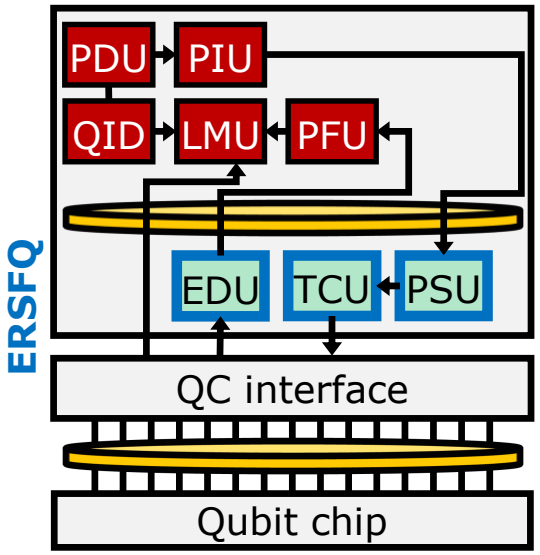
current_300kCMOS_opt
(with fast EDU)



nearfuture_4kCMOS_opt
(+ with voltage-scaled
TCU & PSU)



nearfuture_RSFQ_opt
(+ with low-power
TCU& PSU)

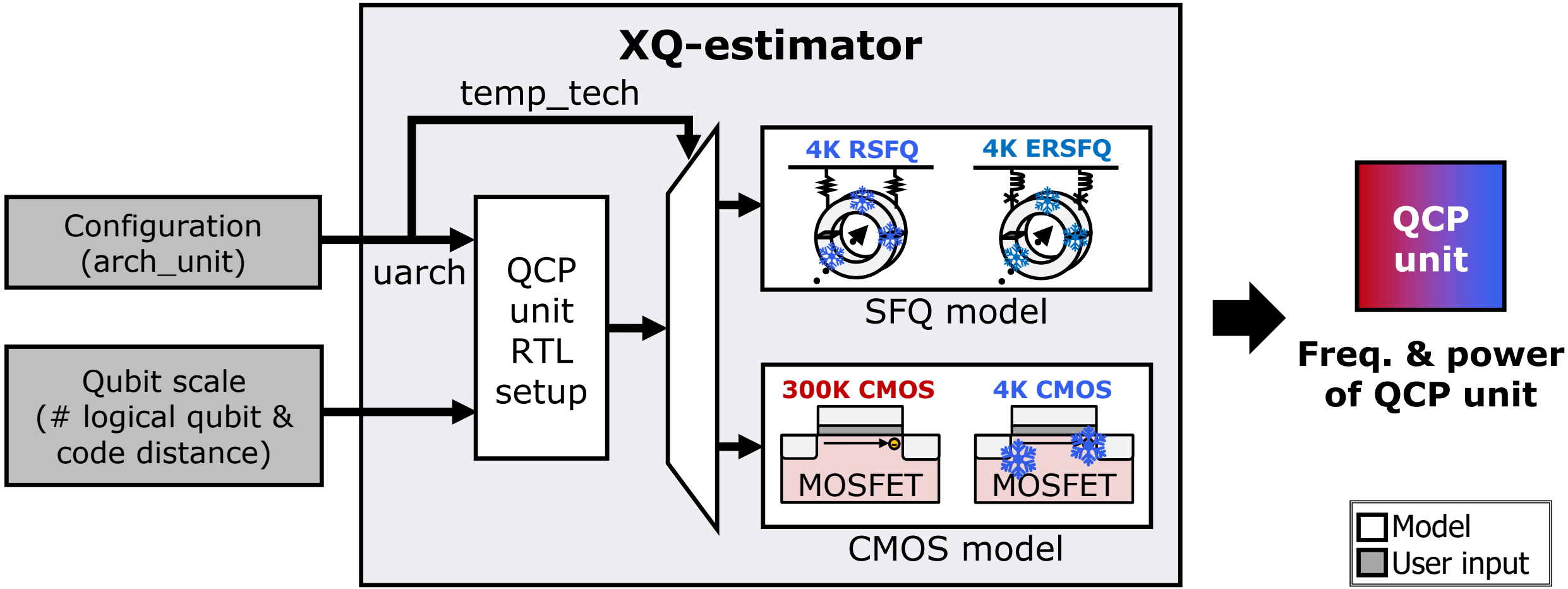


future_ERSFQ_opt
(+ with fast & low-power
EDU)

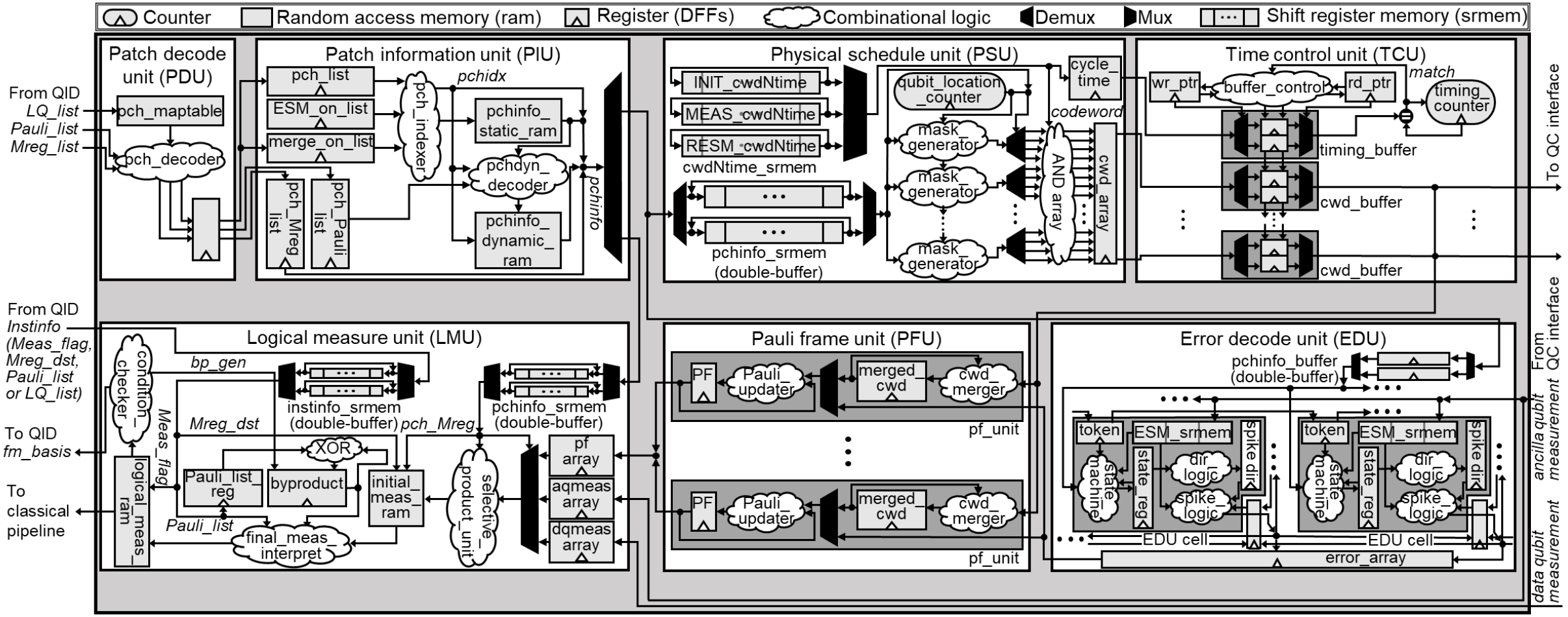
Index

XQ-estimator: Overview

- **Derive frequency & power of each QCP unit with the temp_tech. config**
- **Utilize the RTL design for each QCP unit corresponding to uarch. config**



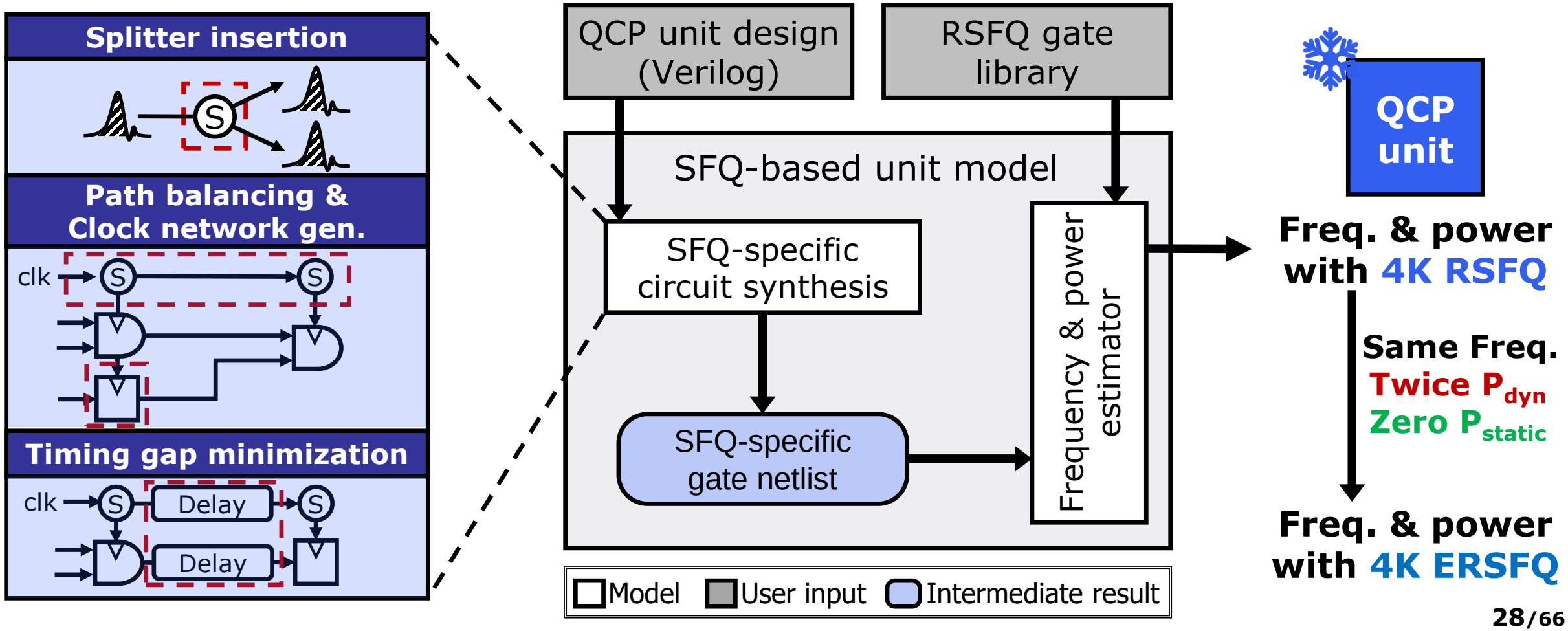
RTL implementation



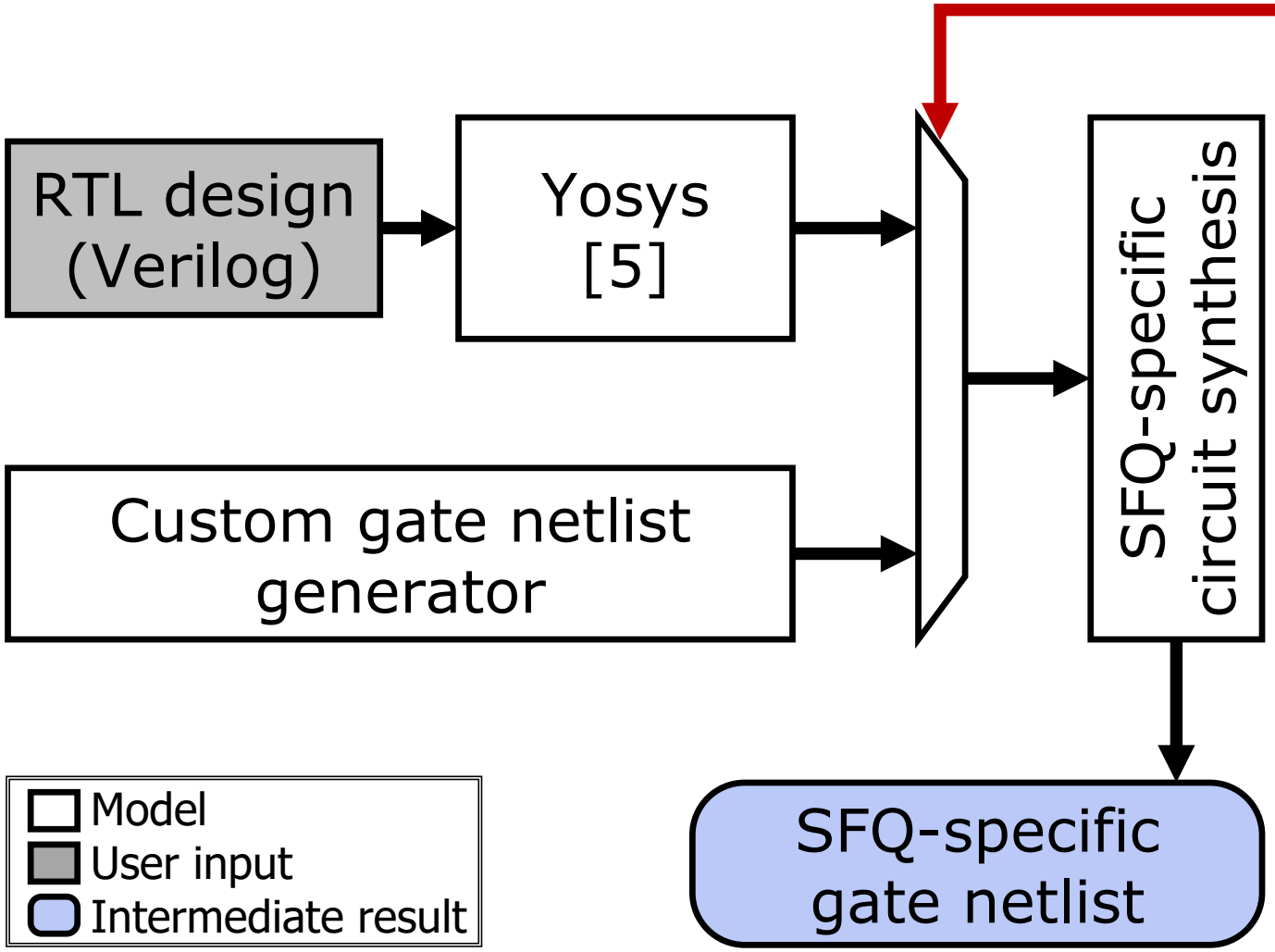
Find our RTL codes in the following directories:
XQsim/src/XQ-estimator/{unit_name}/baseline/rtl/

SFQ model

- **Generate an SFQ gate netlist by applying SFQ-specific circuit features**
- **Estimate the frequency and power by using RSFQ gate library data**



SFQ model: SFQ gate netlist



munit_template

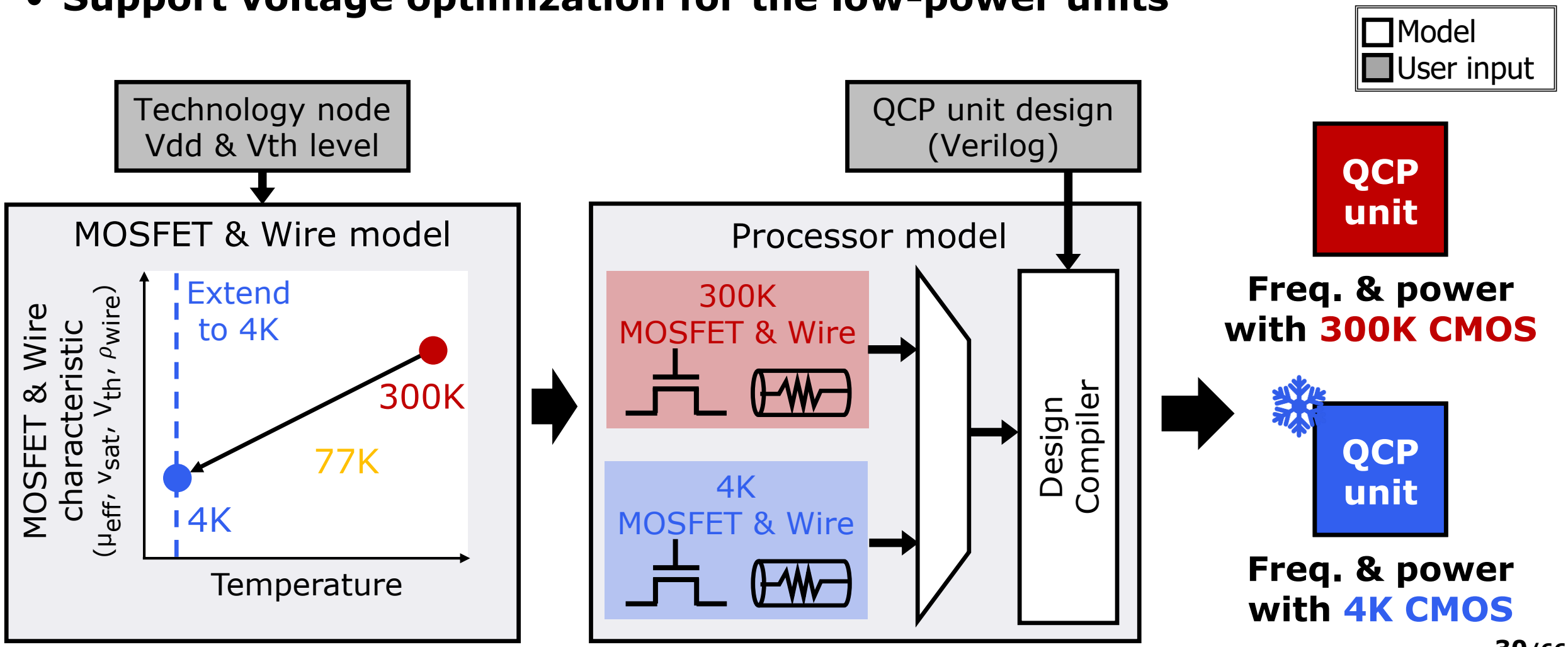
```
munit #1: {
    type: rtl or custom
    num: N,
    param: {
        param1: val1,
        param2: val2, …
    }
    clk: concurrent or counter
},
    ⋮
```

XQsim/src/XQ-estimator/{unit_name}/baseline/rsfq/

[5] Wolf, Clifford. "Yosys open synthesis suite." (2016).

CMOS model

- **Utilize our CryoModel's 300K and 4K models**
- **Support voltage optimization for the low-power units**



XQ-estimator: Demonstration

- **How to run XQ-estimator:**

1. Make an instance of xq_estimator class

```
> estimator = xq_estimator()
```

2. Put the required inputs by calling the setup function

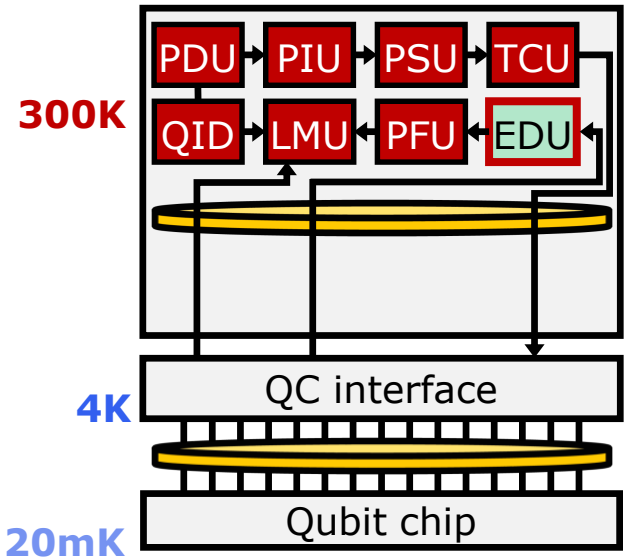
```
> estimator.setup(config, num_lq, dump, regen)
```

3. Call the run function

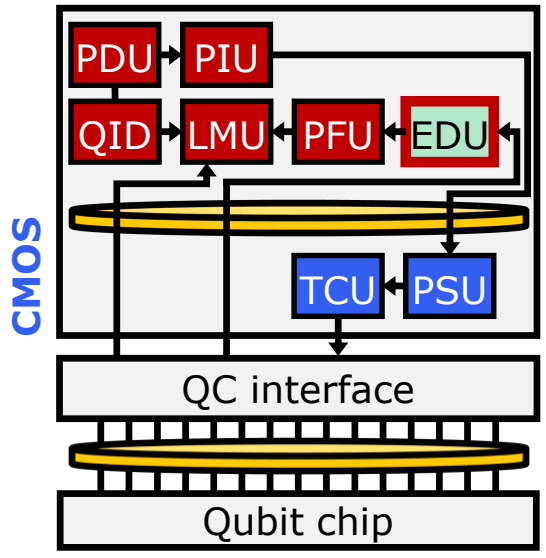
```
> estimator_res = estimator.run()
```

XQ-estimator: CMOS model demo.

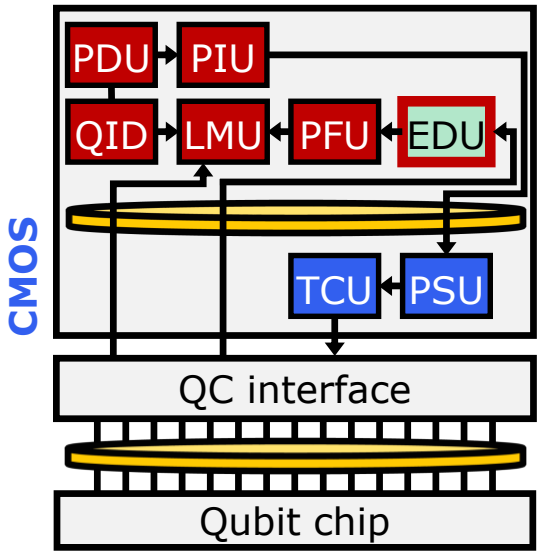
- **Run and compare the XQ-estimator's results**
  - CMOS model run with three architecture configurations



current_300kCMOS_opt
(with fast EDU)



nearfuture_4kCMOS_noopt
(No voltage scaling for 4K TCU & PSU)

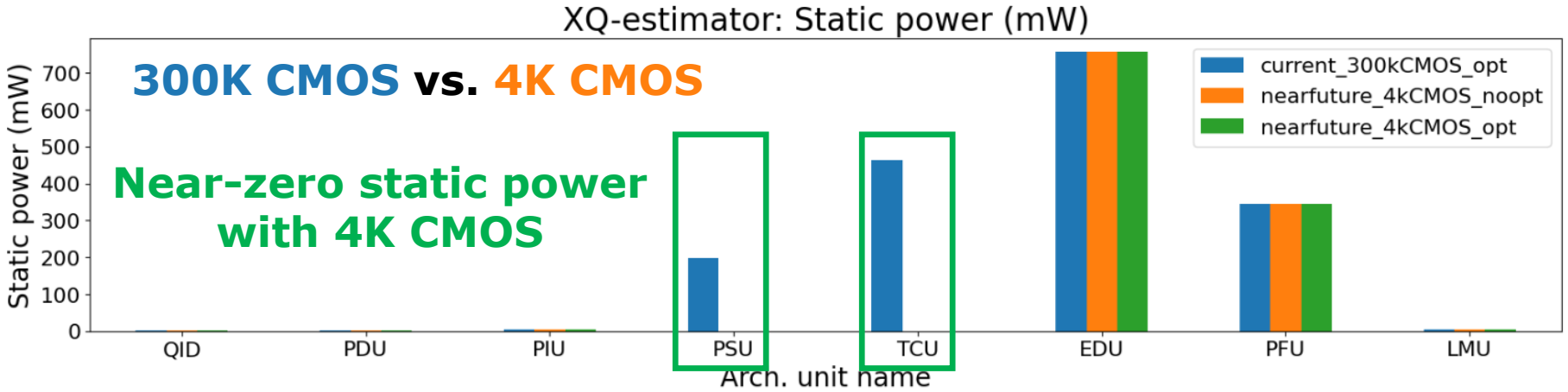


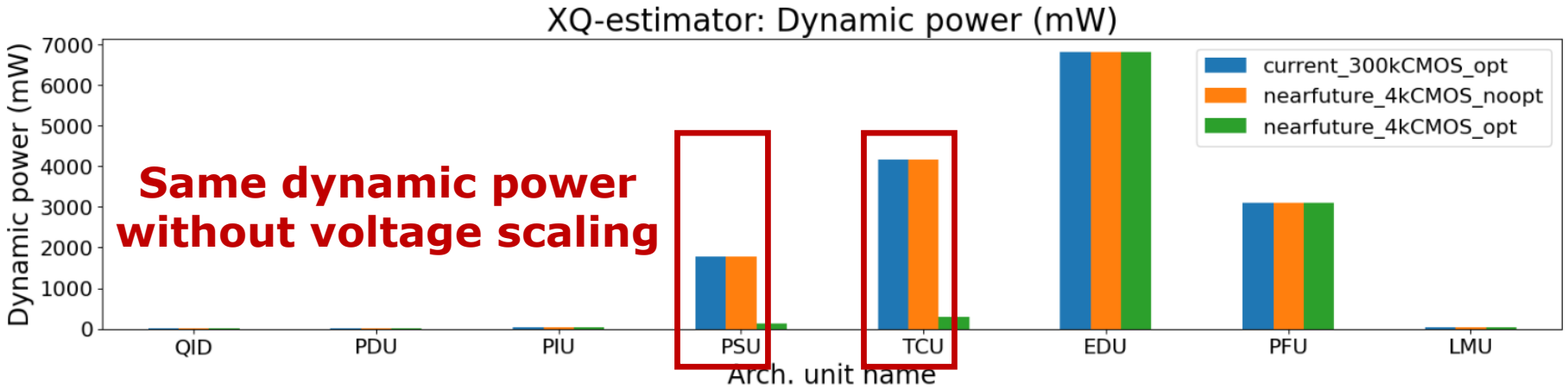
nearfuture_4kCMOS_opt
(+ with voltage-scaled 4K TCU & PSU)

Focusing on PSU & TCU, we will compare 300K CMOS vs. 4K CMOS vs. 4K CMOS Vopt

XQ-estimator: CMOS model demo.

- **Run and compare the XQ-estimator's results**
  - CMOS model run with three architecture configurations

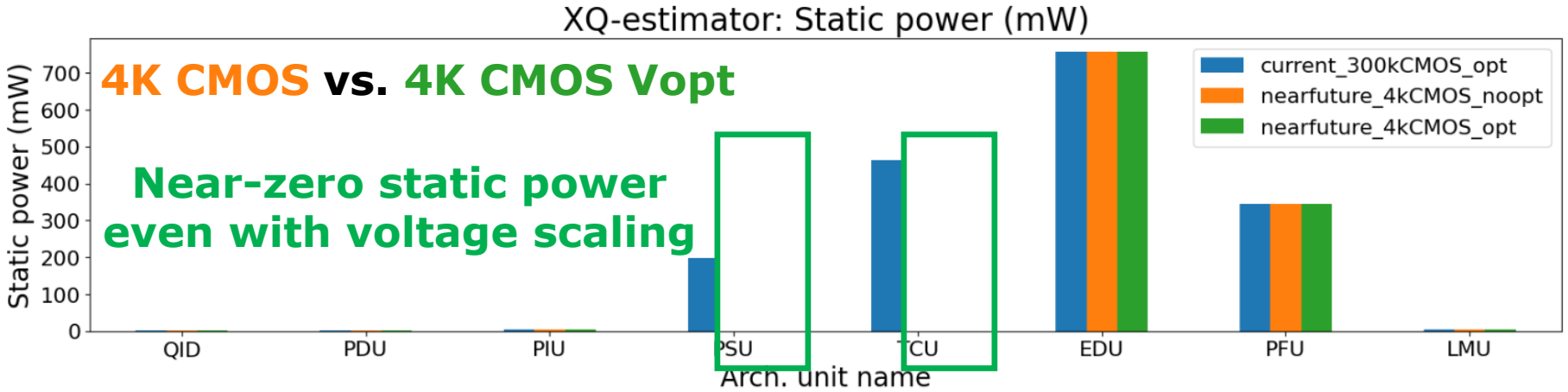


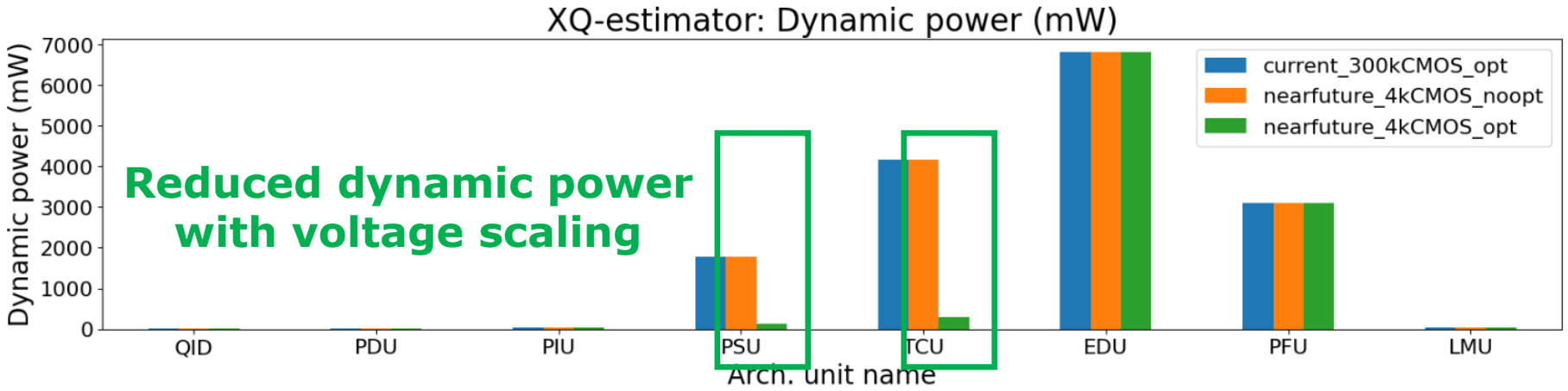


We are polishing our tool with the final check-up, and thus the detailed values can be changed. Please find our final slide with the tool release.

XQ-estimator: CMOS model demo.

- **Run and compare the XQ-estimator's results**
  - CMOS model run with three architecture configurations

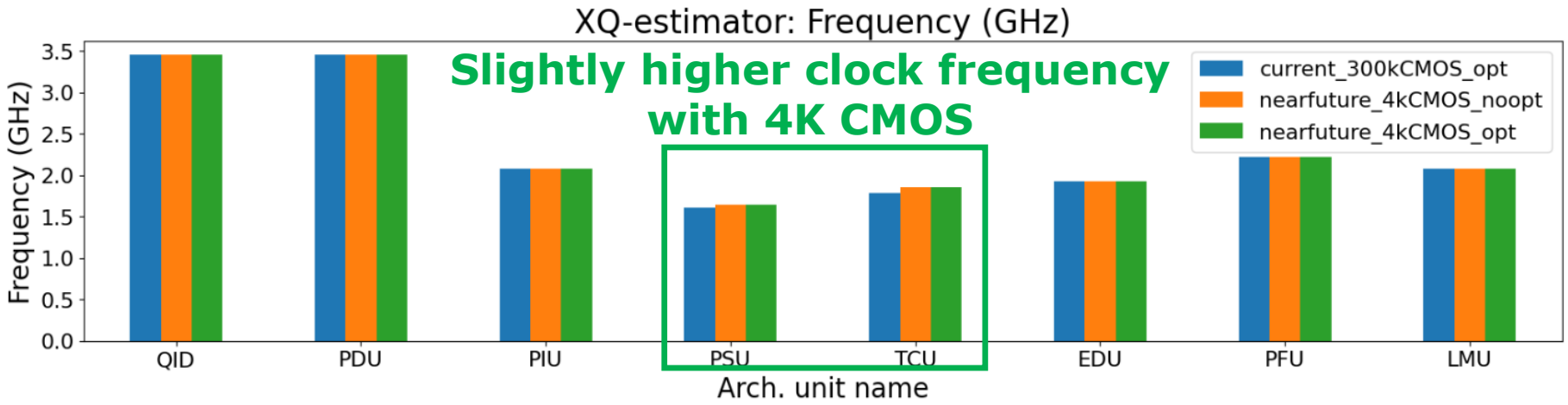




We are polishing our tool with the final check-up, and thus the detailed values can be changed. Please find our final slide with the tool release.

XQ-estimator: CMOS model demo.

- **Run and compare the XQ-estimator's results**
  - CMOS model run with three architecture configurations

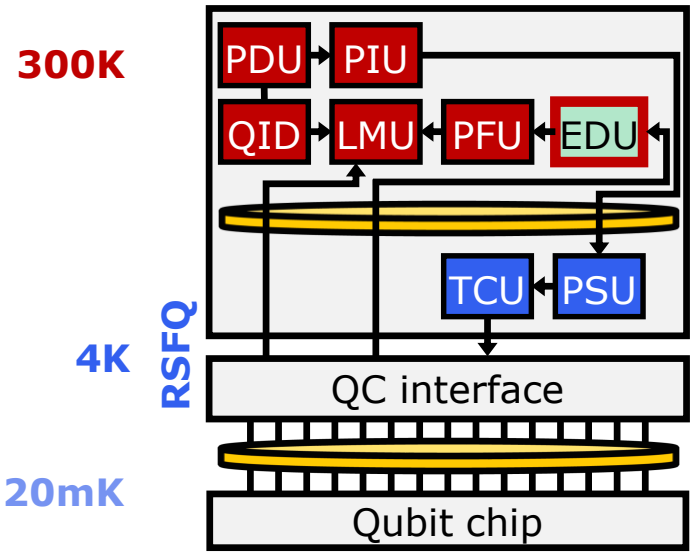


300K CMOS vs. 4K CMOS vs. 4K CMOS Vopt

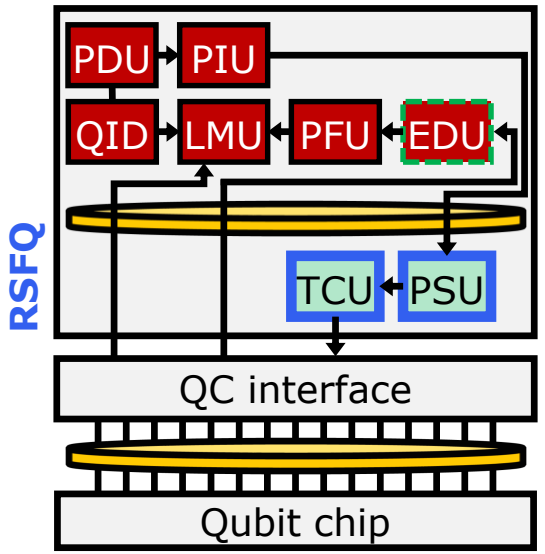
We are polishing our tool with the final check-up, and thus the detailed values can be changed. Please find our final slide with the tool release.

XQ-estimator: SFQ model demo.

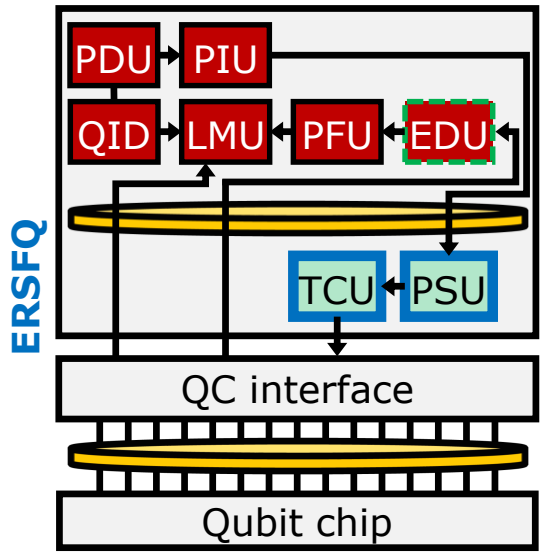
- **Run and compare the XQ-estimator's results**
  - SFQ model run with three architecture configurations



nearfuture_RSFQ_noopt



nearfuture_RSFQ_opt
(+ with low-power TCU & PSU μarch)

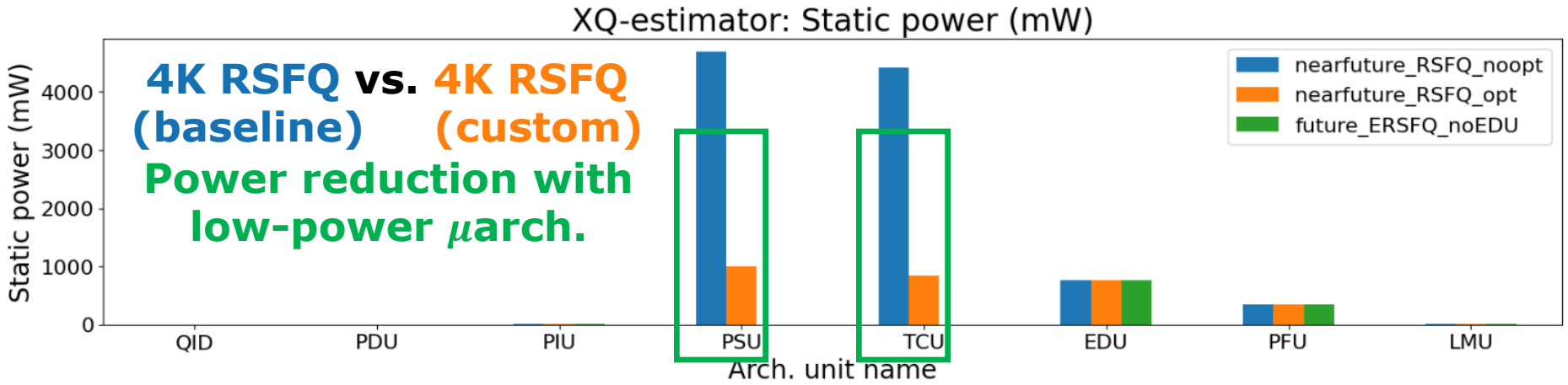


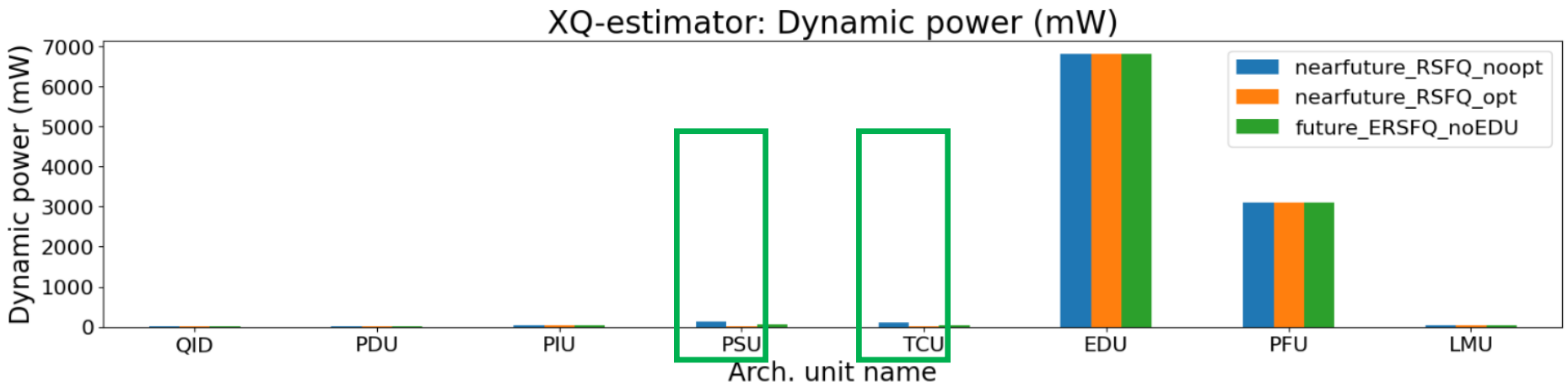
future_ERSFQ_noEDU
(+ with low-power TCU& PSUμarch)

Focusing on PSU & TCU, we will compare
4K RSFQ (baseline) vs. 4K RSFQ (custom) vs. 4K ERSFQ (custom)

XQ-estimator: SFQ model demo.

- **Run and compare the XQ-estimator's results**
  - SFQ model run with three architecture configurations

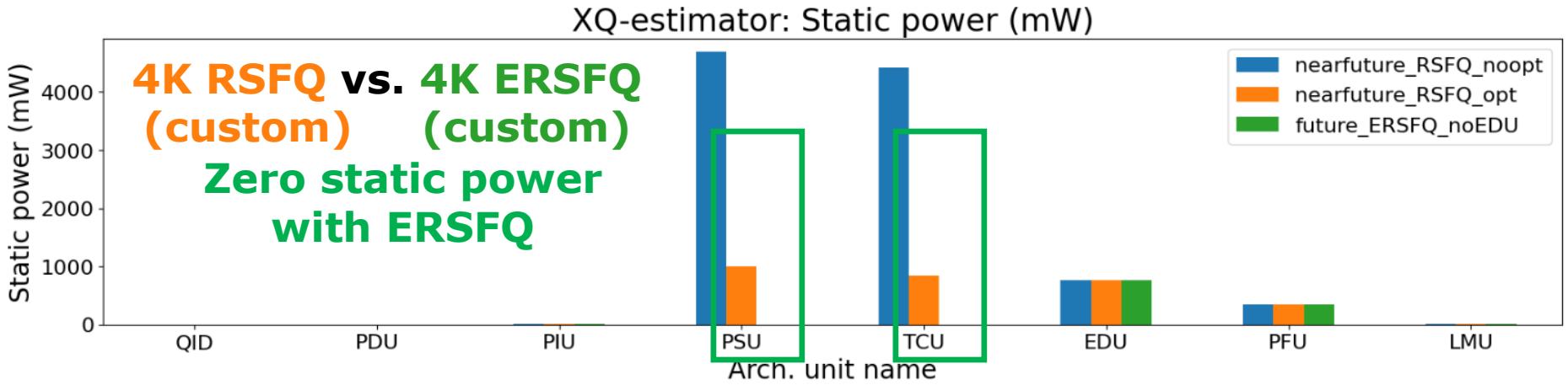


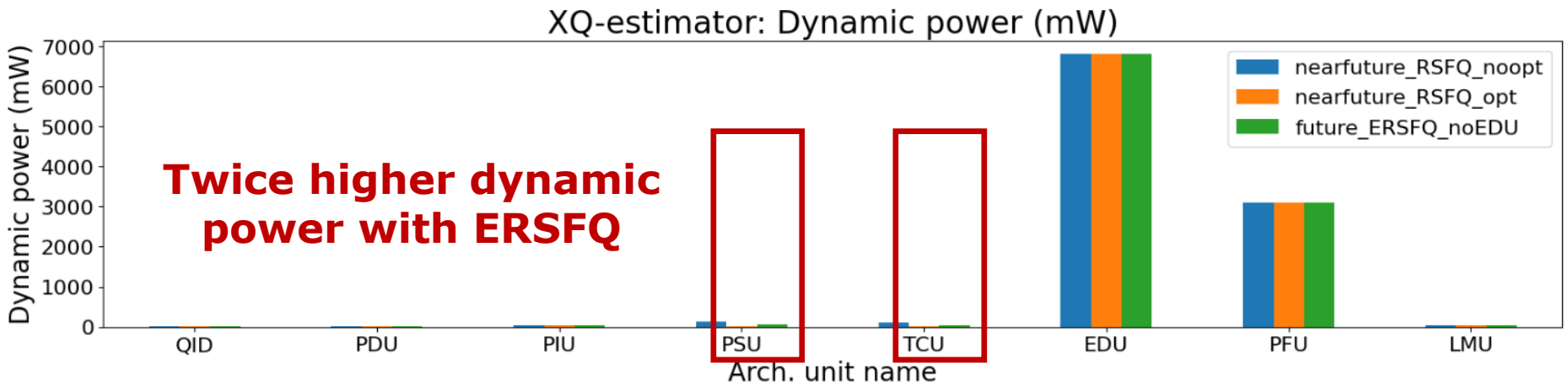


We are polishing our tool with the final check-up, and thus the detailed values can be changed. Please find our final slide with the tool release.

XQ-estimator: SFQ model demo.

- **Run and compare the XQ-estimator's results**
  - SFQ model run with three architecture configurations



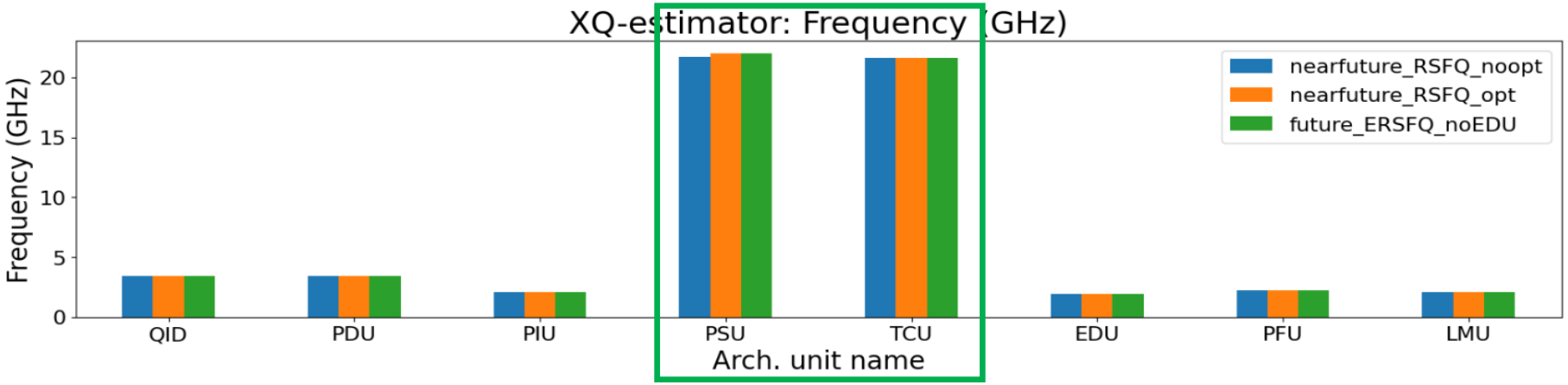


We are polishing our tool with the final check-up, and thus the detailed values can be changed. Please find our final slide with the tool release.

XQ-estimator: SFQ model demo.

- **Run and compare the XQ-estimator's results**
  - SFQ model run with three architecture configurations

Almost same frequency



4K RSFQ (baseline) vs. 4K RSFQ (custom) vs. 4K ERSFQ (custom)

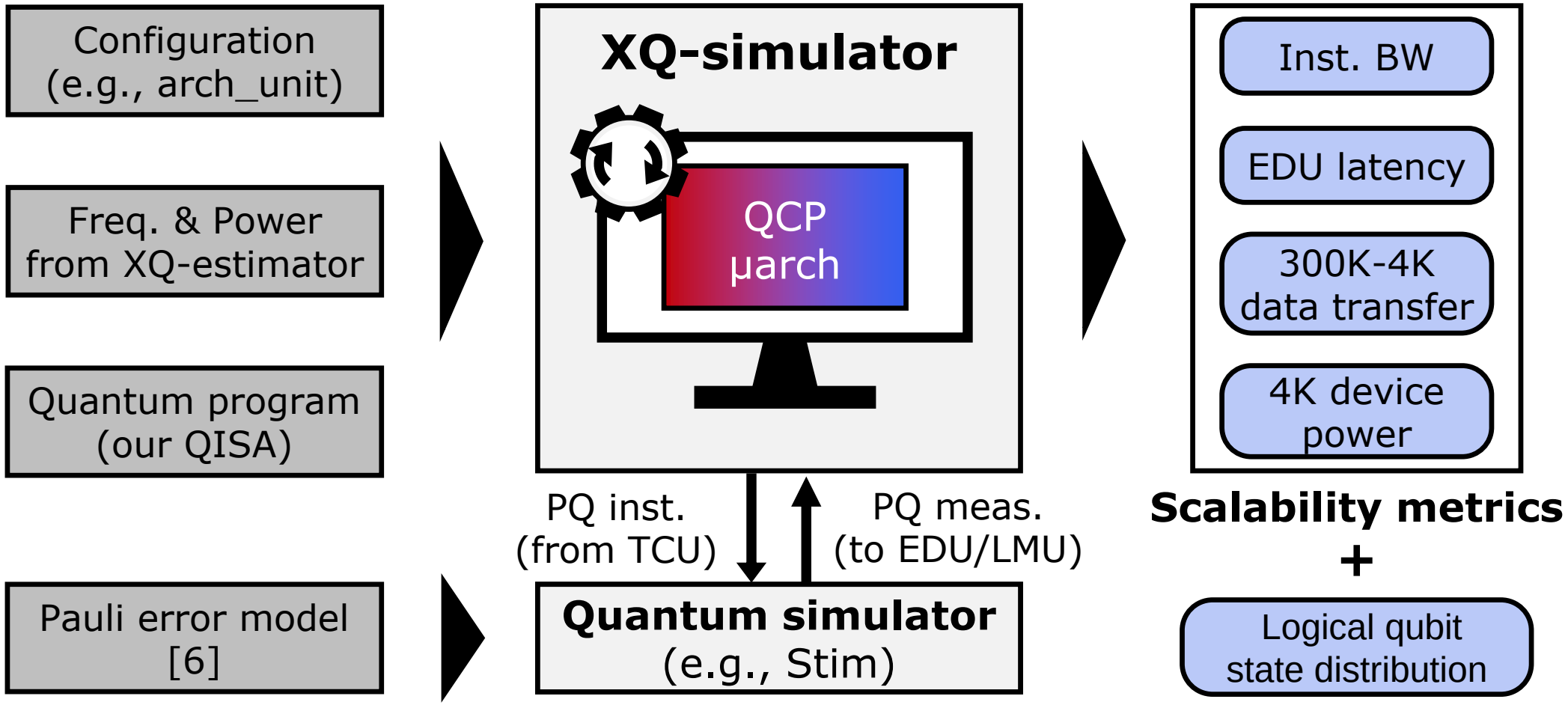
We are polishing our tool with the final check-up, and thus the detailed values can be changed. Please find our final slide with the tool release.

Index

- Motivation & Outline
- Configuration
- XQ-estimator
- **XQ-simulator**
  - Quantum compiler demonstration
  - Single run demonstration
  - Scalability analysis demonstration
- Summary

XQ-simulator: Overview

- **Run simulation to report scalability metrics and manageable qubit scale**
- **Integrate a quantum simulator for the functionally correct simulation**

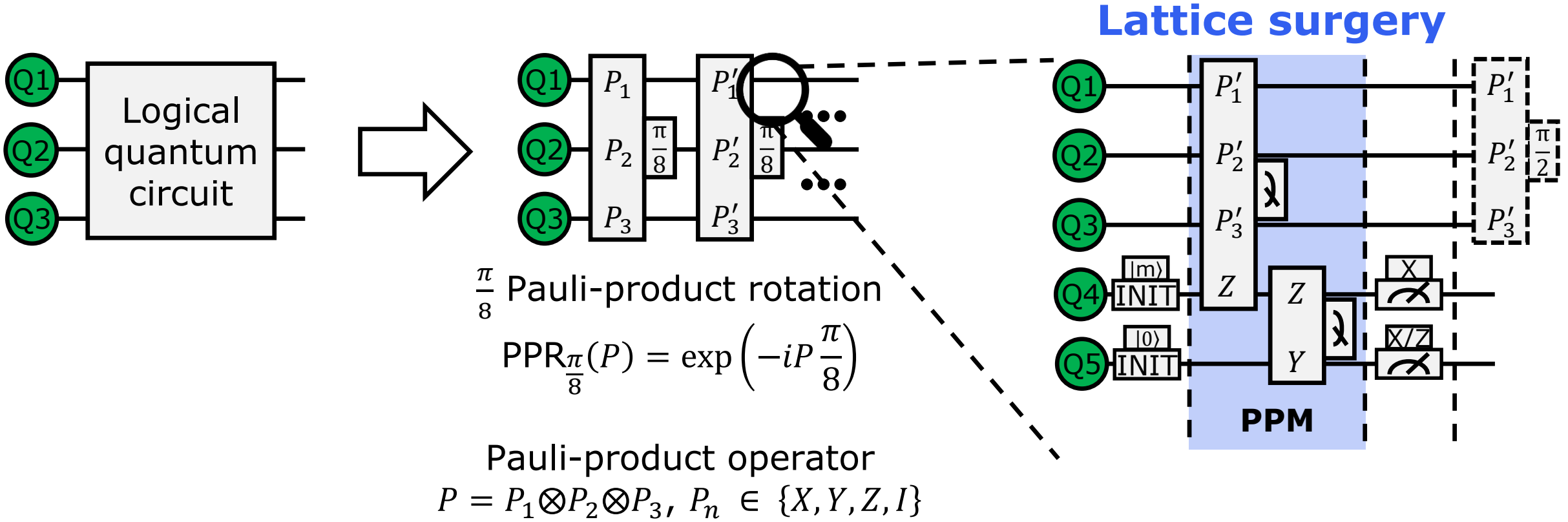


[6] Tomita, Yu, and Krysta M. Svore. "Low-distance surface codes under realistic quantum noise." Physical Review A 90.6 (2014): 062320.

Index

Quantum compiler & ISA

1. Translate logical quantum circuits to the sequence of PPM [1]
   - We support an arbitrary OpenQASM circuit as our compiler's input



[1] Litinski, Daniel. "A game of surface codes: Large-scale quantum computing with lattice surgery." Quantum 3 (2019): 128.

Quantum compiler & ISA

2. Compile the sequence PPM with our custom ISA

- You can find our isa definition at “XQsim/src/isa_format.json”

Opcode [63:60]	Meas_flag [59:54]	Mreg_dst [53:41]	LQ_addr_offset [40:32]	Target [31:0]	Description
LQI			Logical qubit address offset	LQ_list (Logical qubit list)	Logical qubit initialization
MERGE_INFO			Logical qubit address offset	Pauli_list (Target Pauli product)	Patch information update for the Merge
SPLIT_INFO					Patch information update for the Split
INIT_INTMD					Intermediate data qubit initialization
MEAS_INTMD					Intermediate data qubit measurement
RUN_ESM					d-round ESM execution
PPM_INTERPRET	Logical measure flag	Logical measure register destination	Logical qubit address offset	Pauli_list (Target Pauli product)	PPM result interpretation
LQM_X/Z/FM	Logical measure flag	Logical measure register destination	Logical qubit address offset	LQ_list (Logical qubit list)	Single logical qubit measurement

Quantum compiler: Demonstration

- **How to run XQ-estimator:**

1. Make an instance of xq_estimator class

```
> compiler = gsc_compiler()
```

2. Put the required inputs by calling the setup function

```
> compiler.setup(qc_name, compile_mode)
# compile_mode: transpile / qisa_compile / assemble
```

3. Call the run function

```
> compiler.run()  # Automatically store its result to
                  # the Xqsim/src/quantum_circuits
```

Quantum compiler: Demonstration

- **Example Pauli-product rotation compiled with our ISA**
  - XQsim/quantum_circuits/qisa_compiled/pprIIZZZ_n5.qisa

```
PREP_INFO       NA    NA     NA    NA
LQI             NA    NA     0x00  [T,T,T,T,T,T,T,-,-,-,-,-,-,-,-,-,-,]
RUN_ESM         NA    NA     NA    NA
MERGE_INFO      NA    NA     0x00  [Y,Z,I,I,Z,Z,Z,I,I,I,I,I,I,I,I,I,]
INIT_INTMD      NA    NA     NA    NA
RUN_ESM         NA    NA     NA    NA
PPM_INTERPRET   +FFB  0x000  0x00  [Y,Z,I,I,I,I,I,I,I,I,I,I,I,I,I,I,]
PPM_INTERPRET   +TTA  0x001  0x00  [I,Z,I,I,Z,Z,Z,I,I,I,I,I,I,I,I,I,]
MEAS_INTMD      NA    NA     NA    NA
SPLIT_INFO      NA    NA     NA    NA
RUN_ESM         NA    NA     NA    NA
LQM_X           +FFD  0x002  0x00  [-,T,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,]
LQM_FB          +FFC  0x003  0x00  [T,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,]
```

Index

XQ-simulator: Single run demo.

- **How to run XQ-simulator (XQsim):**

1. Make an instance of xqsim class

```
> framework = xqsim()
```

2. Put the required inputs by calling the setup function

```
> framework.setup(config, qbin, num_shots, dump, regen)
```

3. Call the run function

```
> simulator_res, pqsim_res = framework.run()
```

XQ-simulator: Single run

- **Scalability metrics vs. Scalability constraints**

```
Target qubit scale: 480 physical qubits

Check instruction bandwidth
Instruction bandwidth value: 126.0 Gbps
Instruction bandwidth requirement: 79.143 Gbps
SUCCESS: Instruction bandwidth requirement is satisfied

Check error decoding latency
Error decoding latency: 632.381 ns
ESM cycle latency: 1010 ns
SUCCESS: Error decoding latency constraint is satisfied

Check 4K power consumption
4K device power consumption: 0 mW
300K-to-4K data transfser's 4K heat: 246.236 mW
4K power budget: 1500 mW
SUCCESS: 4K power budget constraint is satisfied
```

We are polishing our tool with the final check-up, and thus the detailed values can be changed. Please find our final slide with the tool release.

XQ-simulator: Single run

- Scalability metrics vs. Scalability constraints

```
Target qubit scale: 480 physical qubits
```

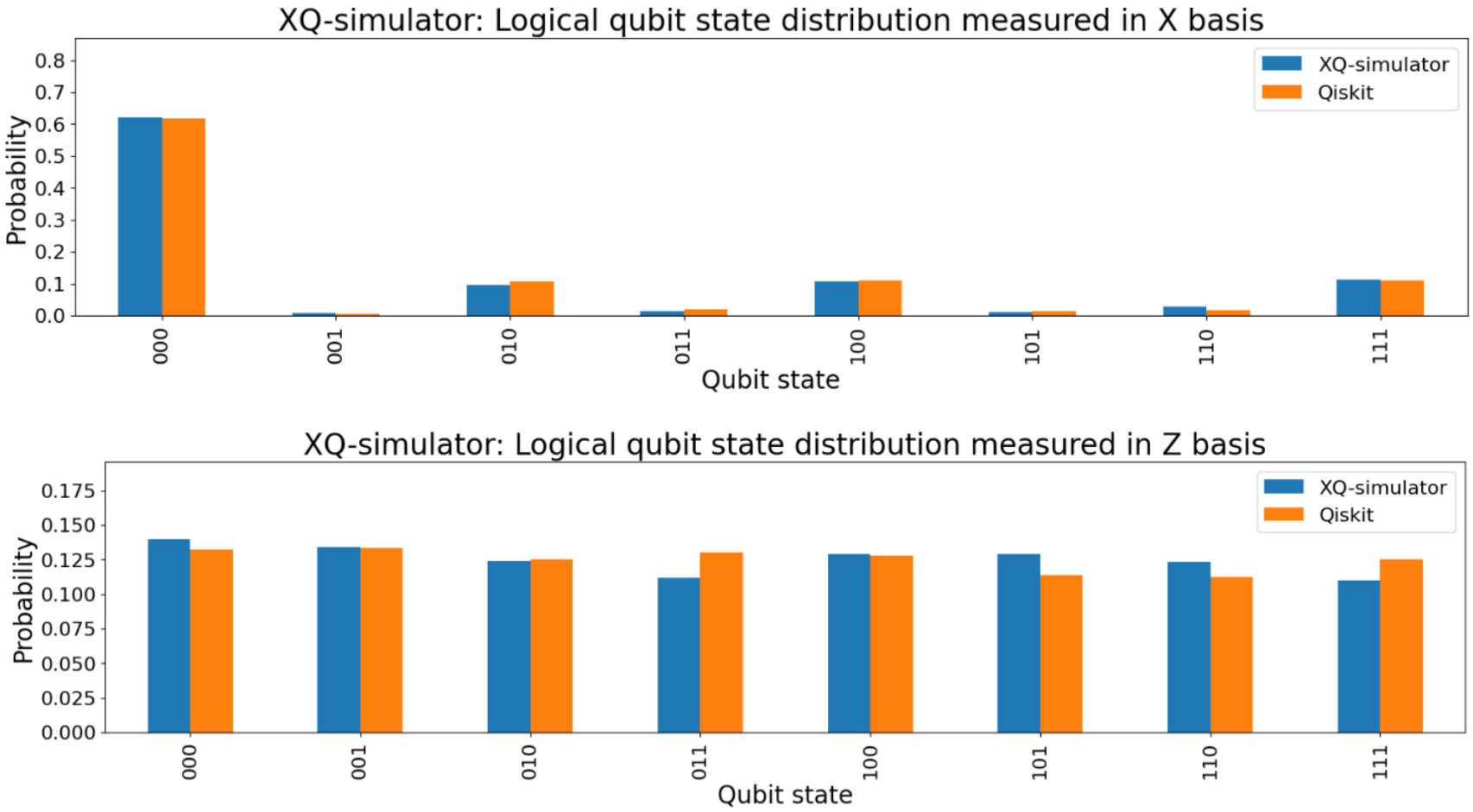
XQ-simulator can check the target QCP's manageable qubit scale with the output scalability metrics!

```
ESM cycle latency: 1010 ns
SUCCESS: Error decoding latency constraint is satisfied

Check 4K power consumption
4K device power consumption: 0 mW
300K-to-4K data transfser's 4K heat: 246.236 mW
4K power budget: 1500 mW
SUCCESS: 4K power budget constraint is satisfied
```

XQ-simulator: Single run

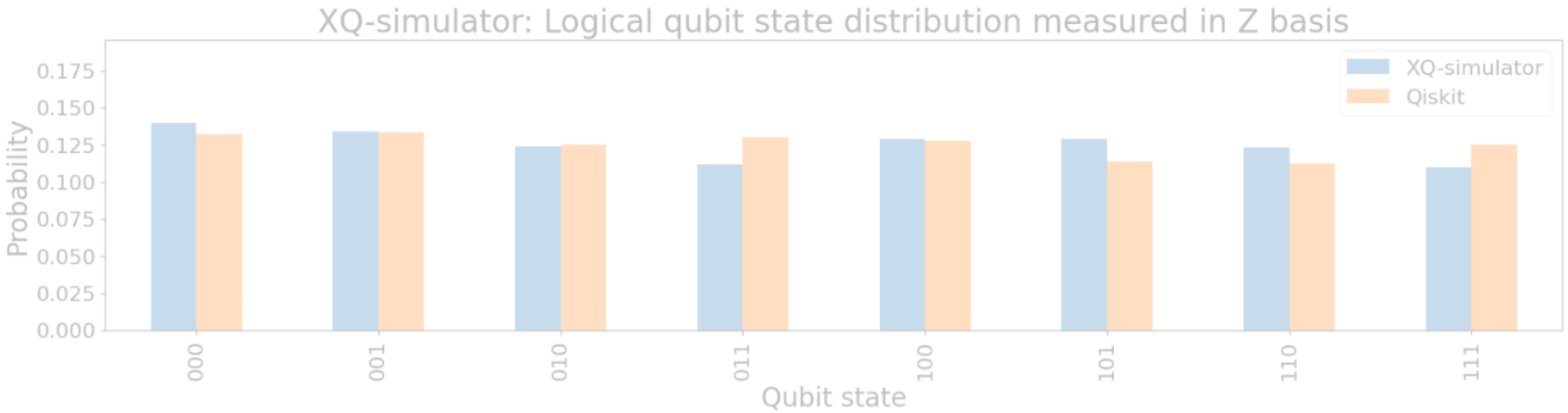
- **Logical-qubit state distribution vs. Qiskit simulation**



XQ-simulator: Single run

- Logical-qubit state distribution vs. Qiskit simulation

XQ-simulator can check its functional correctness with the logical-qubit state distribution output!



Index

XQ-simulator: Scalability analysis

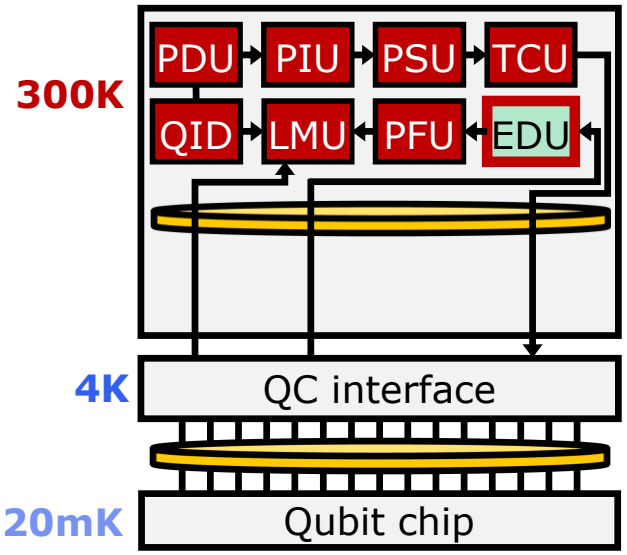
- **Run XQ-simulator with the emulation & scaling mode :**

```
> framework.setup(emulate=True, scaling=True, …)
```

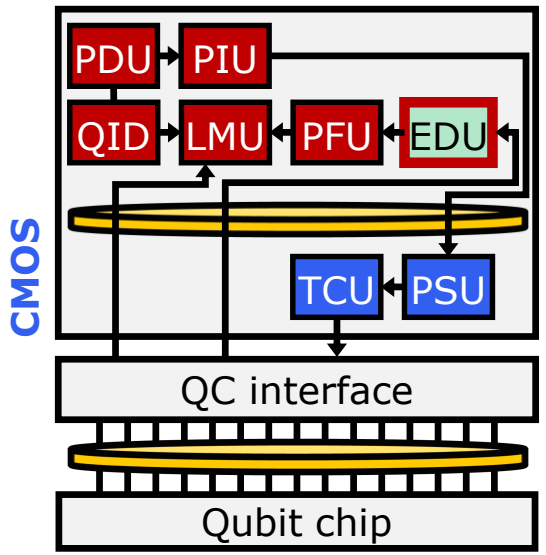
- **Emulation mode**
  - Extract the scalability metrics without running real quantum program
  - Use a maximum-size ESM as its workload
- **Scaling mode**
  - Run simulations for the target qubit scale list
  - Aggregate the scalability metric results for all the target qubit scales
- **(+) Draw graphs to facilitate the bottleneck analysis**

XQ-simulator: Scalability analysis

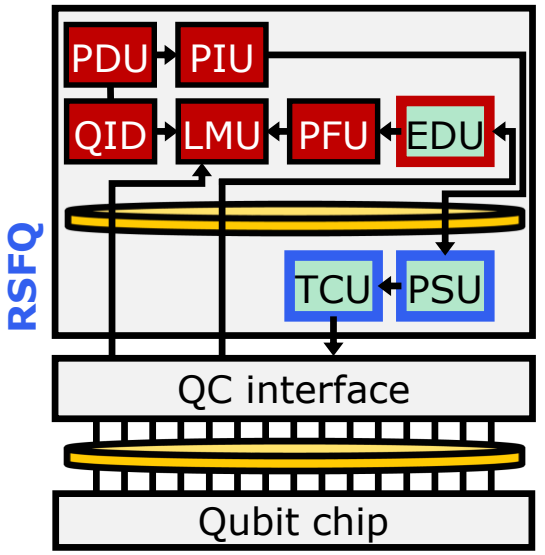
- **Analyze the scalability of four architectures:**



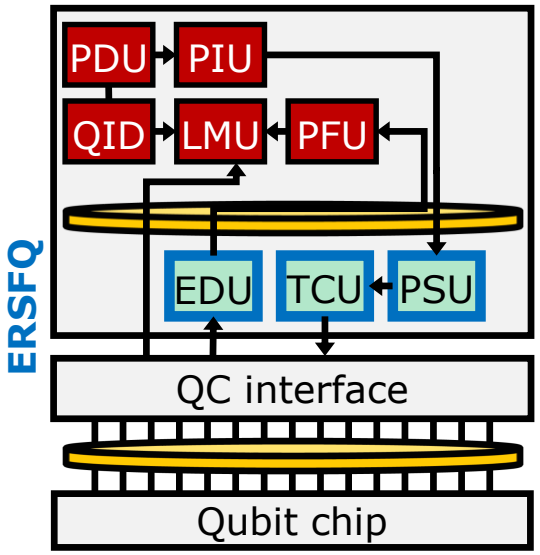
current_300kCMOS_opt
(with fast EDU)



nearfuture_4kCMOS_opt
(+ with voltage-scaled
TCU & PSU)



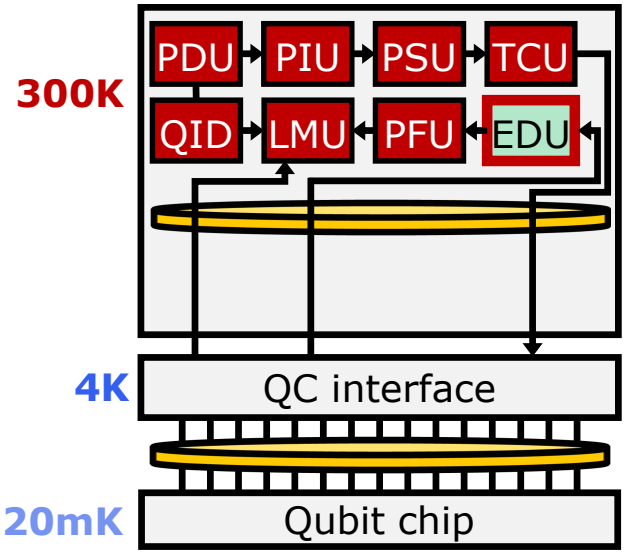
nearfuture_RSFQ_opt
(+ with low-power
TCU& PSU)



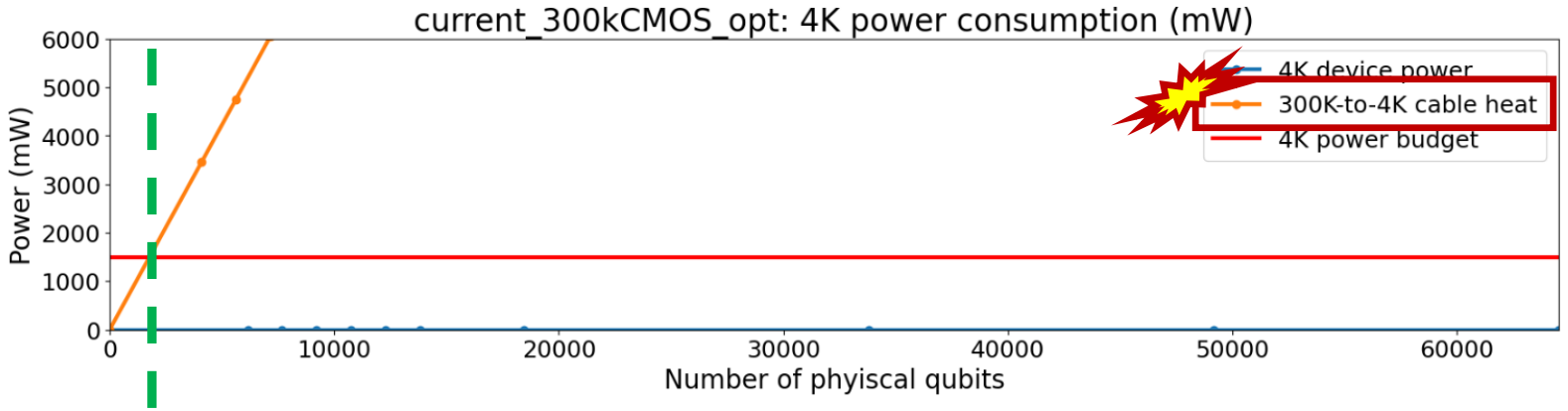
future_ERSFQ_opt
(+ with fast & low-power
EDU)

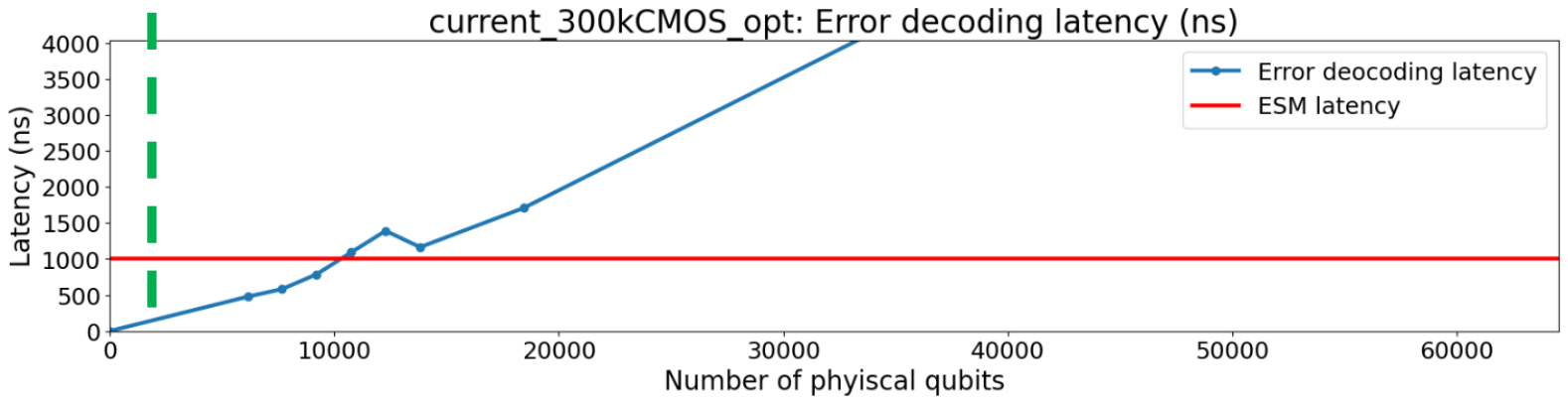
XQ-simulator: Scaling analysis

- **Config: current_300kCMOS_opt**
  - Manageable qubit scale: **1500~2000**
  - Scalability bottleneck: **300K-to-4K data transfer**



current_300kCMOS_opt
(with fast EDU)



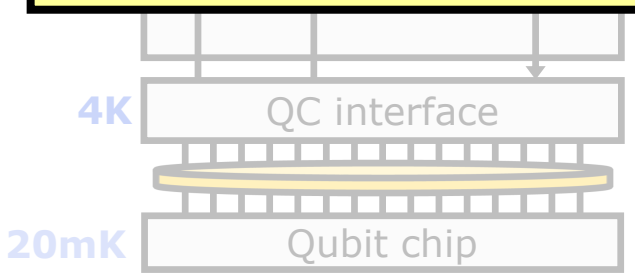


We are polishing our tool with the final check-up, and thus the detailed values can be changed. Please find our final slide with the tool release.

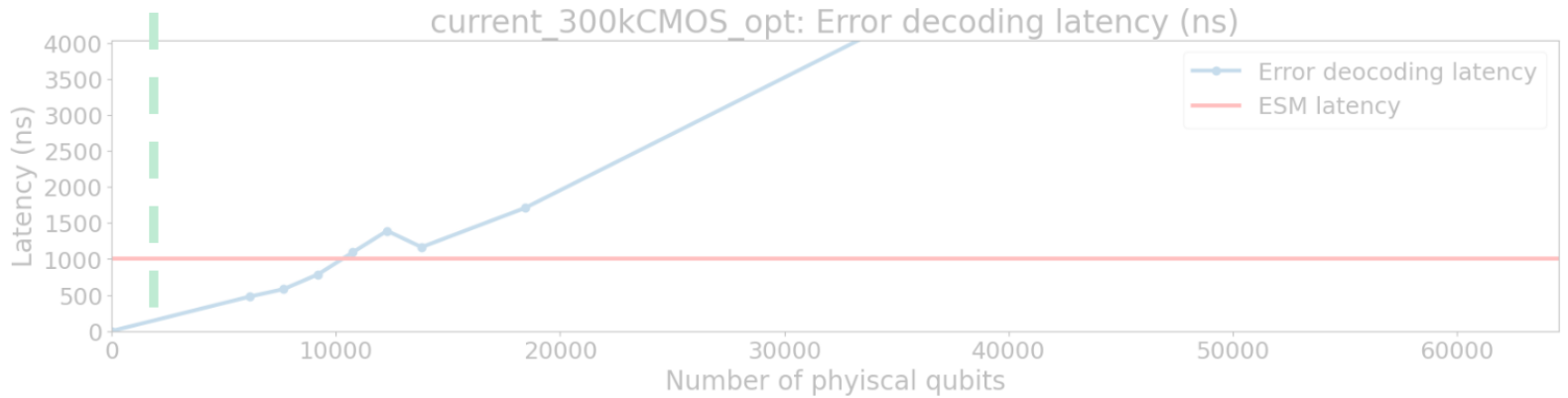
XQ-simulator: Scaling analysis

- **Config: current_300kCMOS_opt**
  - Manageable qubit scale: **1500~2000**
  - Scalability bottleneck: **300K-to-4K data transfer**

We need to move data-transfer dominating units (i.e., PSU and TCU) to the 4K domain



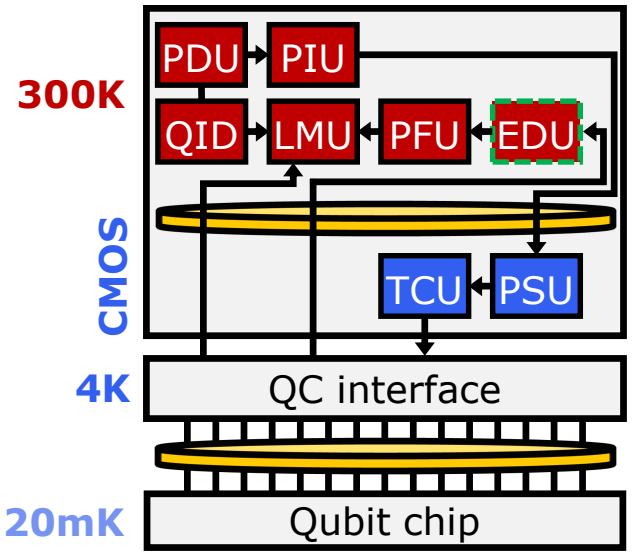
current_300kCMOS_opt
(with fast EDU)



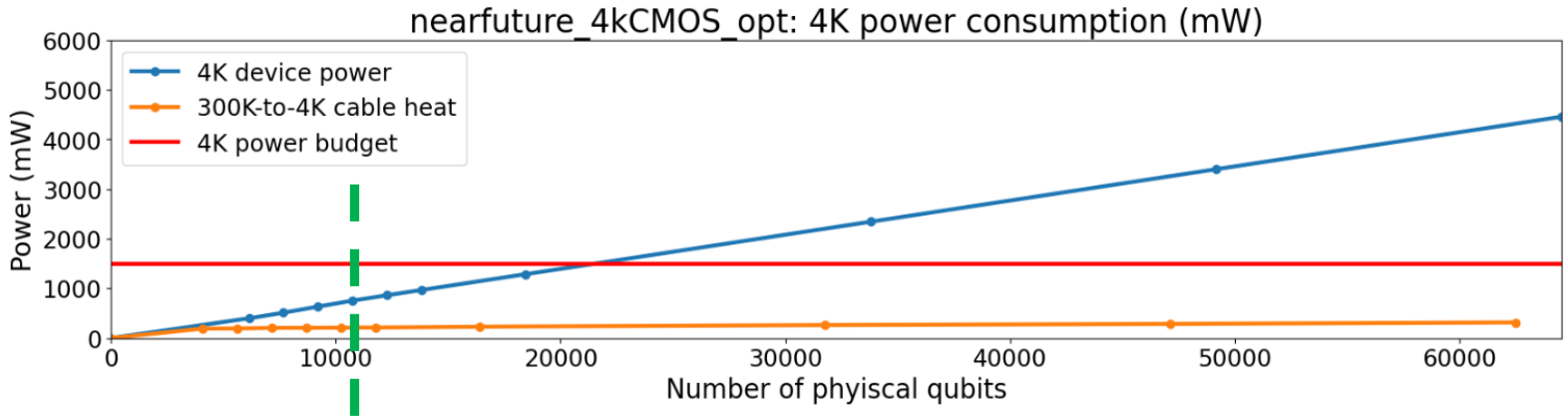
We are polishing our tool with the final check-up, and thus the detailed values can be changed. Please find our final slide with the tool release.

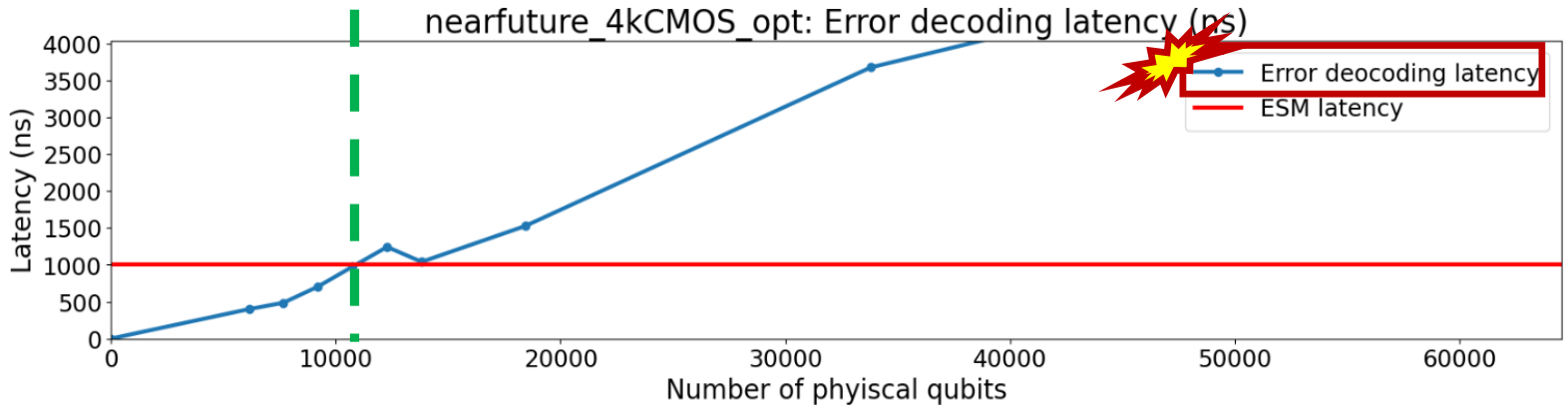
XQ-simulator: Scalability analysis

- **Config: nearfuture_4kCMOS_opt**
  - Manageable qubit scale: **~10,000 qubits**
  - Scalability bottleneck: **Error decoding latency**



nearfuture_4kCMOS_opt
(+ with voltage-scaled
TCU & PSU)



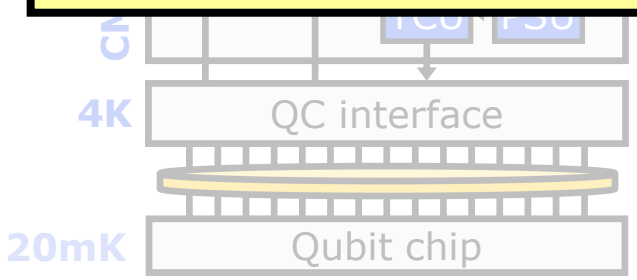


We are polishing our tool with the final check-up, and thus the detailed values can be changed. Please find our final slide with the tool release.

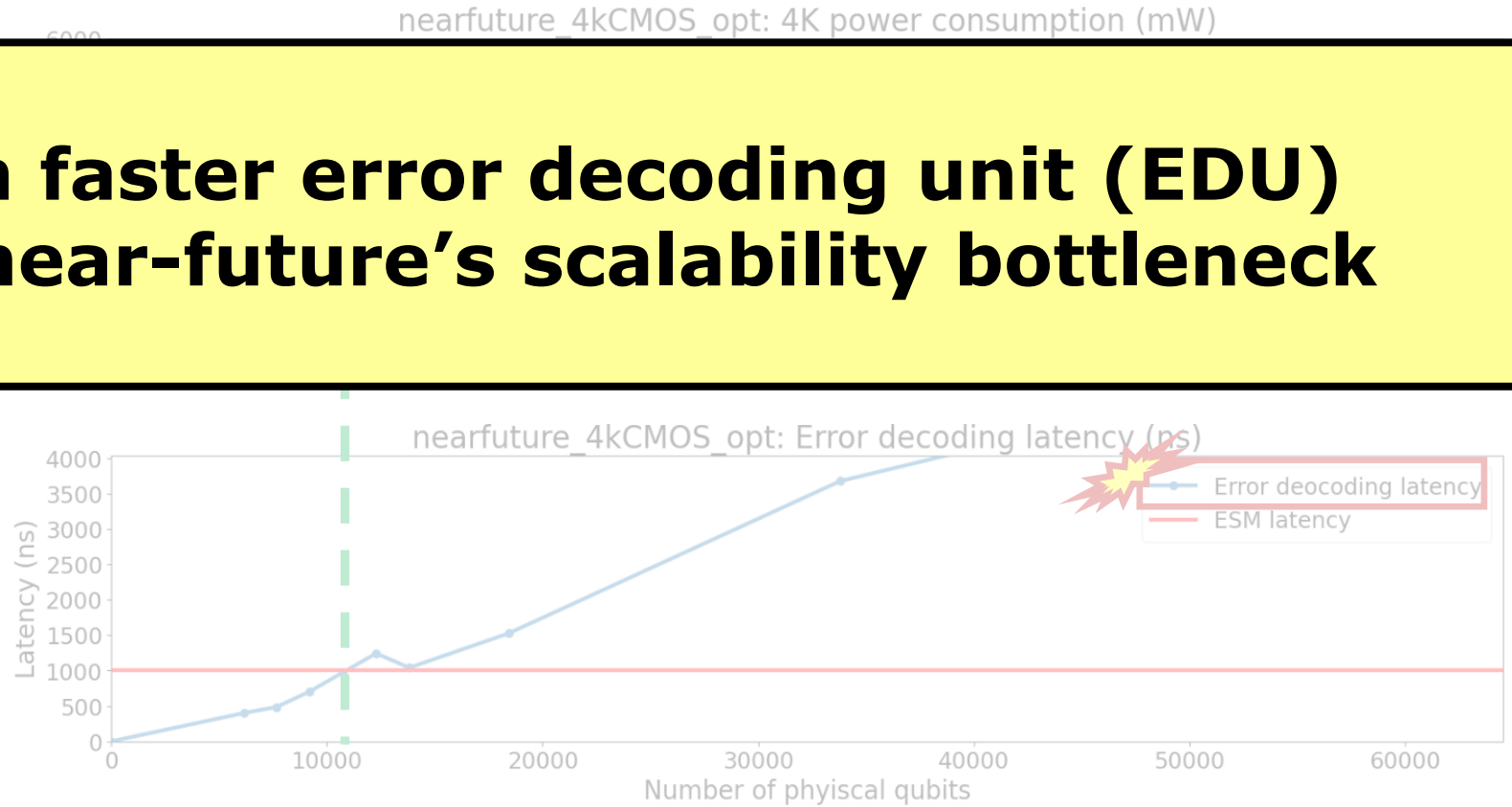
XQ-simulator: Scalability analysis

- **Config: nearfuture_4kCMOS_opt**
  - Manageable qubit scale: **~10,000 qubits**
  - Scalability bottleneck: **Error decoding latency**

We need much faster error decoding unit (EDU)
to resolve the near-future's scalability bottleneck



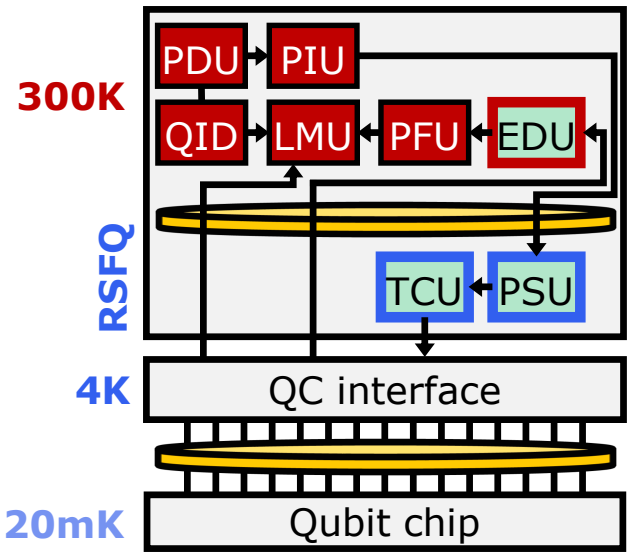
nearfuture_4kCMOS_opt
(+ with voltage-scaled
TCU & PSU)



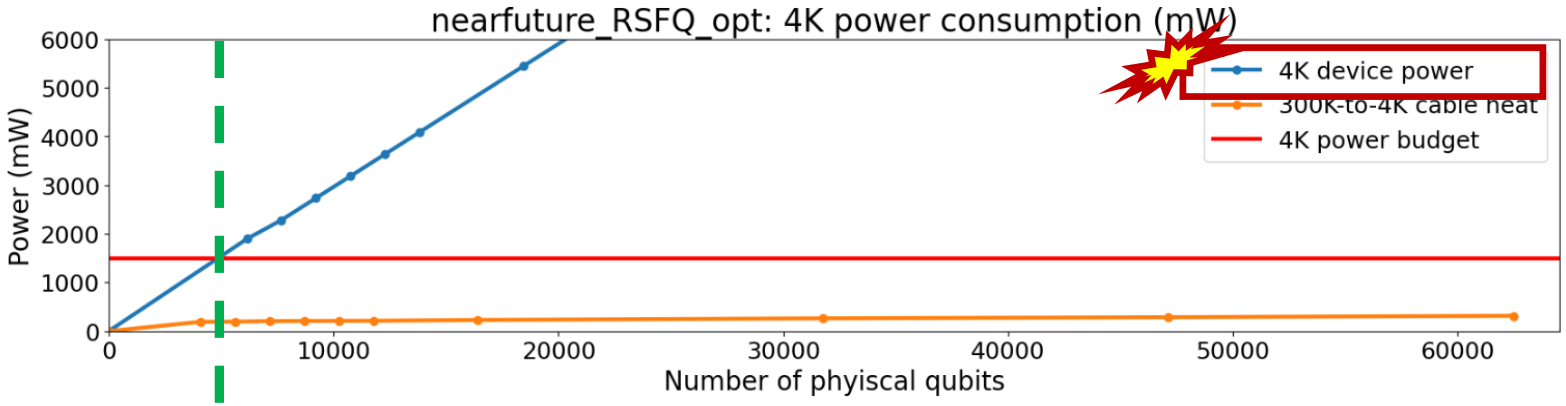
We are polishing our tool with the final check-up, and thus the detailed values can be changed. Please find our final slide with the tool release.

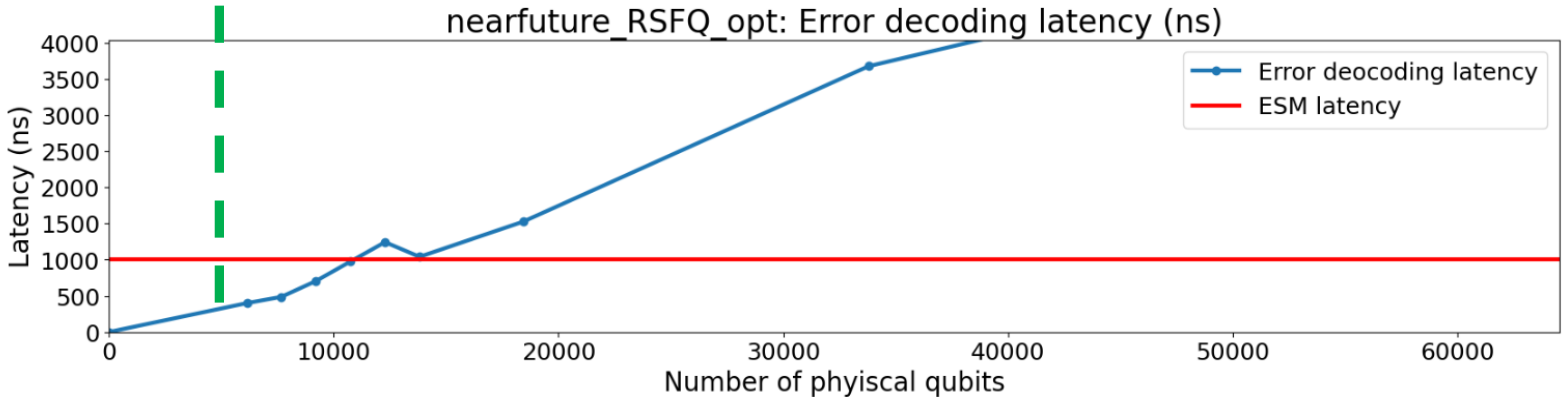
XQ-simulator: Scalability analysis

- **Config: nearfuture_RSFQ_opt**
  - Manageable qubit scale: **4000~5000 qubits**
  - Scalability bottleneck: **4K device power**



nearfuture_RSFQ_opt
(+ with low-power
TCU& PSU)



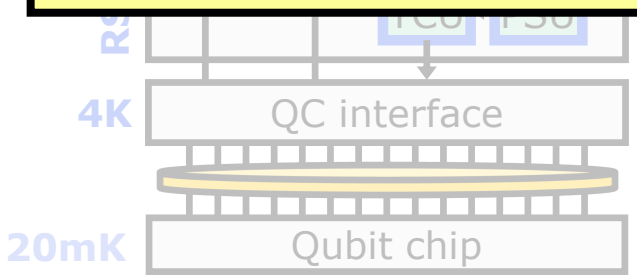


We are polishing our tool with the final check-up and thus the detailed values can be changed. Please find our final slide with the tool release.

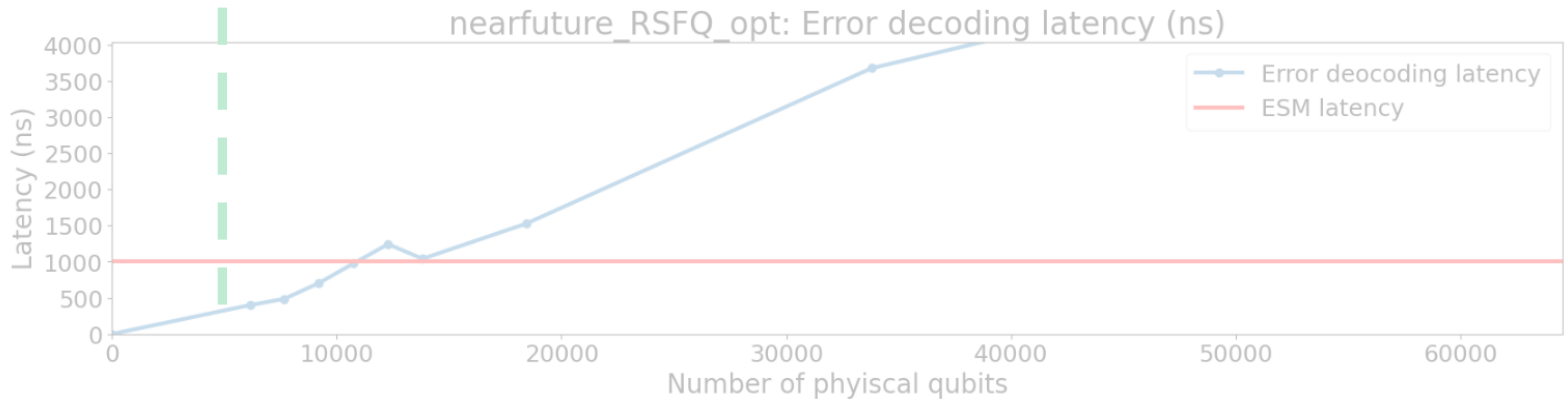
XQ-simulator: Scalability analysis

- **Config: nearfuture_RSFQ_opt**
  - Manageable qubit scale: **4000~5000 qubits**
  - Scalability bottleneck: **4K device power**

We need much lower power 4K units
to resolve the near-future's scalability bottleneck



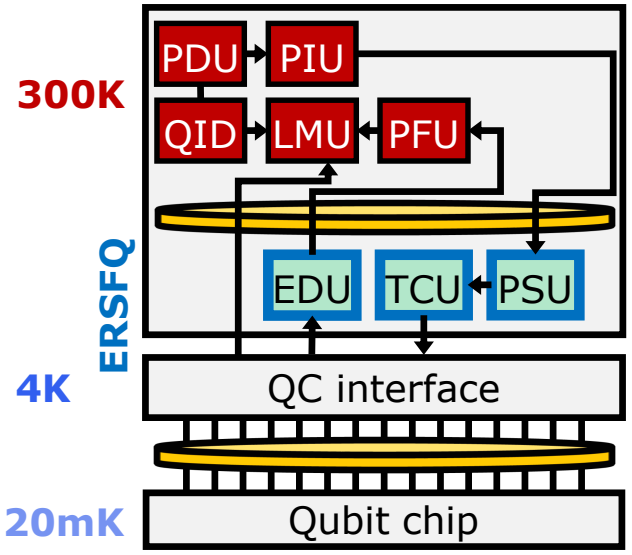
nearfuture_RSFQ_opt
(+ with low-power
TCU& PSU)



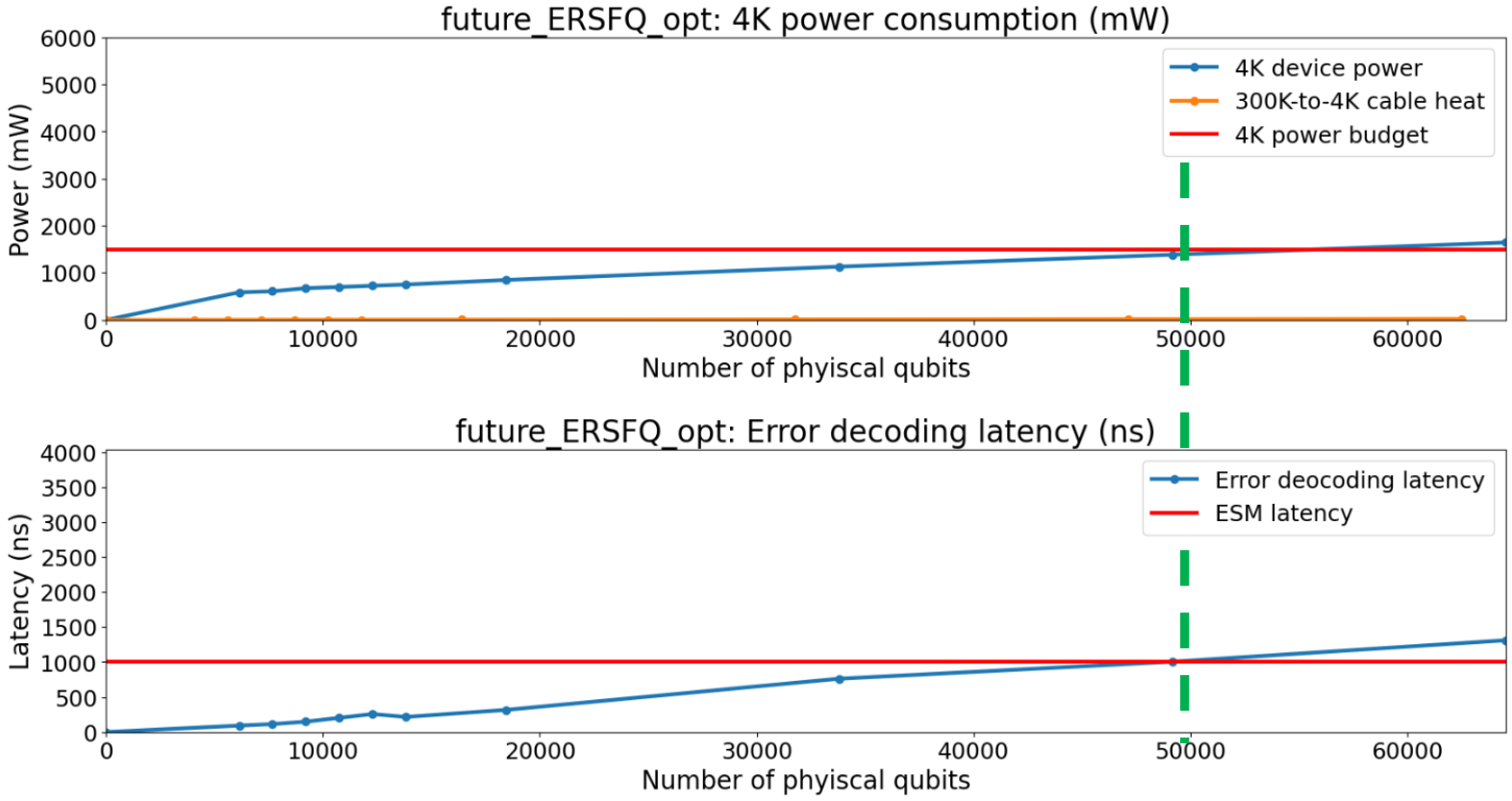
We are polishing our tool with the final check-up and thus the detailed values can be changed. Please find our final slide with the tool release.

XQ-simulator: Scalability analysis

- **Config: future_ERSFQ_opt**
  - Manageable qubit scale: **> ~50,000 qubits**
  - Scalability bottleneck: 4K device power (or Error decoding latency)



future_ERSFQ_opt
(+ with fast & low-power EDU)

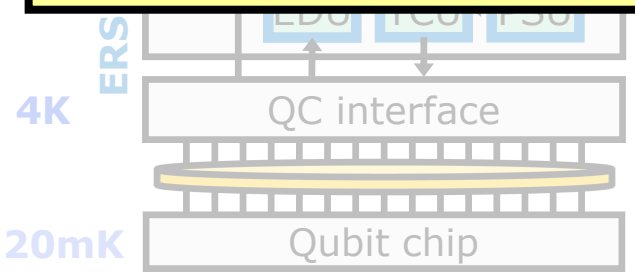


We are polishing our tool with the final check-up, and thus the detailed values can be changed. Please find our final slide with the tool release.

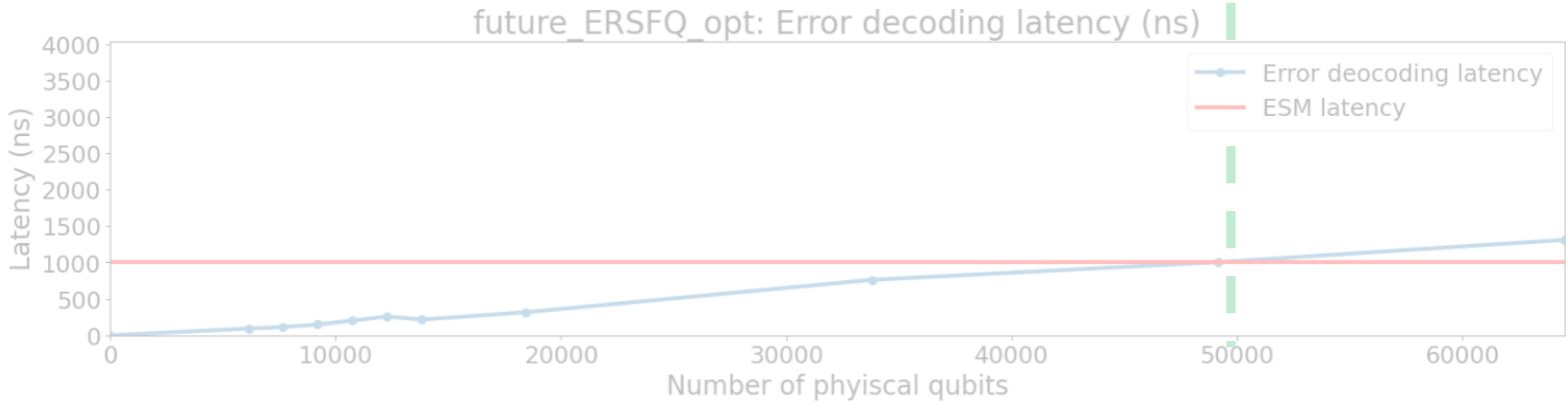
XQ-simulator: Scalability analysis

- **Config: future_ERSFQ_opt**
  - Manageable qubit scale: **> ~50,000 qubits**
  - Scalability bottleneck: 4K device power (or Error decoding latency)

Please explore your own QCP architecture with XQsim!



future_ERSFQ_opt
(+ with fast & low-power EDU)



We are polishing our tool with the final check-up, and thus the detailed values can be changed. Please find our final slide with the tool release.

Index

XQsim tutorial: Summary

- **Scalability analysis tool for the quantum control processor is necessary to realize the large-scale fault-tolerant quantum computer.**
- **XQsim analyze the target quantum control processor's scalability for various microarchitectures, temperatures, and technologies**

Stay tuned for the upcoming XQsim release!

Thank You!

Any questions?

Ilkwon Byun

High Performance Computer System (HPCS) Lab.
Department of Electrical and Computer Engineering
Seoul National University

E-mail: ik.byun@snu.ac.kr