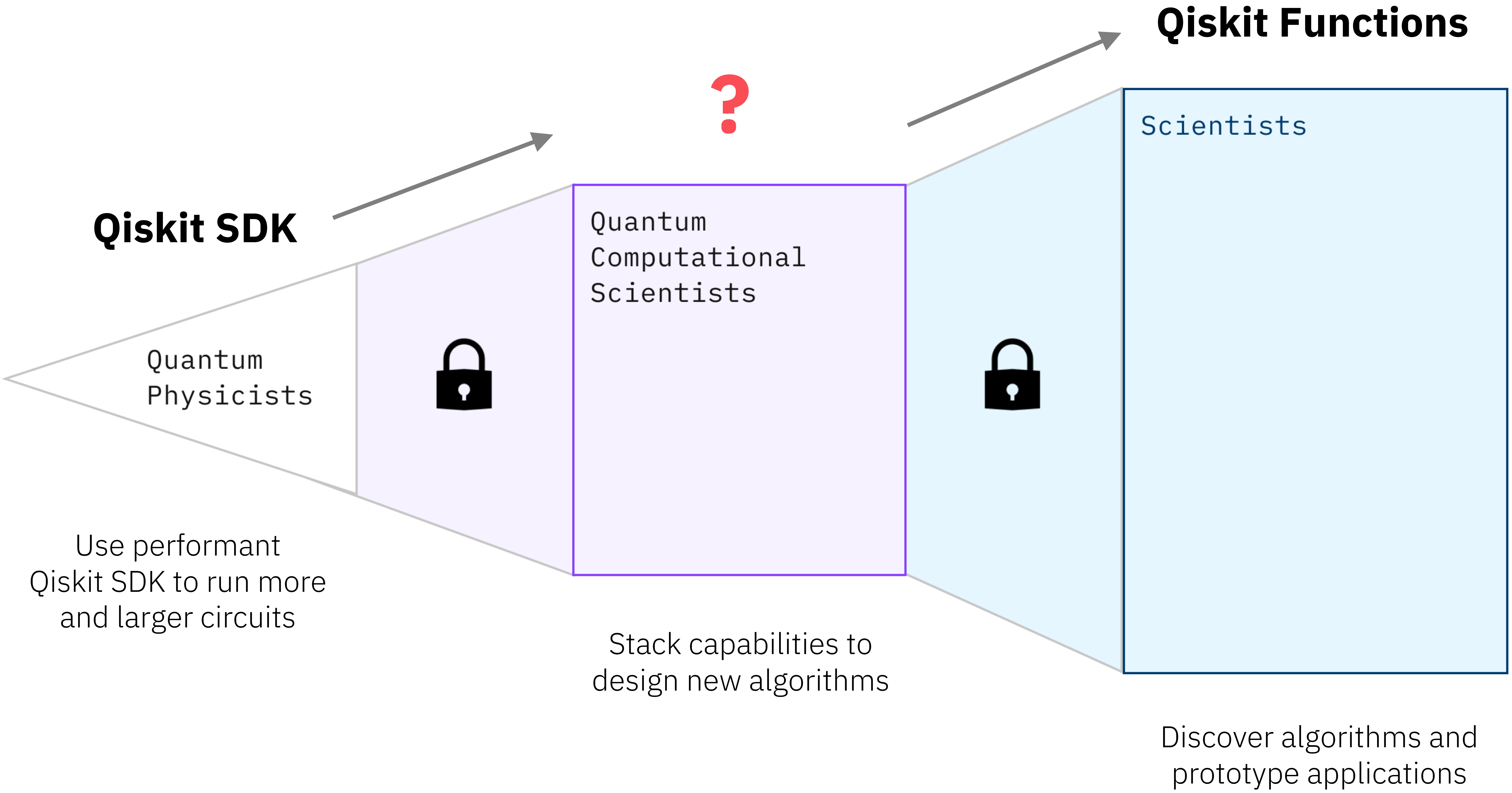
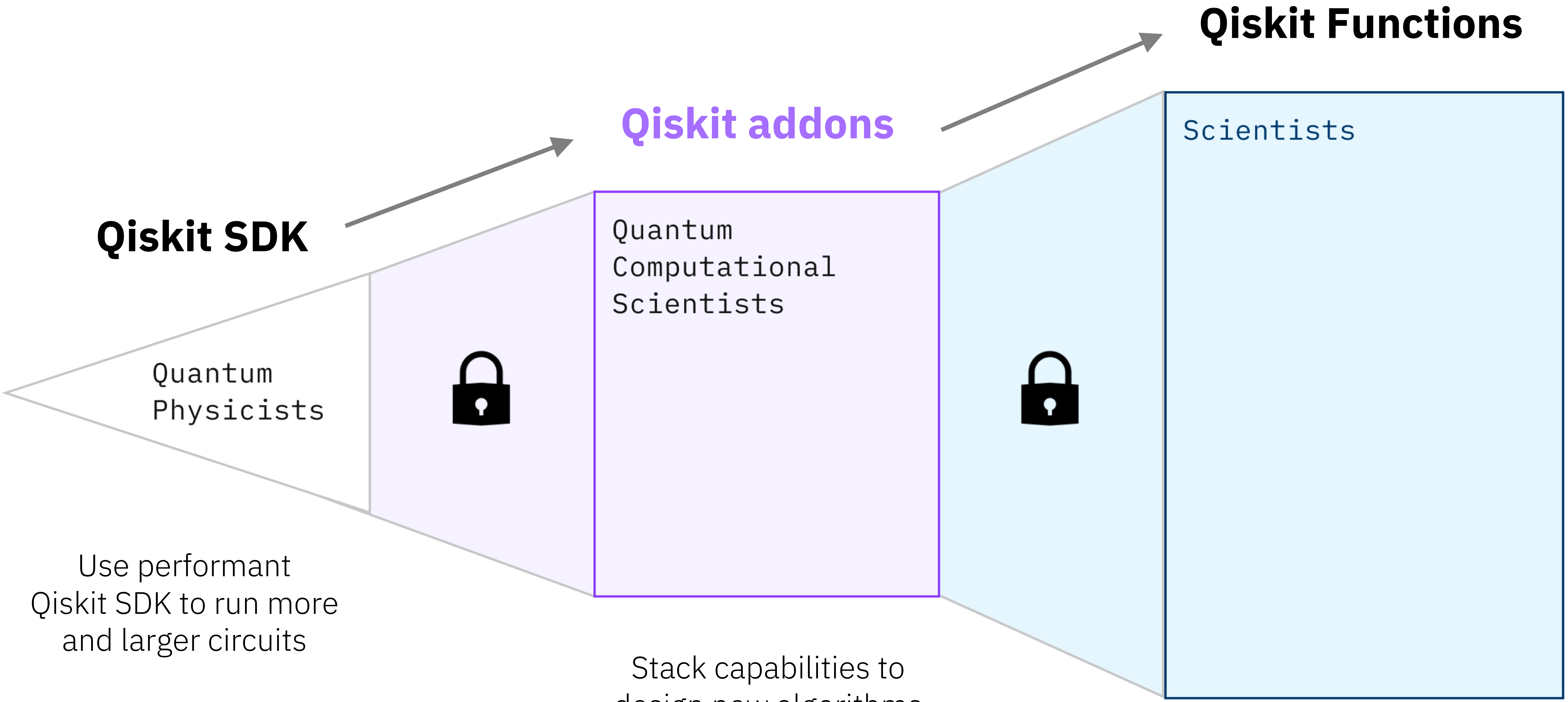
# Tools for Utility Beyond the SDK: Qiskit Addons


Qiskit Functions
Qiskit SDK
Quantum Physicists
Quantum Computational Scientists
Scientists
Use performant Qiskit SDK to run more and larger circuits
Stack capabilities to design new algorithms
Discover algorithms and prototype applications




Use performant Qiskit SDK to run more and larger circuits

Stack capabilities to design new algorithms

Discover algorithms and prototype applications

# Qiskit Functions

Pre-packaged workflows as abstracted services designed to accelerate **algorithm discovery** and **application prototyping** through advanced mapping and performant management techniques

## Application function

↓
Domain-specific inputs

Data / visualizations
↑

Map → Optimize → Execute → Post-process

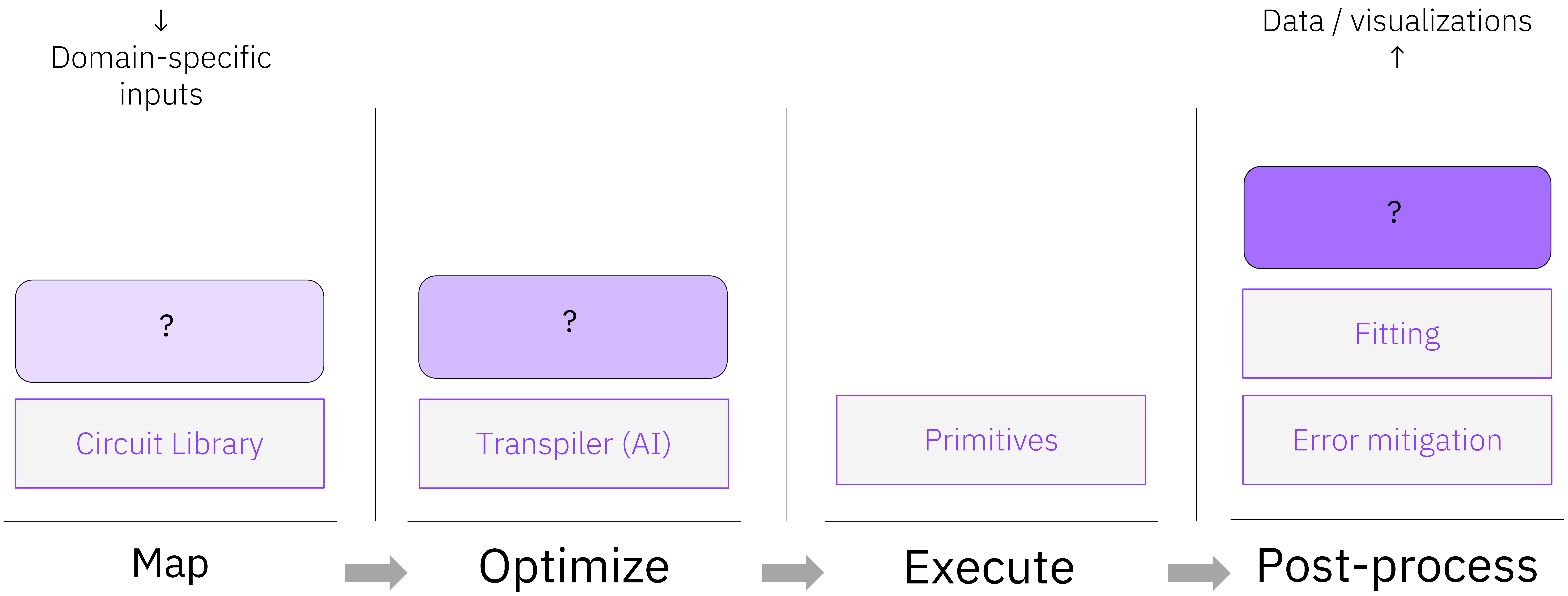
# Qiskit SDK sets the foundation for quantum workflows

↓
Domain-specific inputs

Data / visualizations
↑

| Map | Optimize | Execute | Post-process |
|---|---|---|---|
| Circuit Library | Transpiler (AI) | Primitives | Fitting<br>Error mitigation |

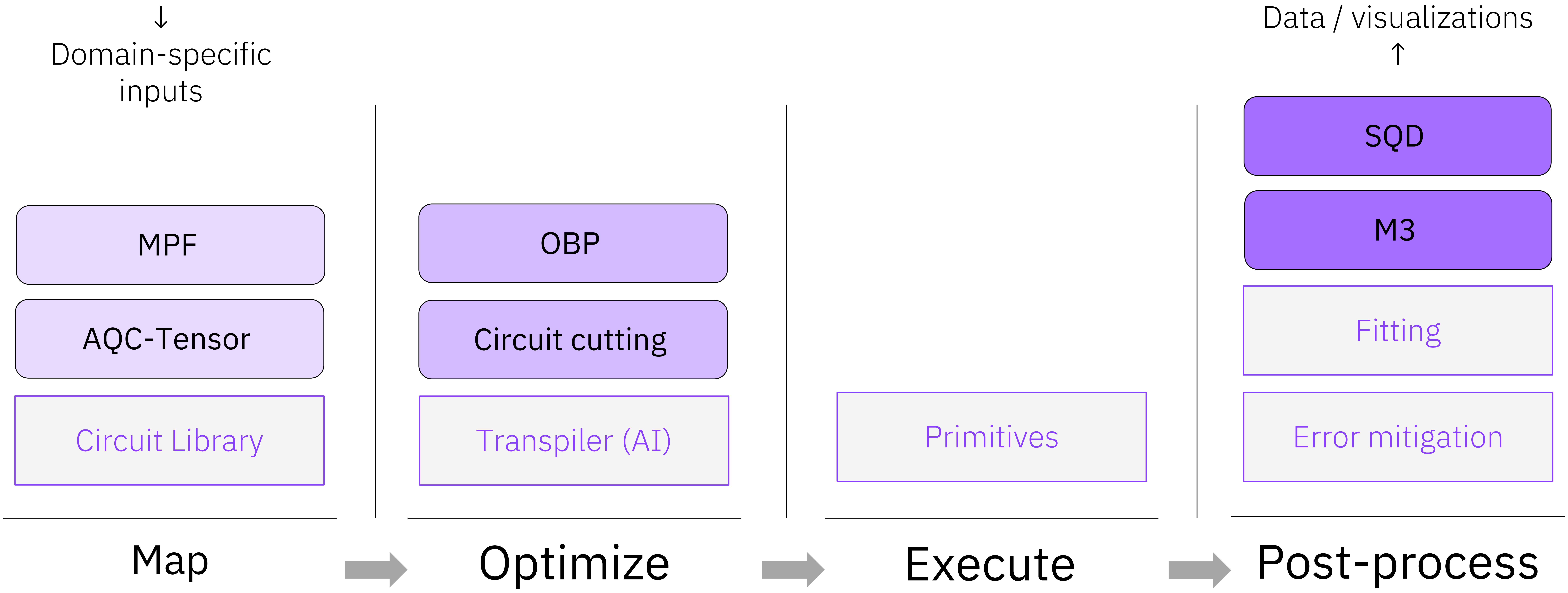
# Qiskit SDK sets the foundation for quantum workflows



# Qiskit addons build on the Qiskit SDK

A collection of research capabilities developed as modular tools that can plug into a workflow to scale or design new algorithms at the utility scale.

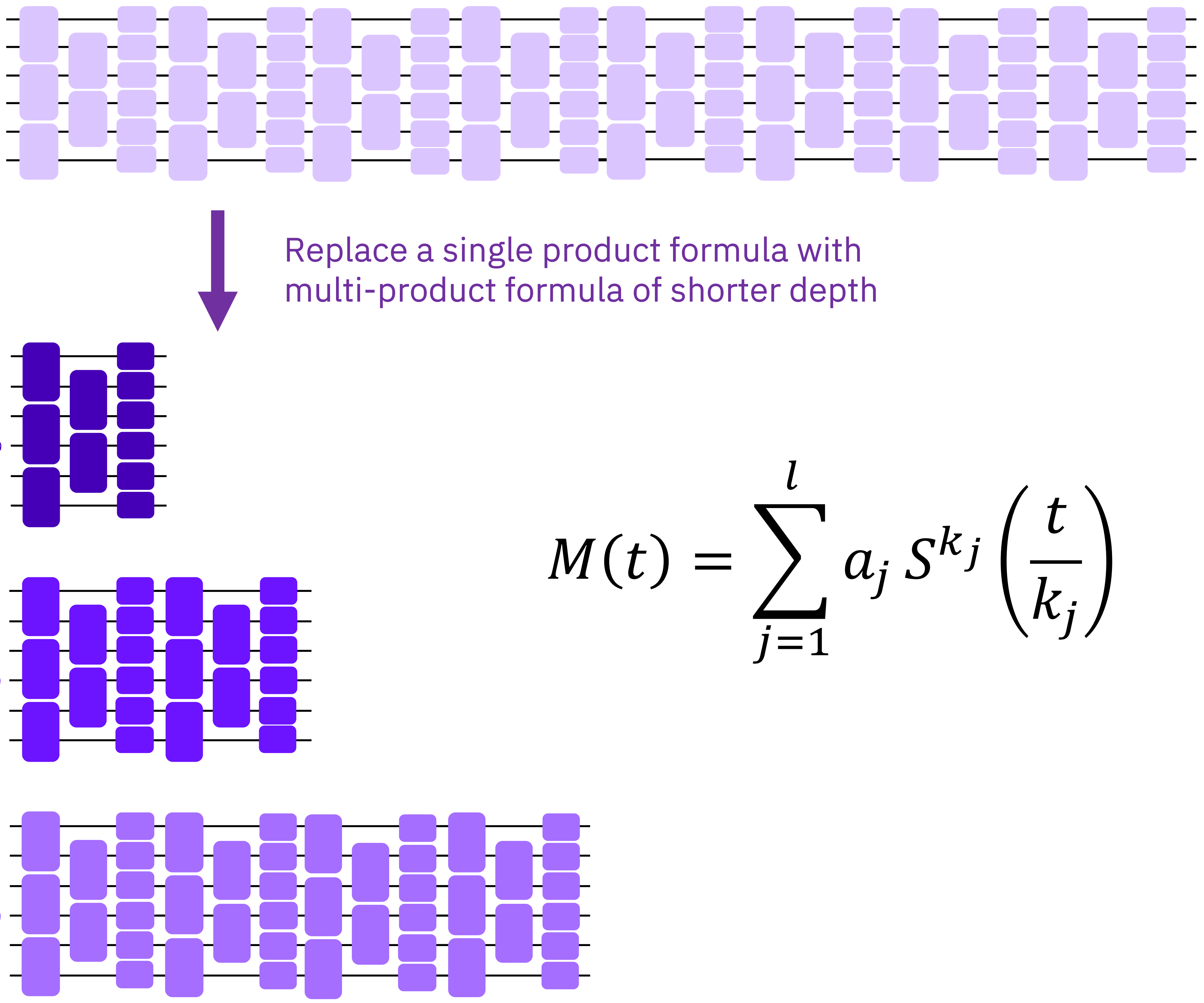
Starting with **multi-product formulas** (MPF), **approximate quantum compilation** (AQC-Tensor), **operator backpropagation** (OBP), and **sample-based quantum diagonalization** (SQD)



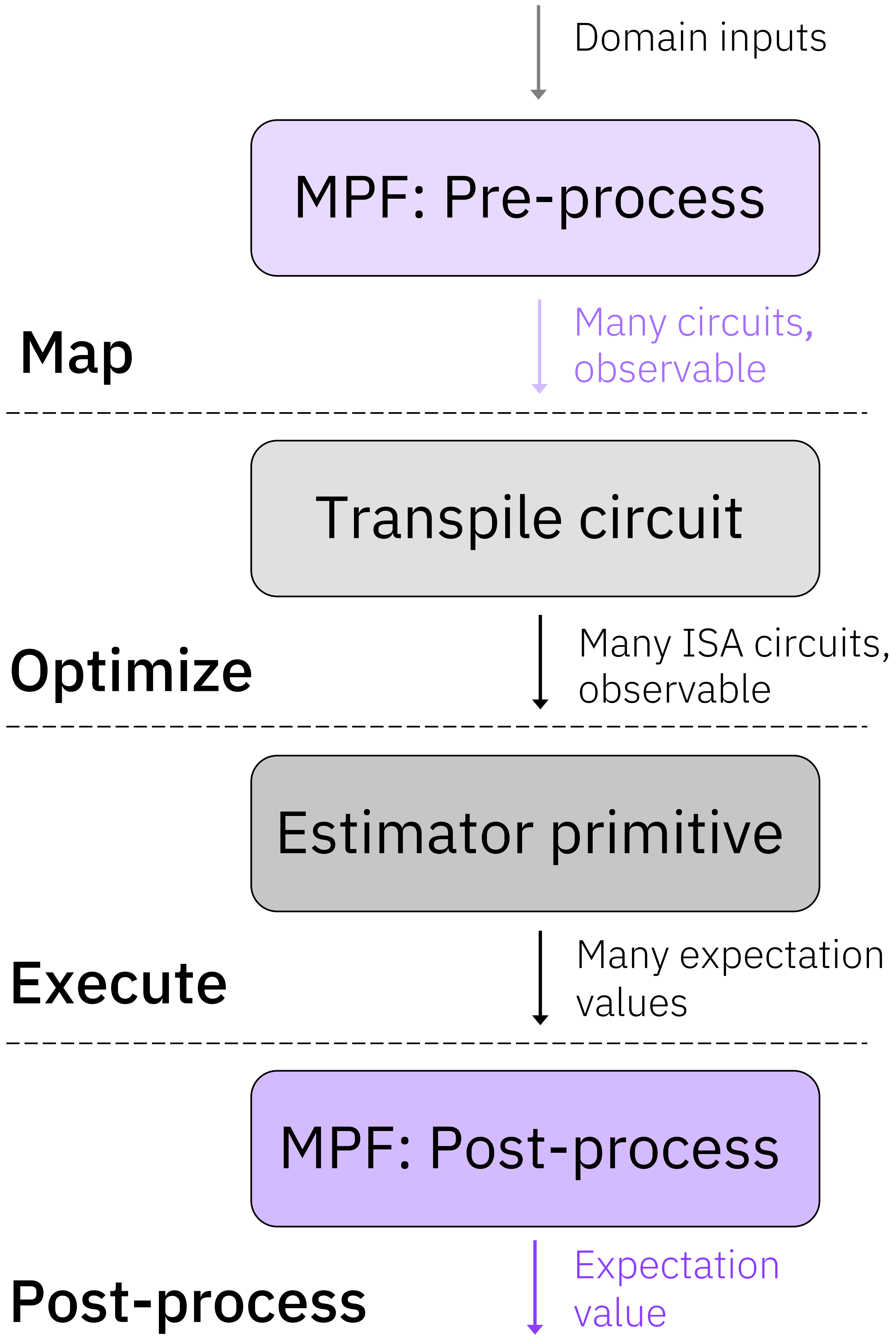
# Multi-product formulas

A technique to reduce algorithmic (Trotter) errors through a weighted combination of several circuit executions

- Reducing Trotter error can require deep circuits
- MPF combines experiments with different Trotter errors to produce an estimate with lower overall Trotter error
- Can leverage TN methods to further reduce Trotter error
- Demonstrated on
  - 50 qubits
  - 27 2q-depth




$$M(t) = \sum_{j=1}^{l} a_j \, S^{k_j}\left(\frac{t}{k_j}\right)$$




**Research**

Quantum 7, 1067 (2023)
arxiv.org/abs/2407.17405
arxiv.org/abs/2306.12569

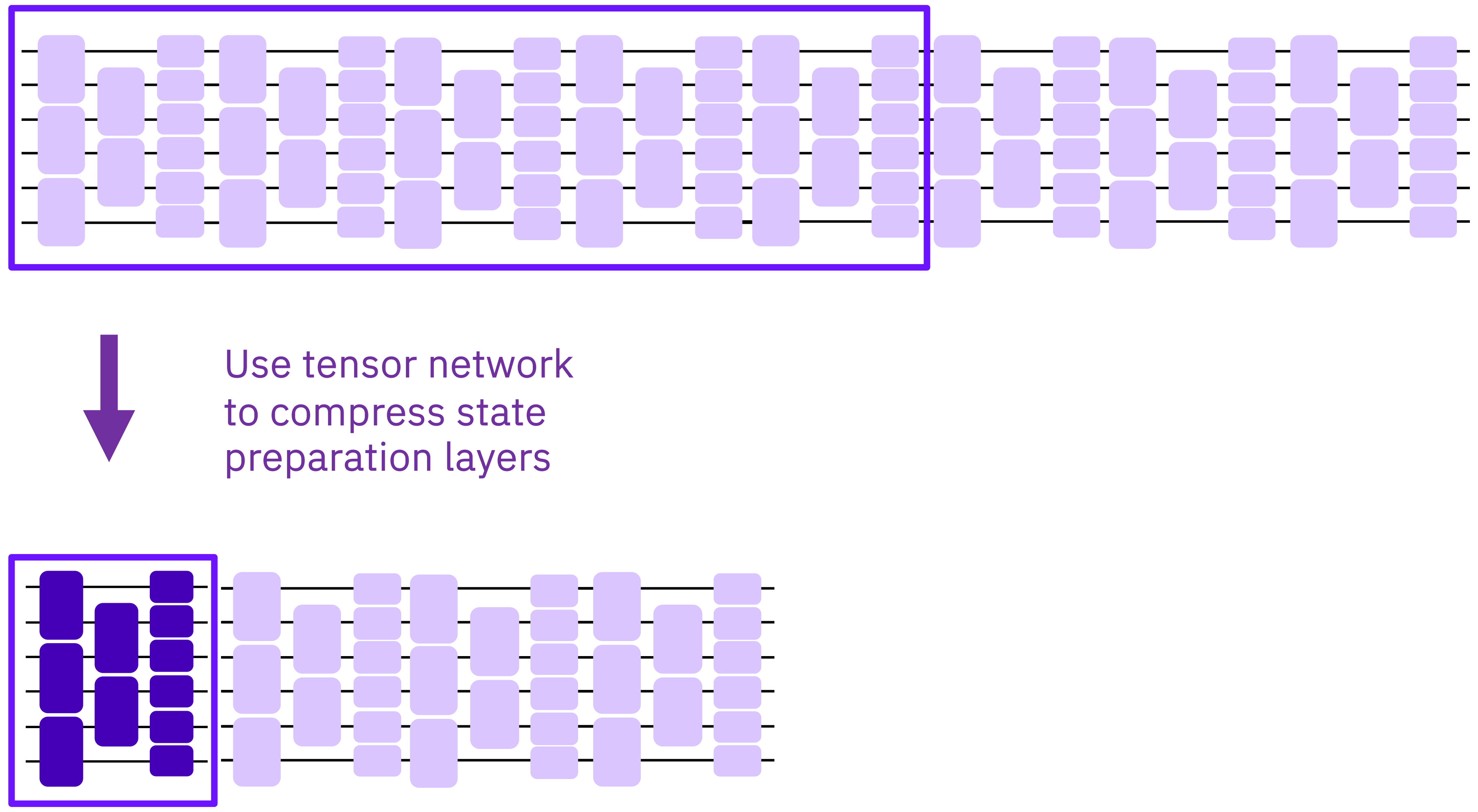
**Development**

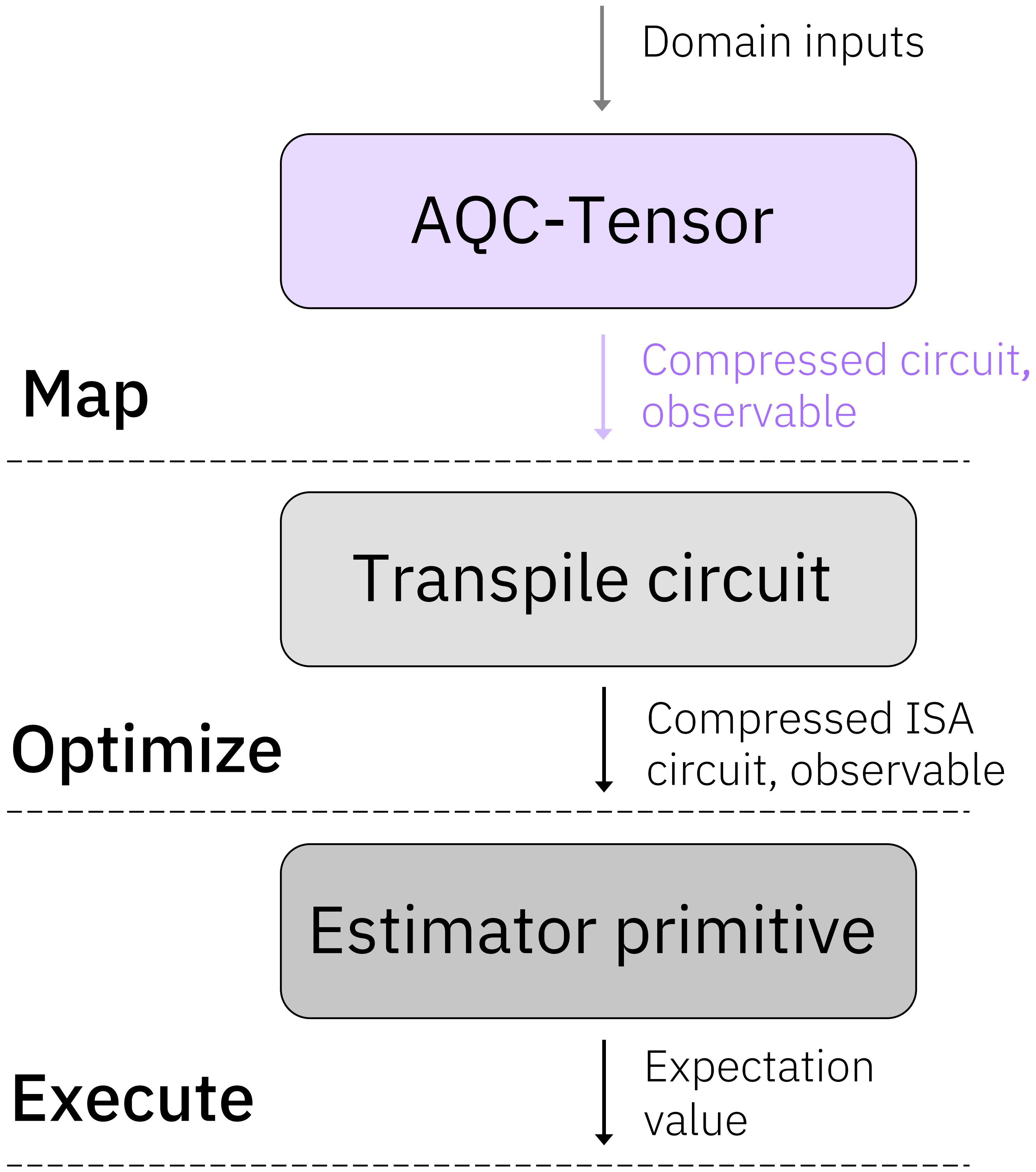
github.com/Qiskit/qiskit-addon-mpf

# Approximate quantum compilation with tensor networks

A technique to produce shorter-depth Trotter circuits for time evolution problems by classically compressing the initial layers

- Trotterized time evolution is often classically tractable for small times, but evolving further becomes classically hard
- AQC uses a tensor network to compress initial circuit layers, allowing more circuit depth to be spent on further time evolution
- Demonstrated on
  - 50 qubits
  - 27 2q-depth




**Research**

arxiv.org/abs/2301.08609
arxiv.org/abs/2407.17405

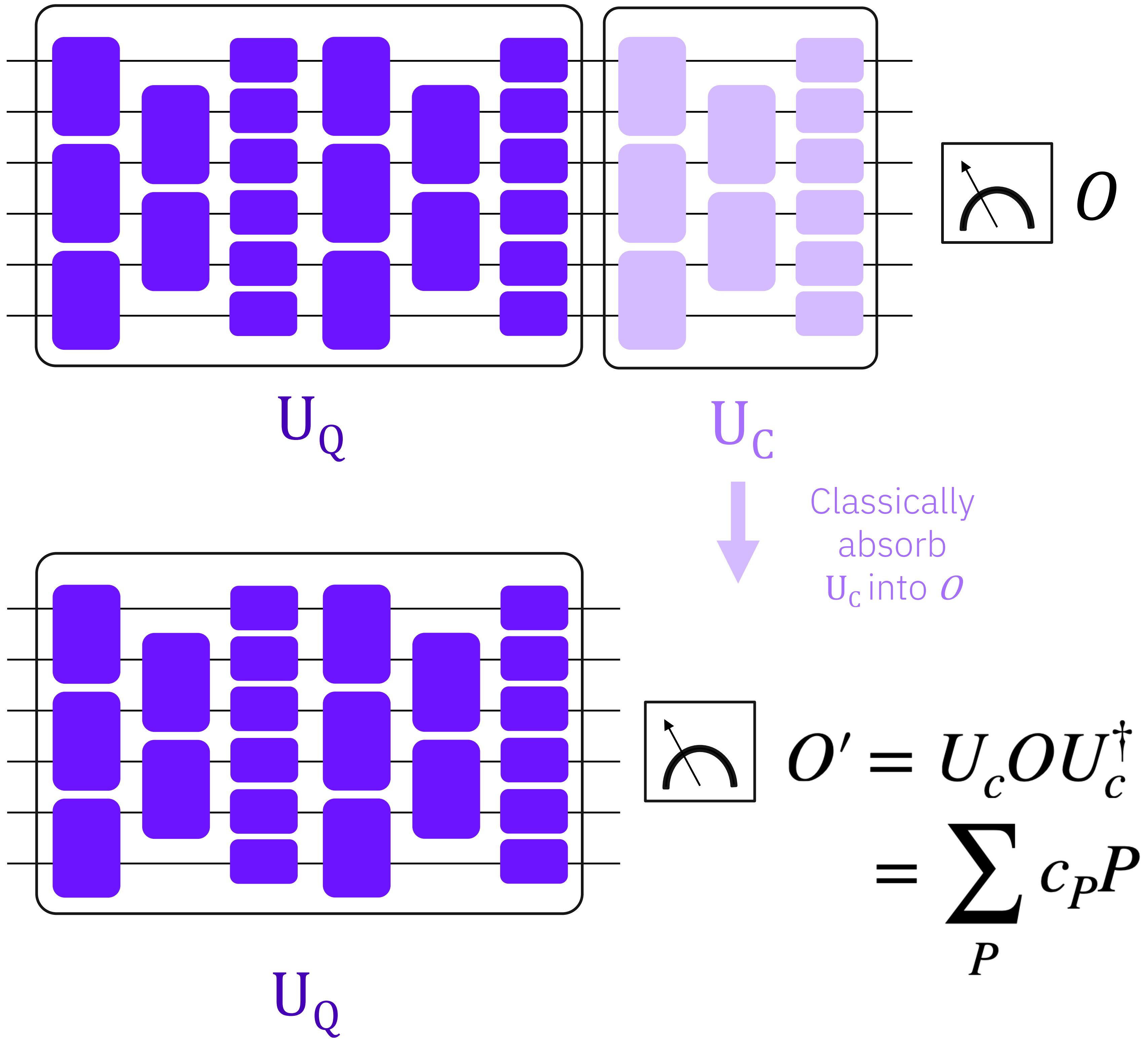
**Development**

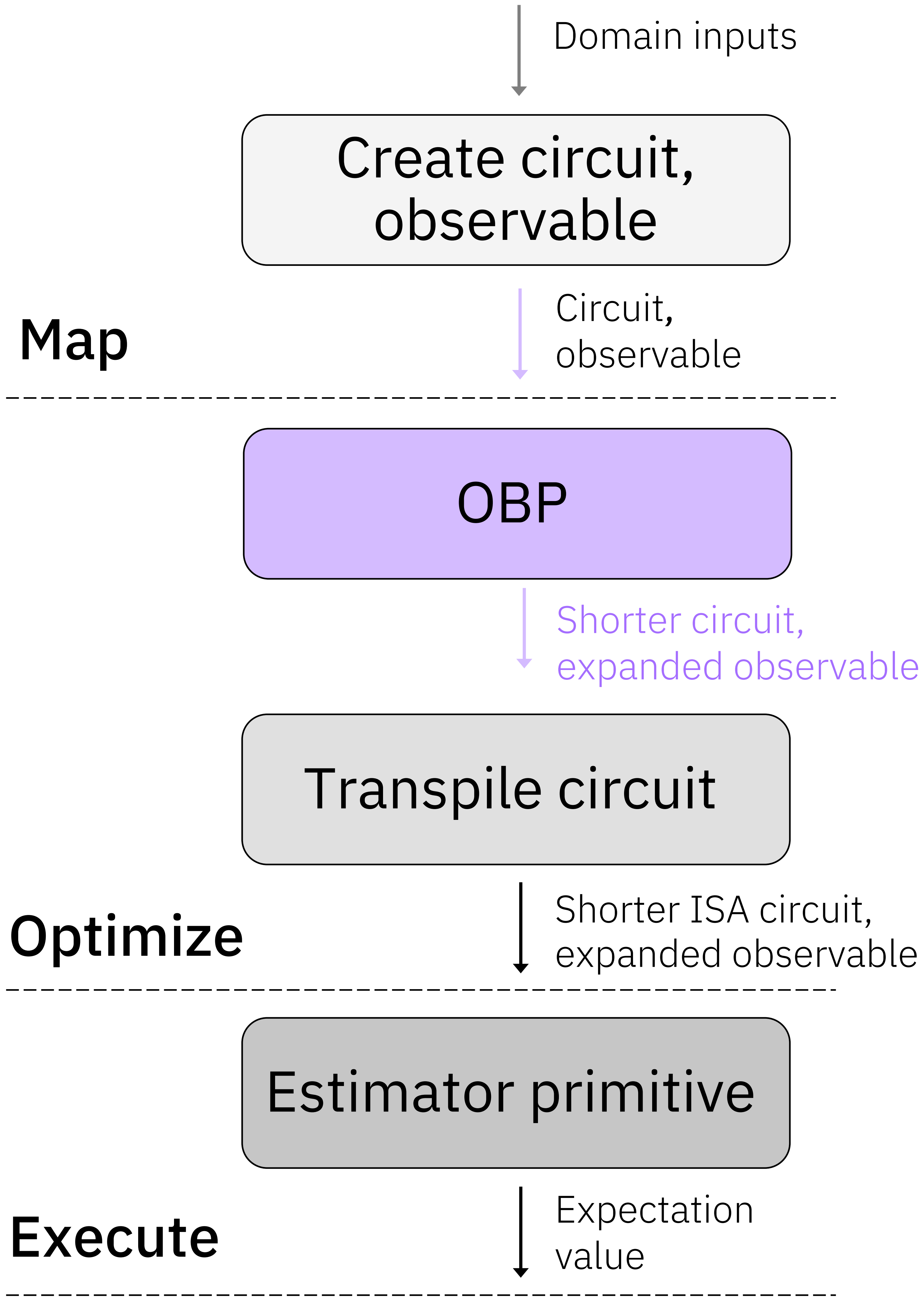
github.com/Qiskit/qiskit-addon-aqc-tensor

# Operator backpropagation

A technique to reduce circuit depth by trimming operations from the end at the cost of more operator measurements.

- Some circuits, such as Trotterized time-evolution, get deeper and nearer to Clifford as they become more accurate
- OBP can reduce maximum depth of these circuits, reducing impact of noise
- Demonstrated on
  - 127 qubits
  - 82 2q-depths




**Research**

In preparation

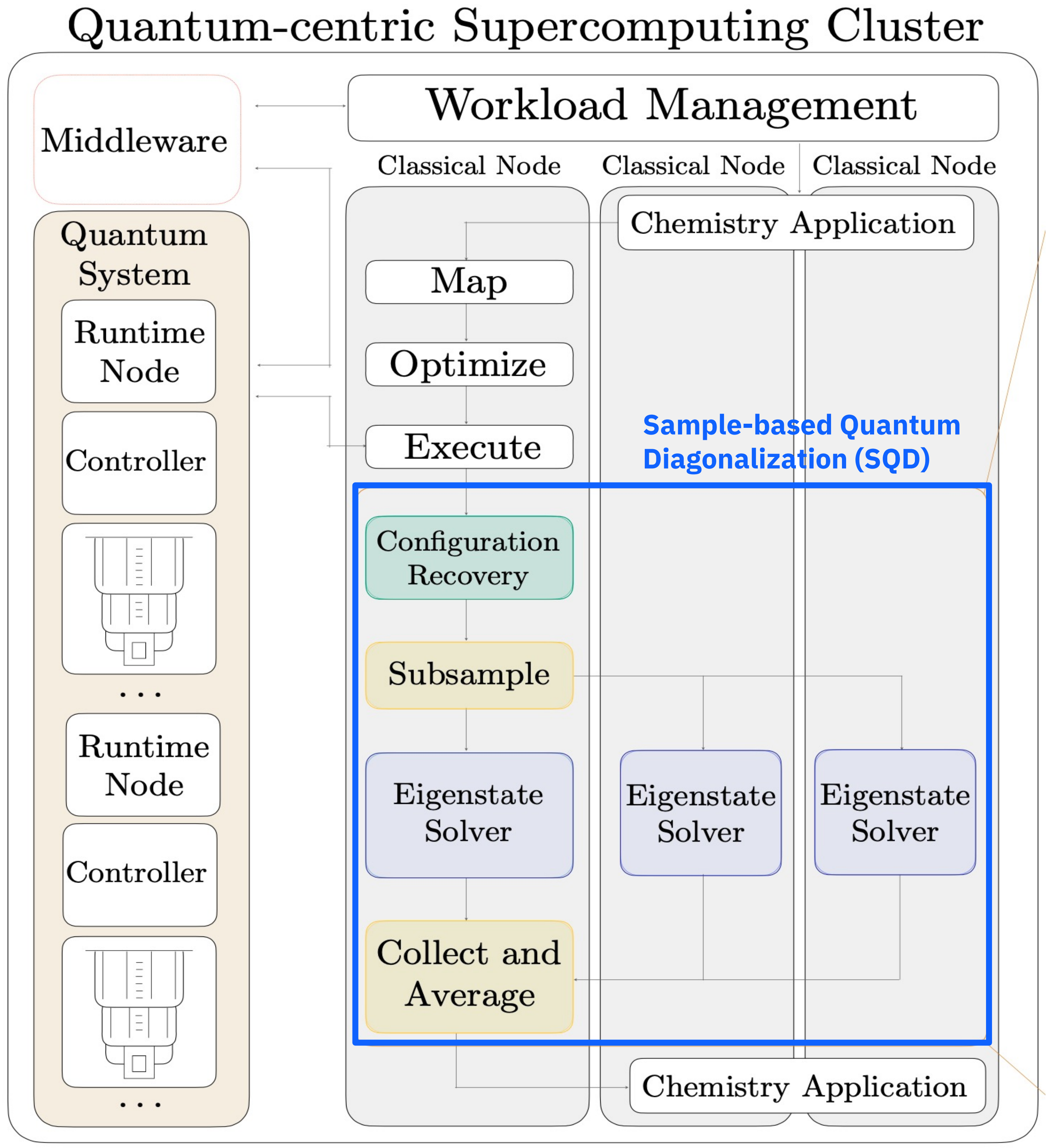
**Development**

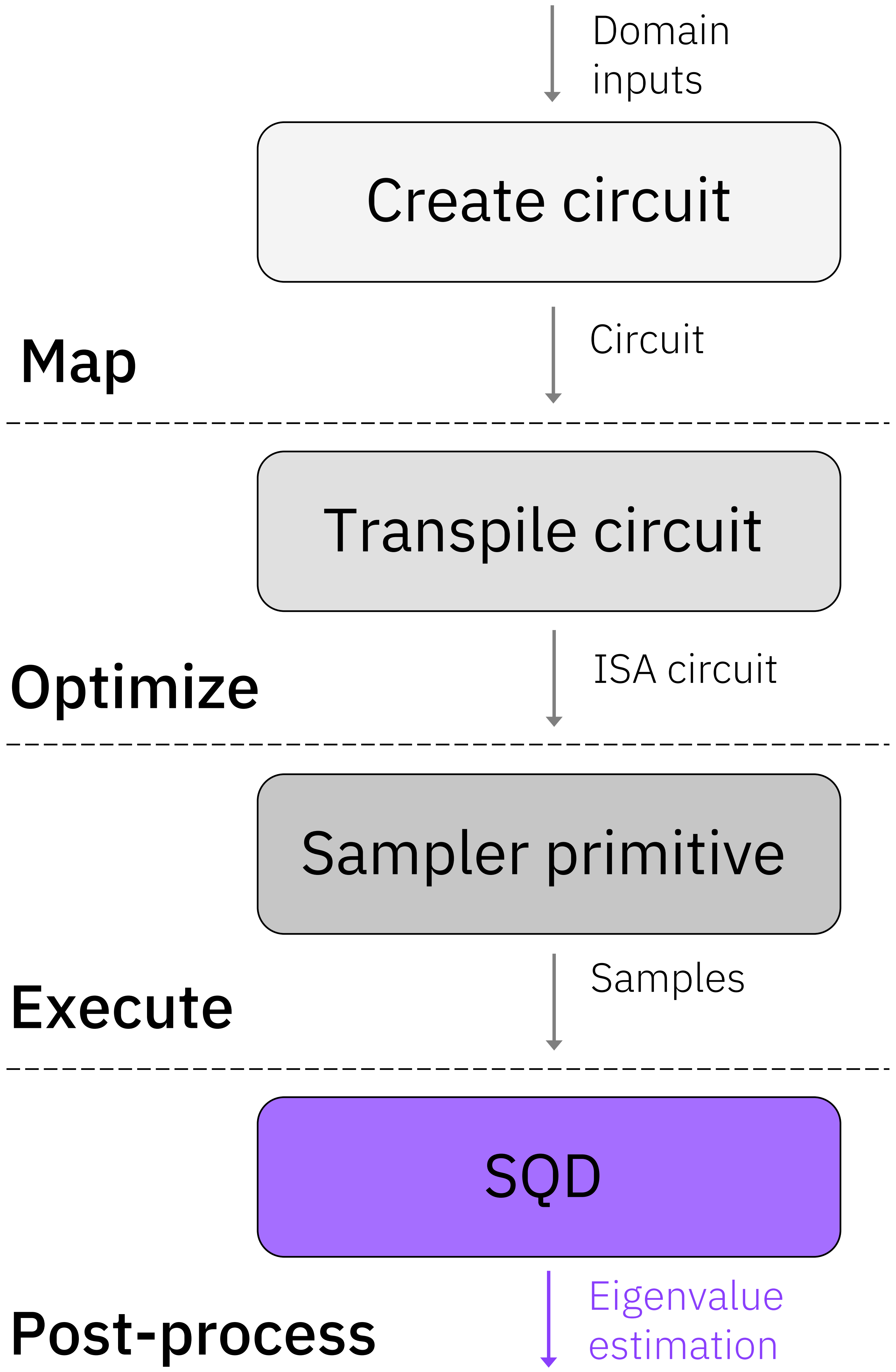
github.com/Qiskit/qiskit-addon-obp

# Sample-based quantum diagonalization

A technique using classical distributed computing to produce more accurate eigenvalue estimations from noisy quantum samples

- Energy estimation on current hardware is limited by circuit depth, cost of error mitigation, overhead of evaluating expectation values
- SQD refines noisy samples with classical distributed computing to address large Hamiltonians
- Demonstrated on
  77-qubit chemistry Hamiltonian
  3590 CNOTs




**Research** → **Development**

arxiv.org/abs/2405.05068
arxiv.org/abs/2410.09209

github.com/Qiskit/qiskit-addon-sqd

# Qiskit addons

- Advanced research capabilities made available as easy-to-use modular software tools

- Designed to plug into workflows to build new algorithms at utility scale

# Qiskit addons workshop

- Pick one or more of the following addons:
  - Multi-product formulas (MPF)
  - Approximate quantum compilation with tensor networks (AQC-Tensor)
  - Operator backpropagation (OBP)
  - Sample-based quantum diagonalization (SQD)
- Download and go through the tutorial
- Note any questions, issues, or feedback that you have
- Reconvene and discuss

https://docs.quantum.ibm.com/guides/addons

IBM

# Code example for the OBP addon

```python
# STEP 1: Map
# define Hamiltonian time evolution problem

circuit = get_circuits_for_experiment( ... )
observable =  ...

# STEP 2: Optimize
# transpile the final circuit
transpiled_circuit = transpile( ... )

# STEP 3: Execute
job = estimator.run(transpiled_circuit, observable)
result = job.result()

# STEP 4: Postprocess
final_expval = result[0].data.evs
```

```python
# STEP 1: Map
# define Hamiltonian time evolution problem

circuit = get_circuits_for_experiment( ... )
observable =  ...

# STEP 2: Optimize
# slice circuit into layers for backpropagation
from qiskit_addon_utils.slicing import slice_by_gate_types
slices = slice_by_gate_types(circuit)

# specify an operator budget for backpropagation
from qiskit_addon_obp.utils.simplify import OperatorBudget
op_budget = OperatorBudget(max_qwc_groups=10)

# OBP: Backpropagate slices onto the observable
from qiskit_addon_obp import backpropagate
bp_obs, remaining_slices, metadata = backpropagate(observable, slices, op_budget)

from qiskit_addon_utils.slicing import combine_slices
bp_circuit = combine_slices(remaining_slices)

# transpile the final circuit
transpiled_circuit = transpile( ... )

# STEP 3: Execute
job = estimator.run(transpiled_circuit, bp_obs)
result = job.result()

# STEP 4: Postprocess
final_expval = result[0].data.evs
```