Weaver

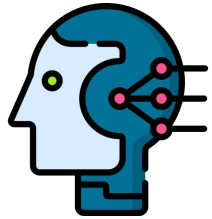
A Retargetable Compiler Framework for FPQA Quantum Architectures

Oğuzcan Kırmemiş*, Francisco Romão*
Emmanouil Giortamis, and Pramod Bhatotia

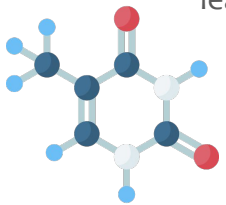
The Quantum Computing Era

- **Speedup:** Quantum computing promises exponential speedup for certain applications
- **Versatility:** Quantum computing capabilities are applicable to many areas

Cryptography



Machine learning



Chemistry

Heterogeneity of quantum hardware architectures

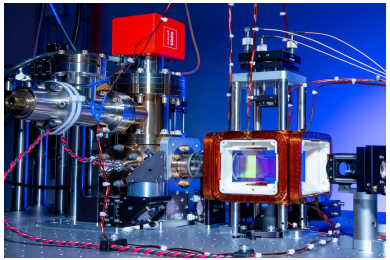
Multiple quantum technologies under development

Heterogeneous Quantum Architectures

Superconducting

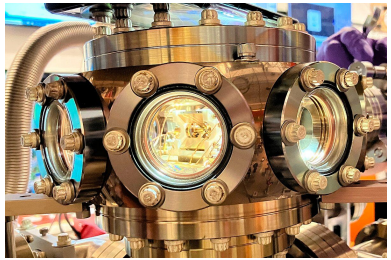
- State of the art
- Fast gates
- Fixed layout

FPQAs (Neutral Atoms)



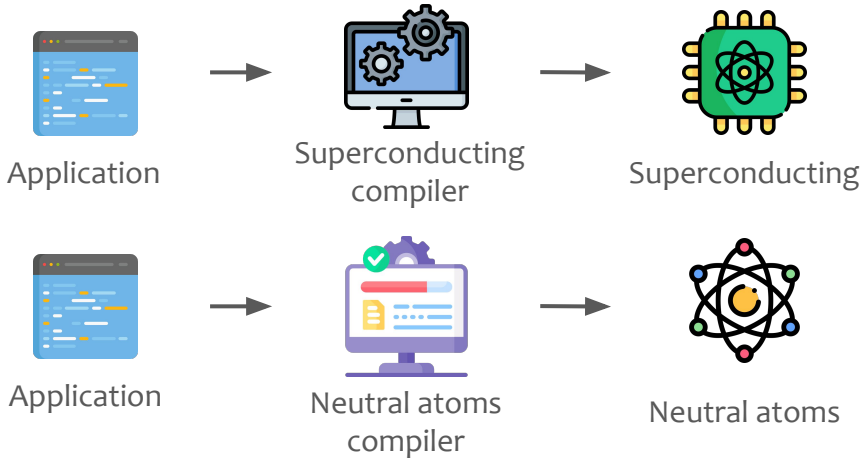
- Reconfigurable layout
- Strong stability
- Slow execution

Trapped Ions



- High fidelity
- Less scalable
- Slow execution

State-of-the-art: Architecture-specific Compilers



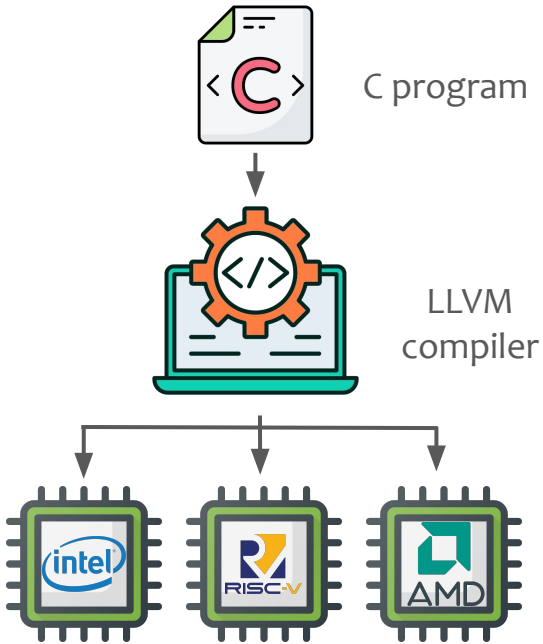
Advantage:
Efficiently leverage *unique* quantum hardware capabilities

Disadvantage:
Do not support *retargetability* across different architectures

Retargetable Compilers for Classical Computing

Compilers for classical computing provide:

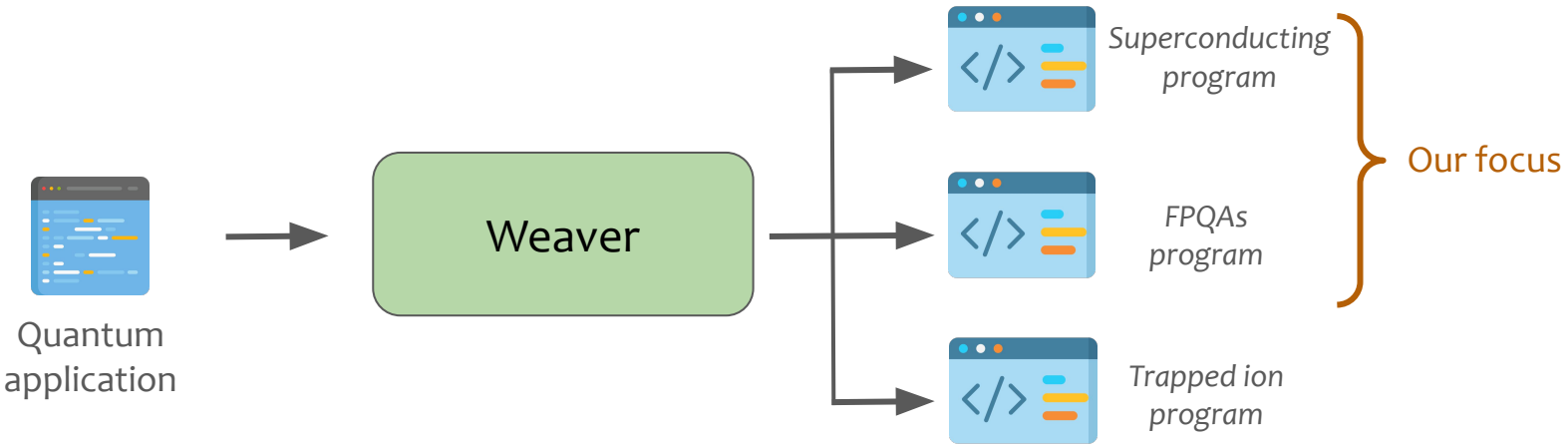
- **Portability:**
  - Retargetability for different architectures
- **Performance:**
  - Architecture-specific optimizations



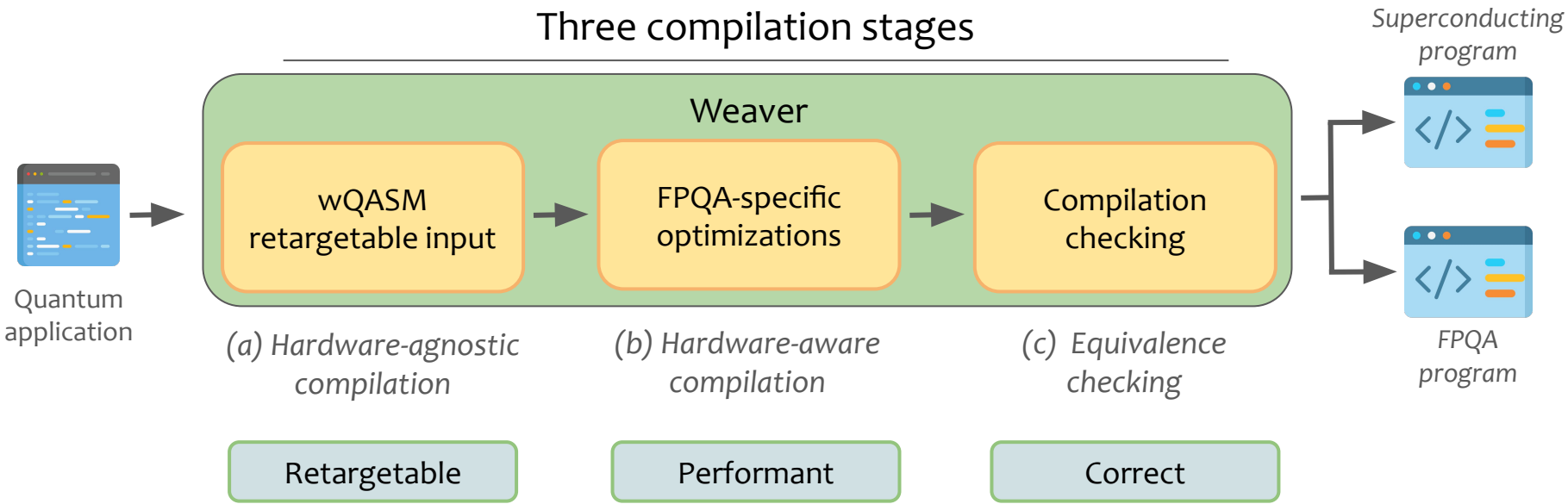
Problem Statement

How can we **retarget** applications to different quantum technologies while leveraging the **unique** hardware capabilities in a **performant** manner?

Weaver: A Retargetable Compiler Framework



Design Goals

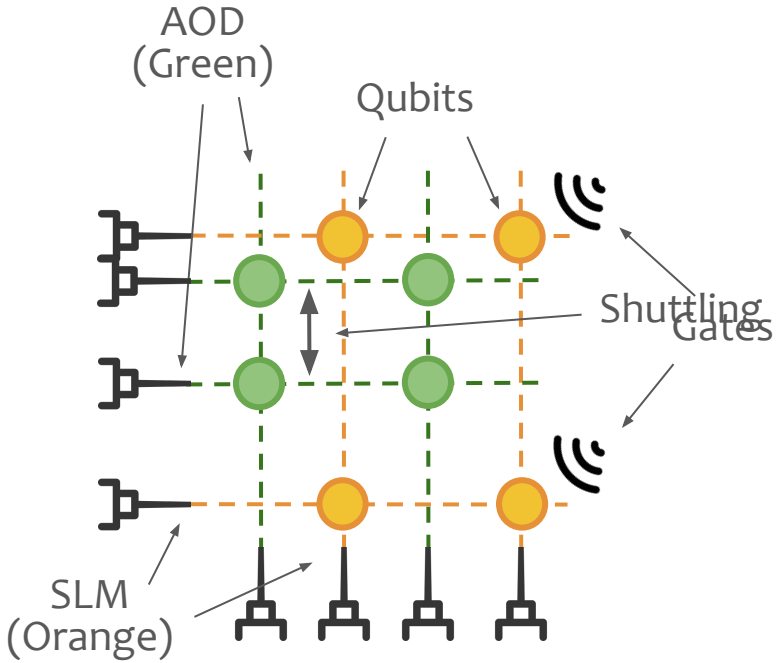


Outline

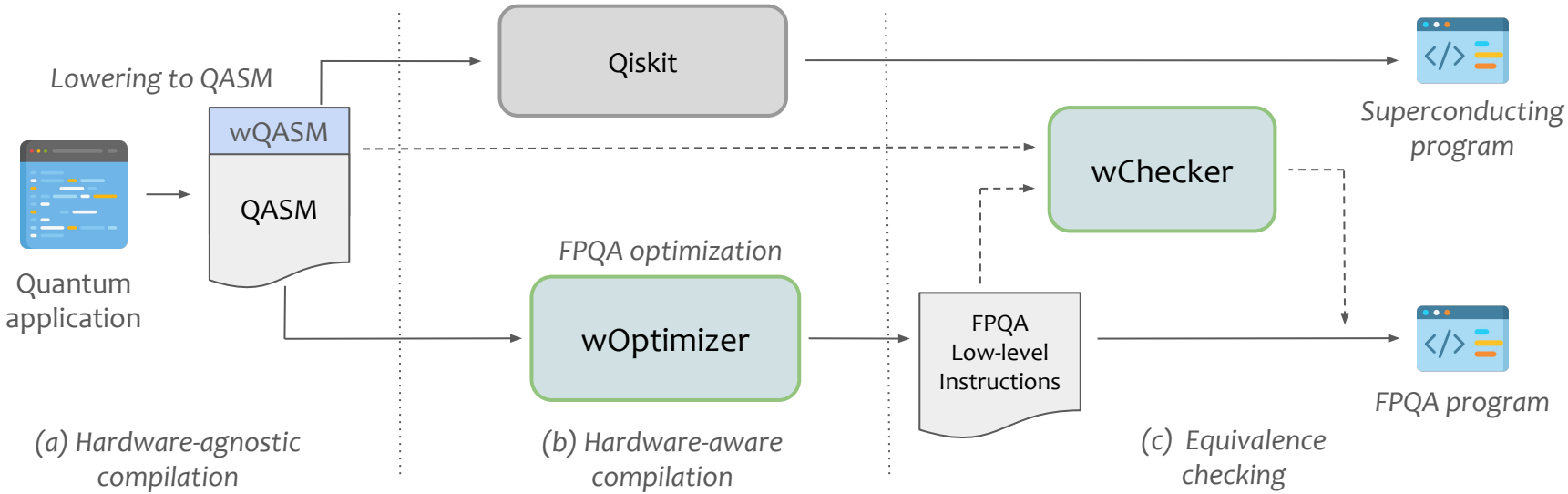
- ~~Motivation~~
- Design
- Evaluation

Neutral Atoms / FPQA Basics

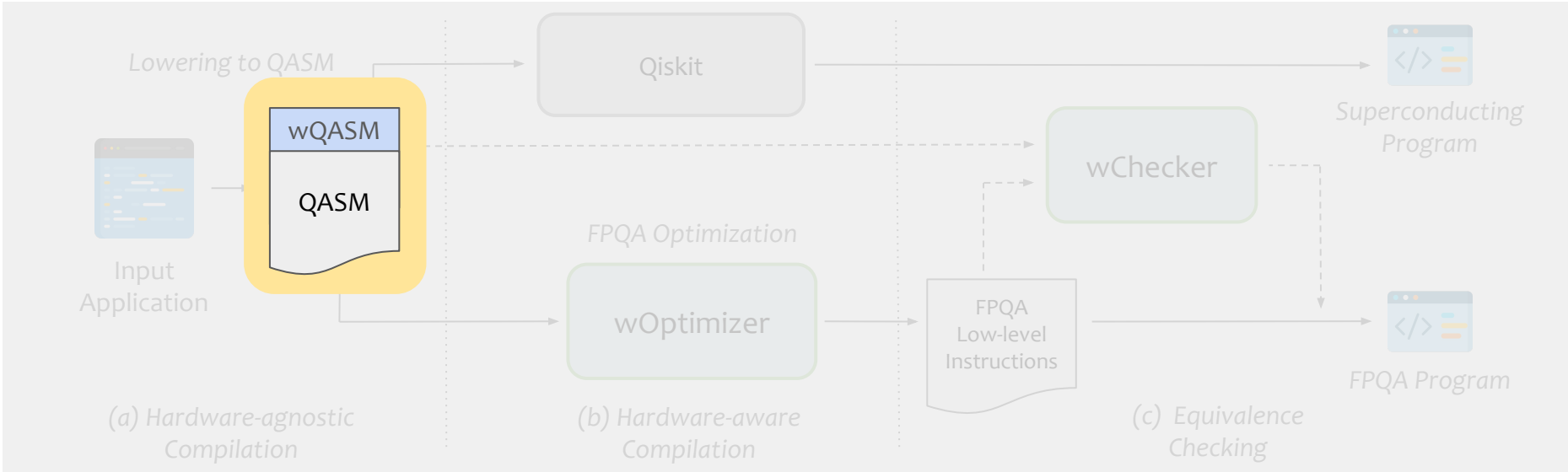
- **Qubits:** Metal atoms
- **Gates:** Microwave pulses
- **Hardware grid:** Laser grid
  - **Static Grid (SLM)**
  - **Dynamic Grid (AOD)**
- **Shuttling:** Laser row and/or column movement



System Overview



System Overview



Challenge #1: Retargetability

Challenge	Our approach
• Retarget quantum application with minimal overhead	→ Hardware-agnostic compilation stage
• Ability to leverage hardware's unique capabilities	→ QASM architecture-specific extension

Our contribution:
wQASM: an FPQA-specific QASM extension

OpenQASM

- OpenQASM is a low-level, hardware-agnostic, quantum description language
- By being hardware-agnostic it cannot leverage unique hardware capabilities

```
OPENQASM 3.0;
include "qelib1.inc";
qreg q[2];
creg c[2];
h q[0];
cx q[0], q[1]; // Apply CNOT gate
measure q -> c; // Measure both qubits
```

Defines two quantum and two classical bits (registers)

Applies Hadamard gate on qubit 0

Applies CNOT gate

Measures qubits and stores on classical bits

wQASM: An OpenQASM Extension for FPQAs

- wQASM extends QASM with FPQA-specific instructions on top of the *annotation* support
- can be interpreted as an FPQA instruction list
- allows leveraging architecture-specific capabilities

```
input q[4];
@rydberg
gate gate_1 a0, a1, a2, a3
{
    cz a0, a1;
    ccz a1, a2, a3;
    ccz a0, a2, a3;
}
gate_1 q[0], q[1], q[2], q[3]
```

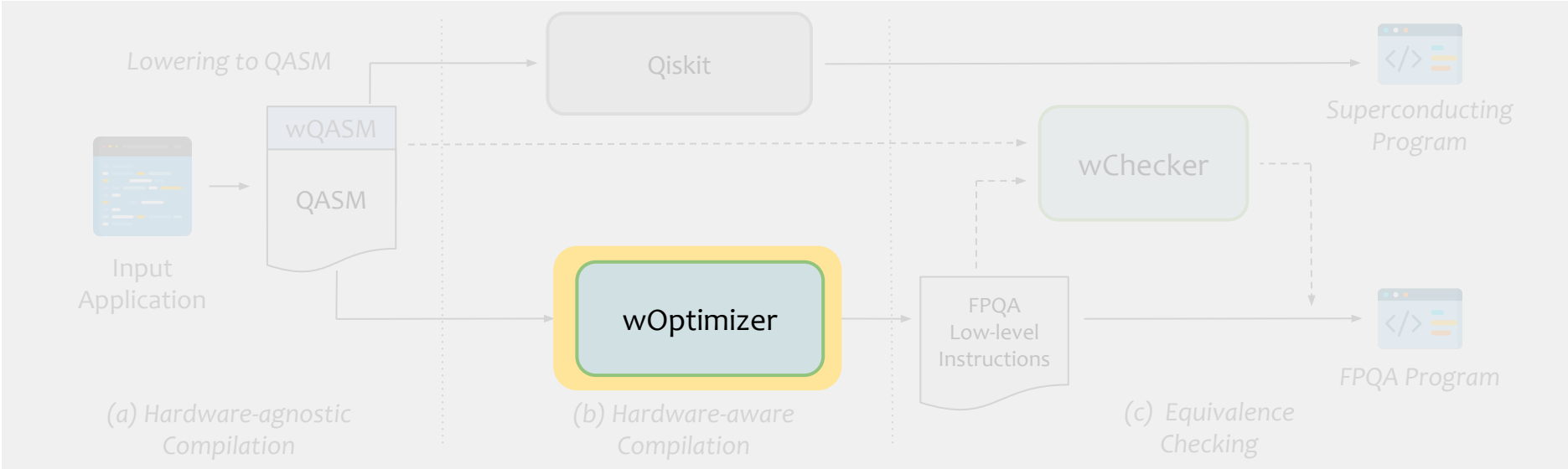
FPQA-specific instruction

Defines *gate_1* with 4 qubit inputs

Superconducting-compatible instructions

Applies *gate_1* on qubits *0*,*1*,*2* and *3*

System Overview



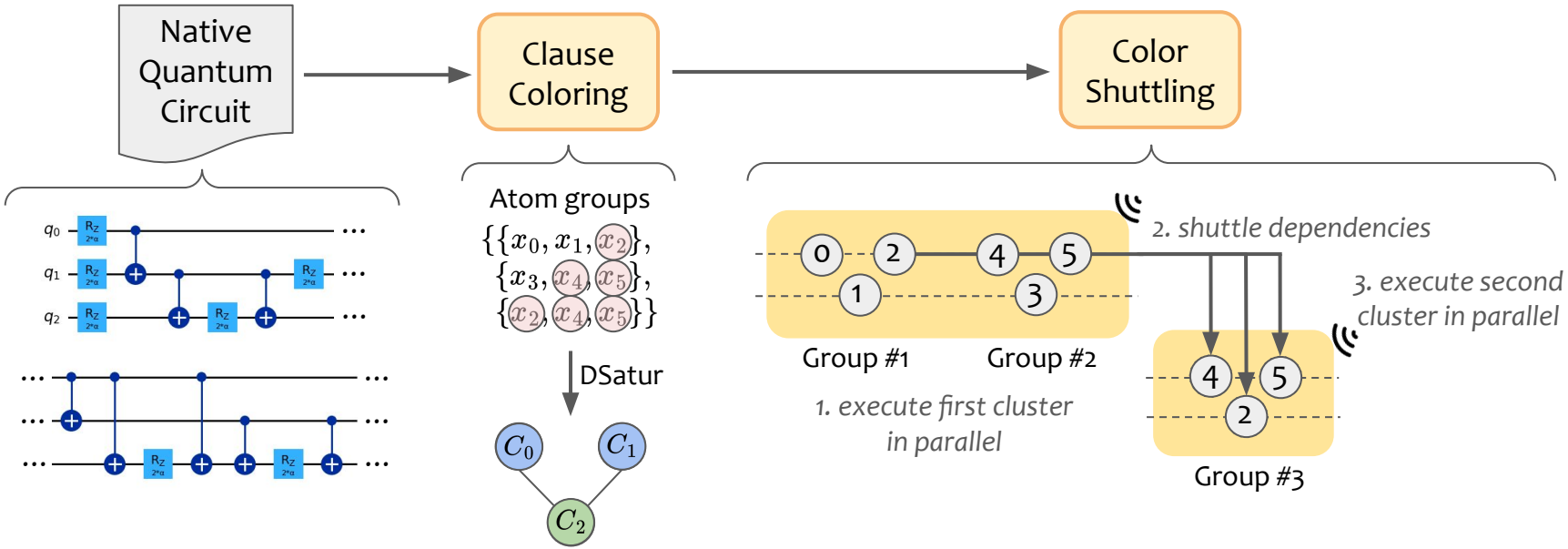
Challenge #2: Performance

Challenge	Our approach
• FPQA gate execution is slow	→ High parallelization of gate execution
• Movable rows or columns cannot overlap	→ A smart shuttling algorithm avoids overlap

Our contribution:
wOptimizer leverages FPQA-specific capabilities for code optimization

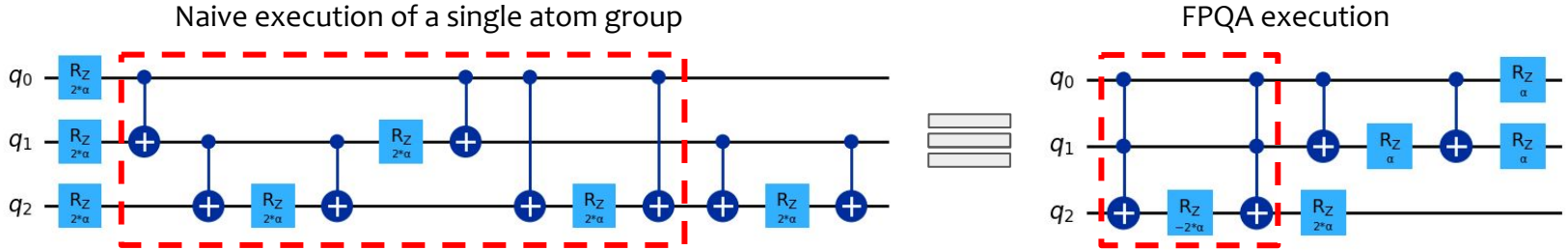
wOptimizer: Leveraging FPQAs - Showcase

- Are there actual quantum programs where hardware-specific optimizations are **beneficial?**
  - Max-3SAT: $(x_0 \vee x_1 \vee x_2) \wedge (x_3 \vee x_4 \vee x_5) \wedge (\neg x_2 \vee \neg x_4 \vee \neg x_5)$

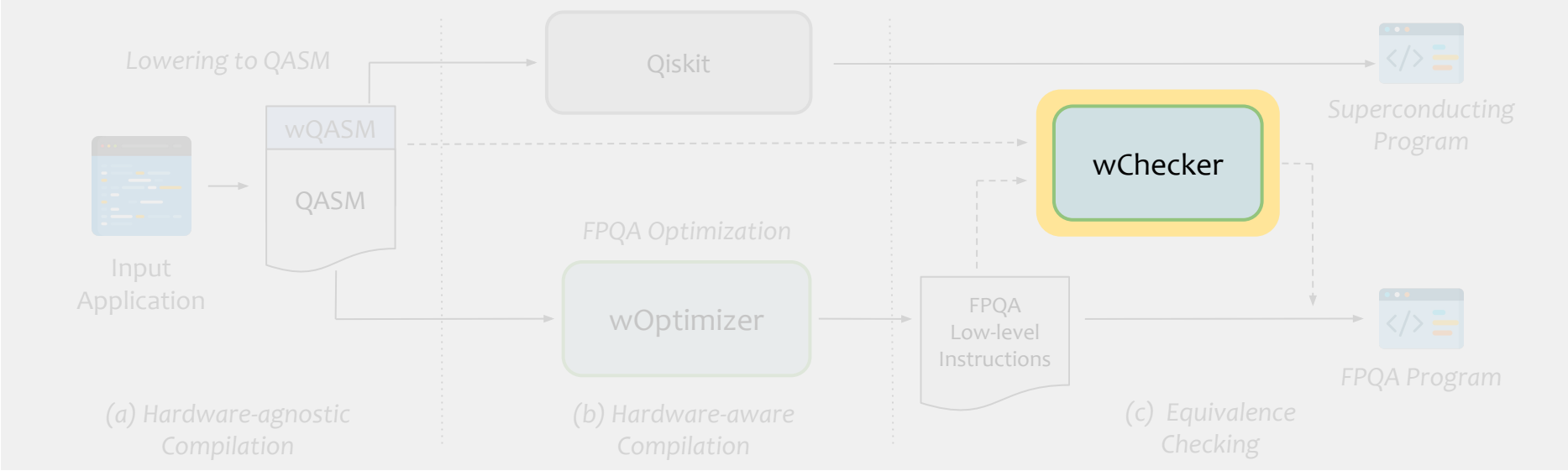


wOptimizer: Leveraging FPQAs - Showcase

- Are there actual quantum programs where hardware-specific optimizations are **beneficial?**
  - Max-3SAT: $(x_0 \vee x_1 \vee x_2) \wedge (x_3 \vee x_4 \vee x_5) \wedge (\neg x_2 \vee \neg x_4 \vee \neg x_5)$
- Unlike others, FPQAs offer native support for higher multi-qubit gates (3+)



System Overview

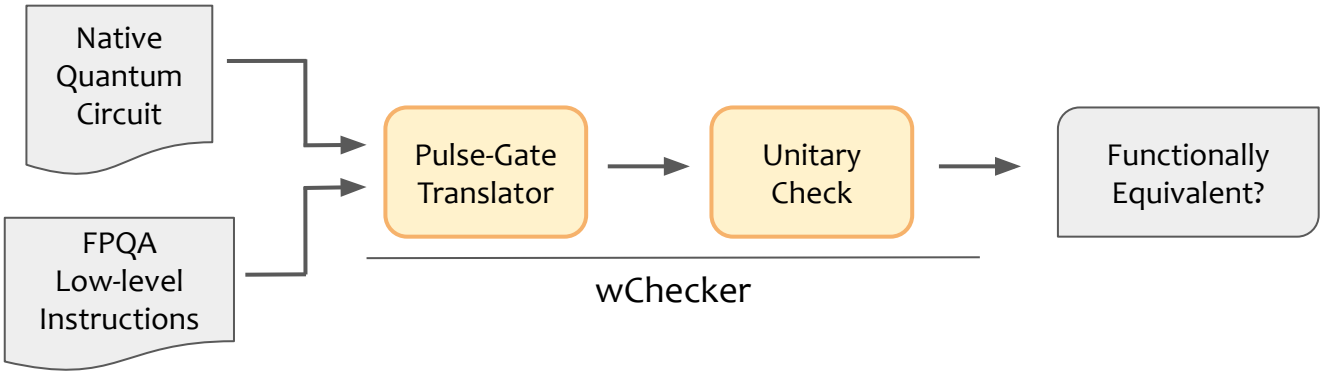


Challenge #3: Correctness

Challenge	Our approach
• Ensure hardware-specific equivalence	→ FPQA instructions translation
• Fast equivalence checking to avoid compilation delays	→ Efficient one to one gate comparison

Our contribution:
wChecker ensures functional equivalence after code optimizations

wChecker: A Functional Equivalence Checker



- Pulse-Gate Translator
  - Translates FPQA-specific pulses back into universal quantum gates
- Unitary Check
  - Compares gates one to one between the native and the translated circuit

Outline

- ~~Motivation~~
- ~~Design~~
- **Evaluation**

Research Questions

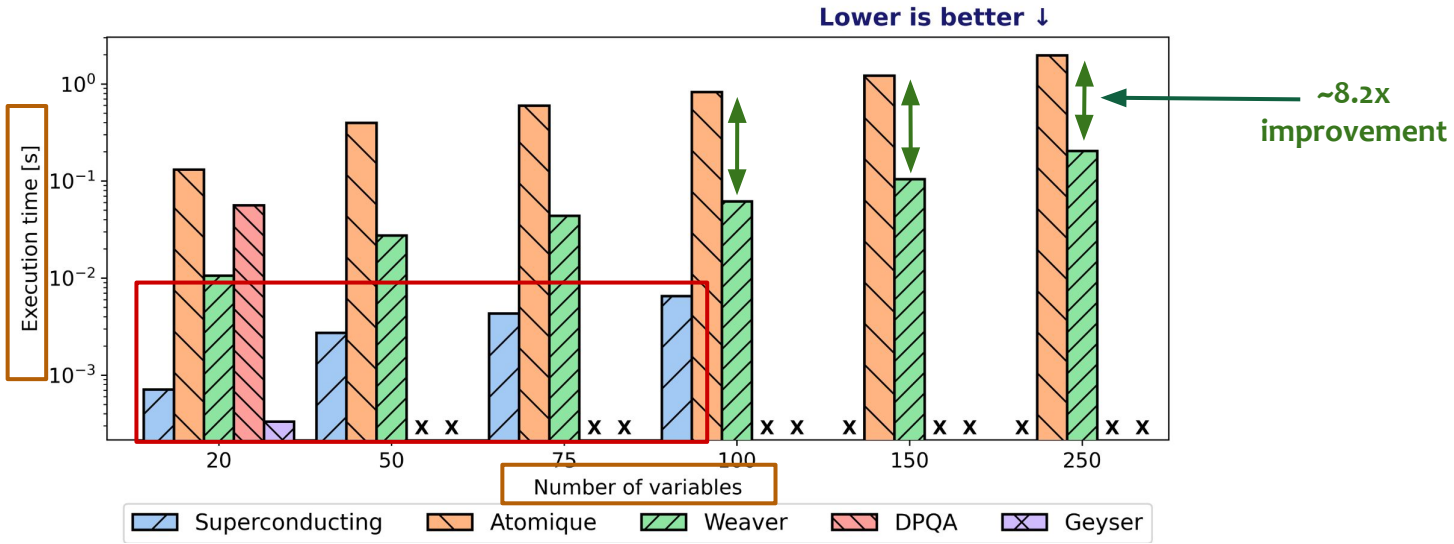
- **RQ #1:** What is Weaver's execution time?
- **RQ #2:** What is the fidelity improvement?
- **RQ #3:** What is the compilation time overhead?

Evaluation Methodology

- Benchmarks:
  - QAOA circuits of MAX-3SAT formulas from the SATLIB benchmark
  - Varying the number of variables (20, 50, 75, 100, 150, 250)
- Baselines:
  - Superconducting (Qiskit)
  - DPQA [ICCAD '22]
  - Atomique [ISCA '24]
  - Geyser [ISCA '22]

RQ #1: Execution Time

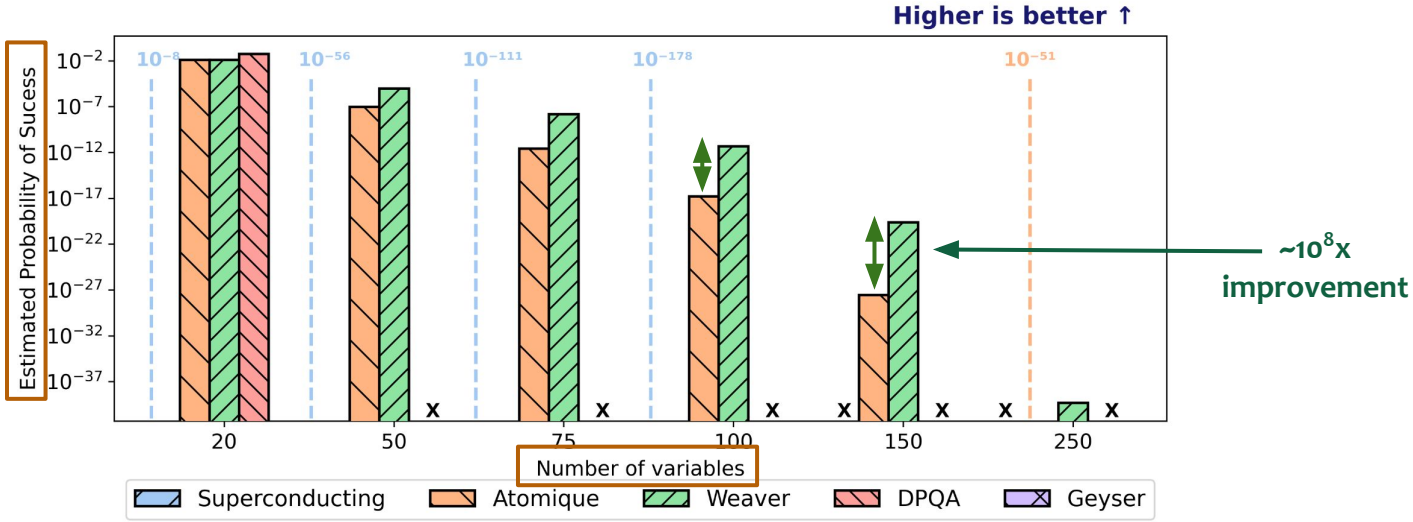
Hypothesis: Weaver aims for a highly parallelized execution, decreasing execution time



Weaver produces solutions with the fastest execution on large benchmarks

RQ #2: Fidelity (EPS)

Hypothesis: Weaver reduces the number of gates which should improve overall fidelity



Weaver achieves exponential increase in fidelity compared to the baselines

Summary

How can we **retarget** applications to different quantum technologies while leveraging their **unique** hardware capabilities?

Weaver: A retargetable compiler framework for FPQA quantum architectures

- OpenQASM extension with FPQA-specific instructions
- FPQA-specific optimization
- Compilation correctness with functional equivalence checking

Try it out!

Weaver (source code)