

#23: OpenQASM 3.0 Reference Python Implementation | “QAMP-21”

Mentored by – Jack Woehr

Abeer Vaishnav, Adrien Suau, Vishal Bajpe

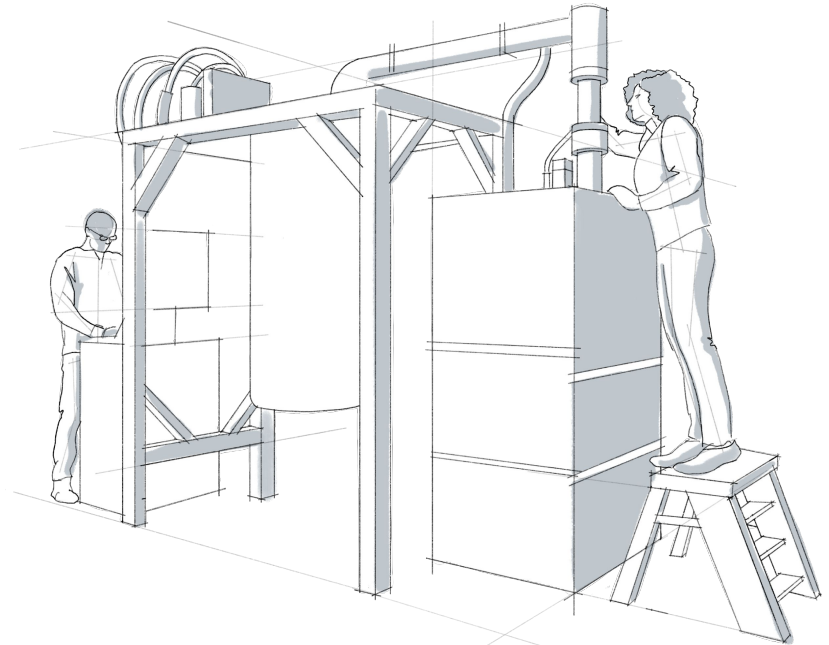
```
from qiskit import QuantumCircuit, execute
from qiskit import Aer, IBMQ
from qiskit.providers.aer.noise import NoiseModel

# Choose a real device to simulate from IBMQ provider
provider = IBMQ.load_account()
backend = provider.get_backend('ibmq_vigo')
coupling_map = backend.configuration().coupling_map

# Generate an Aer noise model for device
noise_model = NoiseModel.from_backend(backend)
basis_gates = noise_model.basis_gates

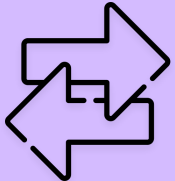
# Generate 3-qubit GHZ state
num_qubits = 3
circ = QuantumCircuit(3, 3)
circ.h(0)
circ.cx(0, 1)
circ.cx(1, 2)
circ.measure([0, 1, 2], [0, 1, 2])

# Perform noisy simulation
backend = Aer.get_backend('qasm_simulator')
job = execute(circ, backend,
              coupling_map=coupling_map,
              noise_model=noise_model,
              basis_gates=basis_gates)
result = job.result()
print(result.get_counts(0))
```



Why OpenQASM?

Convenient &
standardised format
for quantum circuits



Hardware-agnostic
representations



Closer to real
hardware



Straightforward syntax



Upgrades in OpenQASM 3.0

A complete language for quantum circuits now with salient features as compared to OpenQASM 2.0:

- Complete type system (constants, variables, operators, casting, expressions, ...)
- Control flow statements
- Support for versatile circuit and operation expression
- Dynamic circuit subroutines and external function calls
- Support for lower level operation definition
- Extended grammar for pulse operations

Physical level

`delay` statements, adding relative timings to operation

Type `stretch` to resolve concrete durations at compile time for granular calibration

Support for qubit-specific calibration instructions via `defcal` construct

Design Philosophy & execution

Arbitrary classical control flow, gate modifiers and timings

Ability to perform new kinds of circuits and experiments

Pulse level calibration and **multi level optimization**

Logical Level

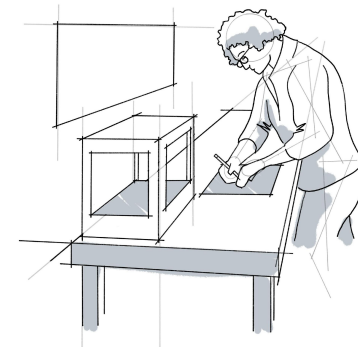
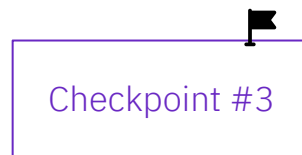
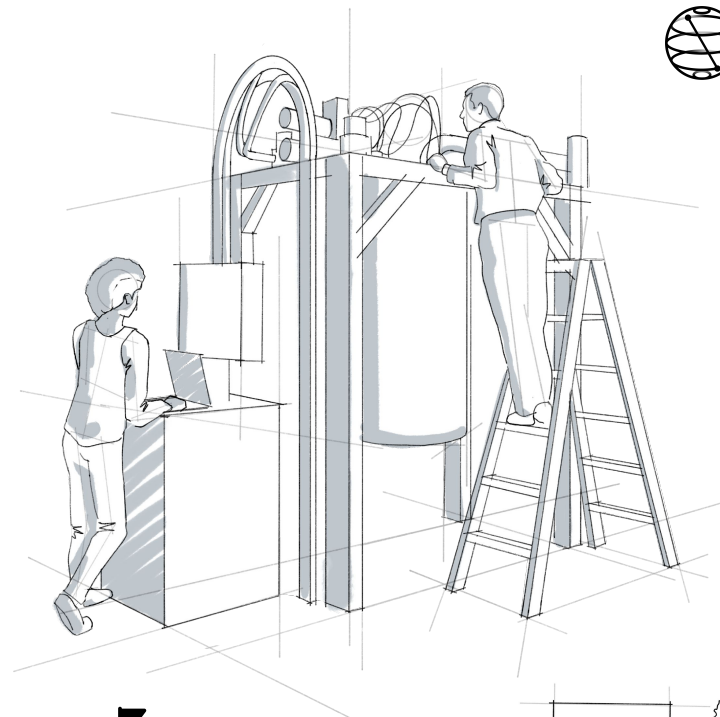
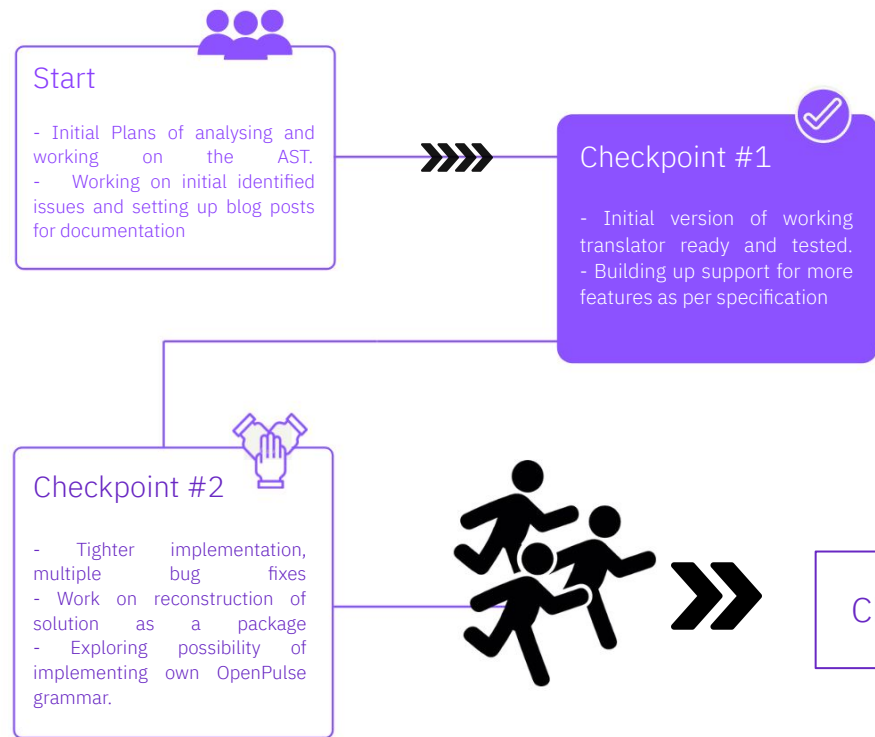
Ability to use quantum-classical dependencies in quantum circuits

Native support for classical computation on measurement results

Robusts classical types with support for classical control flow

Support for `int`, `uint`, `float`, `bool`, and `bit` for classical types with functionality to specify a type with exact bit-precision for **low-level** and **bitwise** operations

Progress so far:

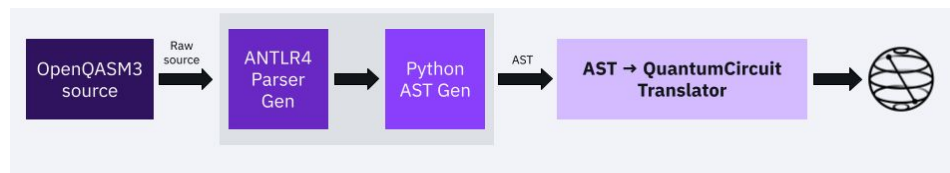


Checkpoint #3 Update

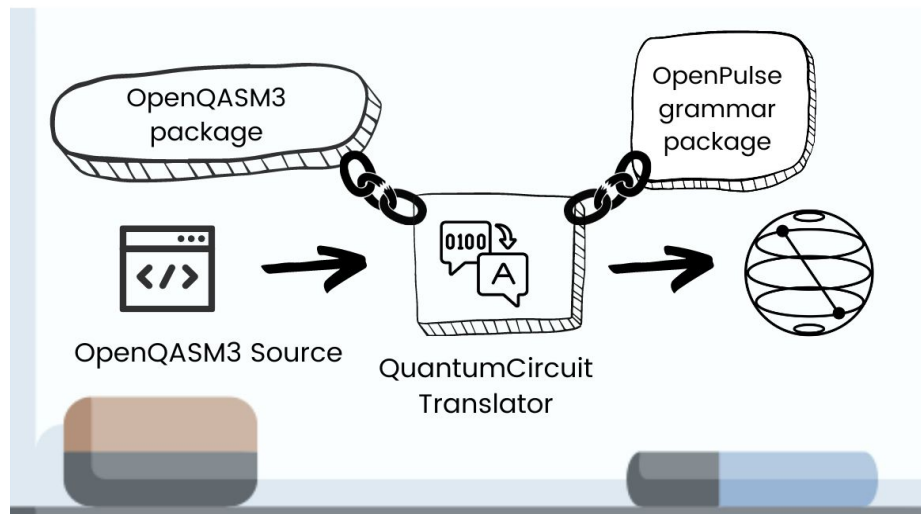
Architecture



Current



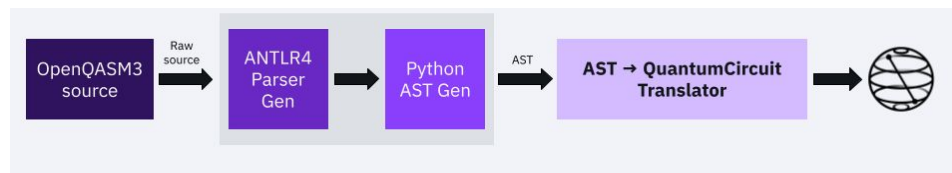
Proposed defca1 support (incomplete)



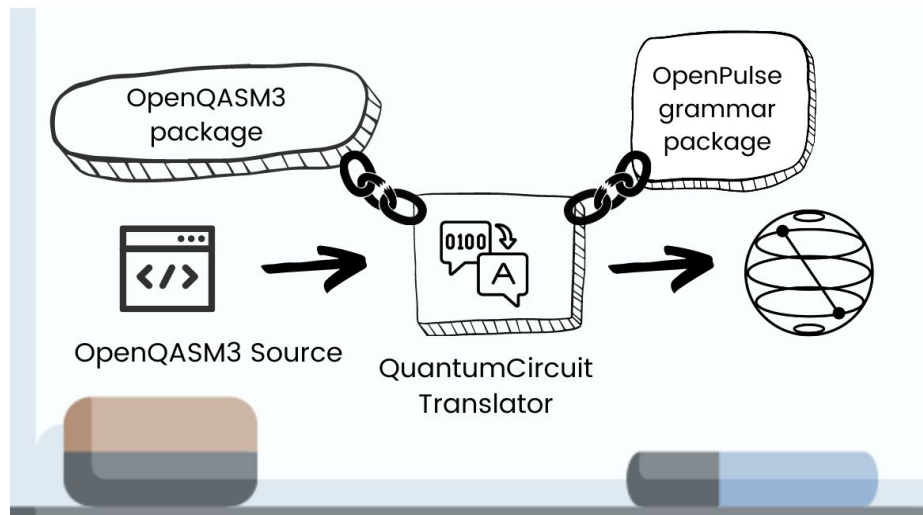
Architecture



Current



Proposed defca1 support (incomplete)



Challenges faced by the team:

- ❑ Frequent changes in the specification and parser to reflect the active work of the OpenQASM 3.0 TSC.
- ❑ Missing features:
 - PyPi packaging for reference AST
 - defcal grammar for OpenPulse
- ❑ Rich and complex type system continuously evolving



Summary

PR Contributions



PR #295

- ❑ Identifying an important hole in the specification about mathematical functions
- ❑ Addition of the power (**) operator for complex numbers

PR #269

- ❑ Extensive testing on various inputs
- ❑ Remarks to improve overall ease of use
- ❑ A few defects raised to the attention of the authors

PR #296

- ❑ Following the progressive definition of OpenPulse grammar
- ❑ Waiting for advances from the main team

PR #288

- ❑ Automatic packaging of the OpenQASM 3.0 parser using GitHub Actions
- ❑ Waiting for the “openqasm” package to be available

Issue Discussions:



Issue #296

Following discussion with @taalexander at IEEE Quantum Week on OpenPulse grammar implementation and Jacks further discussions, team is now added to the #openpulse closed group for discussion with guidance from people and teams working on OpenPulse.

Issue #304

Bug report for openqasm3 PyPi package import behaviour.

How are we bringing about changes?

- Tight integration with the OpenQASM 3 Technical Steering Committee (TSC) OpenQASM3 reference AST

- Several contributions to Qiskit/openqasm repo while working on the Translator

Open Casting angles to floats - slow and precise or quick and lossy? #280
mbhealy opened this issue 13 days ago · 4 comments

would really love to see the ability to be able to precision and provide a specific operator or indicator to ensure better precision. In C, I have seen where people will explicitly encode the floating point number they want to ensure speed&accuracy. Something like that might be a useful addition as well.

Hope this helps some :)

jwoehr commented 2 days ago Contributor

Some folks concerned with this issue might be interested in QAMP Fall 2021 Team No. 23's ways of handling this which will be presented in our Oct. 7 session for QAMP checkpoint.

Open Reference implementation of OpenQASM 3.0 AST in Python #269
aspcompiler wants to merge 53 commits into @qiskit:master from aspcompiler:ast

AbeerVaishnav13 commented 7 days ago

Hi @aspcompiler, we were in the process of using your QASM3 to AST generator for the Qiskit Advocate Mentorship Program project on OpenQASM3 and noticed a few bugs/potential changes...

1. No restriction on re-assignment to const types

Presently there are no restrictions on re-assignment to const types. For example, if I write the following code:

```
const my_const = 10; // parses as a 'ConstantDeclaration' statement
my_const = 5; // parses as a 'ClassicalAssignment' statement - WRONG
```

Shouldn't the AST gen step throw an error for statement-2 i.e. re-assignment to an immutable identifier?

2. 'bool' instead of NoDesignatorTypeName.bool

Open Reference implementation of OpenQASM 3.0 AST in Python #269
aspcompiler wants to merge 53 commits into @qiskit:master from aspcompiler:ast

nelimee commented 19 days ago

Hi @aspcompiler,

We started to use your AST generator for the Qiskit Advocate Mentorship Program project on OpenQASM3 and found some things that might be considered as issues.

Here is a quick description of each, sadly I did not have the time to investigate the source of these issues yet:

```
# Code to get the AST from one of the example files in https://github.com/aspcompiler/openqasm/tree/ast/examples
from openqasm.parser.antlr.qasm_parser import parse

file = "adder.qasm"
with open(file, "r") as f:
    qasm_str = f.read()

ast = parse(qasm_str)
```

Open Reference implementation of OpenQASM 3.0 AST in Python #269
aspcompiler wants to merge 53 commits into @qiskit:master from aspcompiler:ast

jakeshman commented 7 days ago · edited · Contributor

@AbeerVaishnav13, some quick answers, since I'm looking at this (but am not on the main author team).

- reassignment to const: that probably isn't the role of the AST to throw an error on that, because AST generation is more about syntax than programme correctness. The first pass of AST generation (which the reference implementation here is) doesn't generally track which identifiers are being assigned, and what their types are - that would most likely come in a later, compilation stage.
- cast operator doesn't preserve enum: that looks like a bug to me.

Fixed NoDesignatorType parsing ✓ @93ddd

aspcompiler commented 7 days ago Contributor Author

@AbeerVaishnav13: @jakeshman just answered 1. I just pushed a fix for 2.

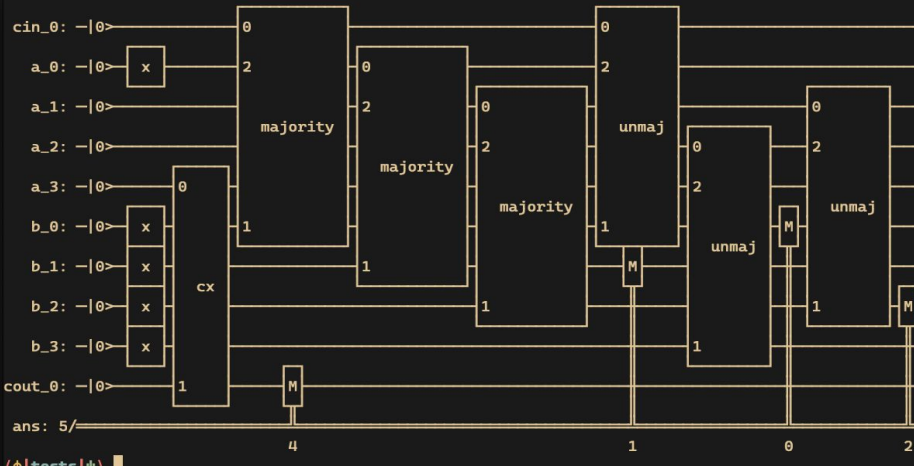
shiyunon and others added 5 commits 18 days ago

- fixed some issues raised in #269 ✗ ad0a9ce
- Small subtle adjustment to the identifier list and constant declaration ✗ 347841c
- adding a code snippet from adder.qasm as a new test case. ✗ 5696e17
- Merge branch 'master' into ast 2ccb56a
- Removed fixed type to sync up with grammar ✓ f380a95

Demo

```
*adder.qasm *alignment.qasm *cphase.qasm *dd.qasm >... buffers (phi|tests|psi) python3 build_ast.py ../../examples/adder.qasm -I ../../examples/ -t
```

```
18 */
17 OPENQASM 3;
16 include "stdgates.inc";
15
14 gate majority a, b, c {
13   cx c, b;
12   cx c, a;
11   ccx a, b, c;
10 }
9
8 gate unmaj a, b, c {
7   ccx a, b, c;
6   cx c, a;
5   cx a, b;
4 }
3
2 qubit[1] cin;
1 qubit[4] a;
22 qubit[4] b;
1 qubit[1] cout;
2 bit[5] ans;
3 uint[4] a_in = 1; // a = 0001
4 uint[4] b_in = 15; // b = 1111
5 // initialize qubits
6 reset cin;
7 reset a;
8 reset b;
9 reset cout;
10
11 // set input states
12 for i in [0: 4] {
13   if(bool(a_in[i])) x a[i];
14   if(bool(b_in[i])) x b[i];
15 }
16 // add a to b, storing result in b
17 majority cin[0], b[0], a[0];
18 for i in [0: 2] { majority a[i], b[i + 1], a[i + 1]; }
19 cx a[3], cout[0];
20 for i in [2: -1: 0] { unmaj a[i], b[i+1], a[i+1]; }
21 unmaj cin[0], b[0], a[0];
22 measure b[0] -> ans[0];
23 measure b[1] -> ans[1];
24 measure b[2] -> ans[2];
25 measure cout[0] -> ans[4];
```



```
(phi|tests|psi) █
```

Published translator package

Check it out here:

<https://test.pypi.org/project/qamp2021-test-openqasm3/>

⚠ You are using TestPyPI – a separate instance of the Python Package Index that allows you to try distribution tools and processes without affecting the real index.

qamp2021-test-openqasm3 0.1.0

✓ Latest version

Released: Dec 3, 2021

```
pip install -i https://test.pypi.org/simple/ qamp2021-test-openqasm3
```

Reference OpenQASM AST in Python

Navigation

- Project description
- Release history
- Download files

Project links

- Homepage

Statistics

Project description

OpenQASM 3 Python Reference

license Apache-2.0 pyPI v0.0.0

The `openqasm3` package contains the reference abstract syntax tree (AST) for representing OpenQASM 3 programs, tools to parse text into this AST, and tools to manipulate the AST.

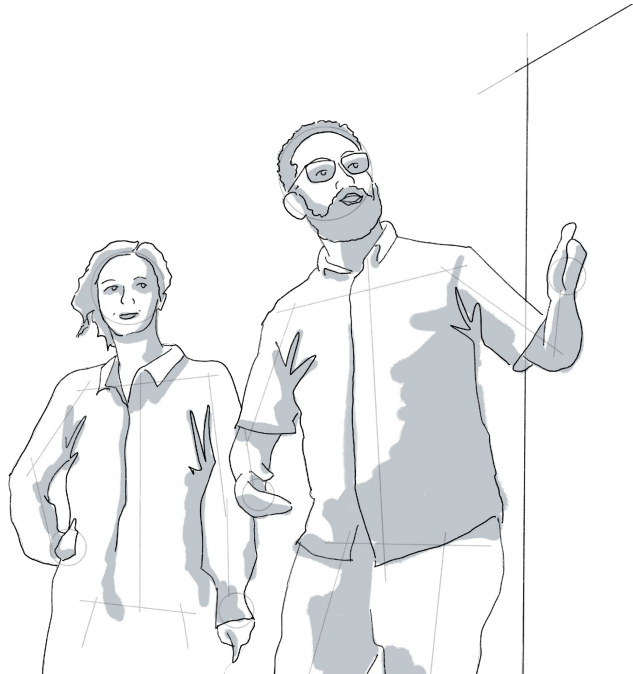
The AST is intended to help with writing compiler passes for OpenQASM 3 in Python. It aims to have no dependencies for users who consume the Python tree structure, and minimal dependencies for parsing a string to this tree structure. The AST is simpler than a Concrete Syntax Tree (CST) which preserves comments, spacing, etc for use by editor plugins.

The package consists of the modules:

IEEE Quantum Week - Progress Presentation



IEEE International Conference
on Quantum Computing
and Engineering — QCE21

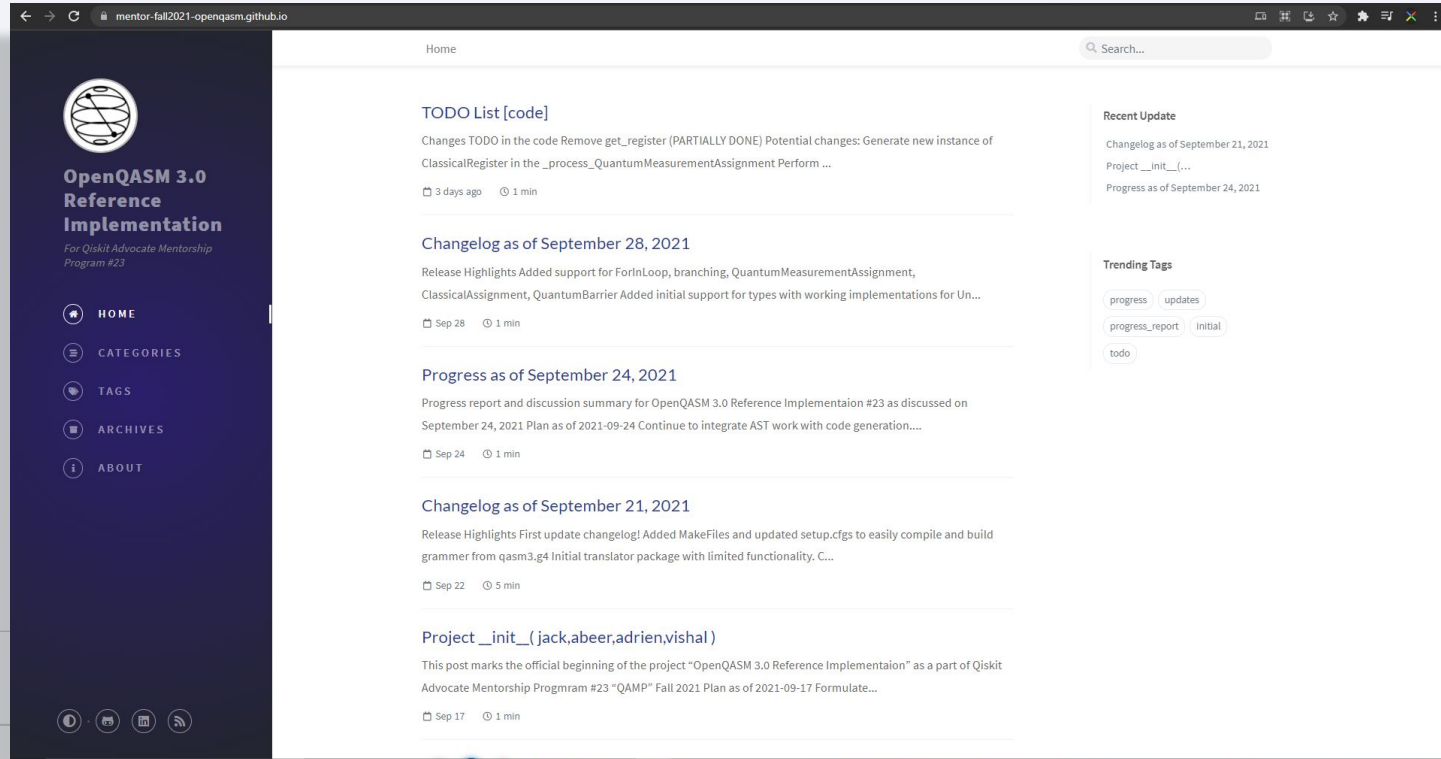


Join IBM Quantum on 10/19 at 12:15-1:00 pm MDT to learn about Qiskit Advocates. The Qiskit Advocate program is a global program that provides support to individuals actively contributing to Qiskit. Being a part of this enthusiastic community offers mentorship with experts on specific projects, networking, and priority access to events. Stop by to meet Qiskit Advocates, learn about the program, and how to get involved.

Dedicated blog website

Check it out here:

<https://mentor-fall2021-openqasm.github.io/>



Next steps

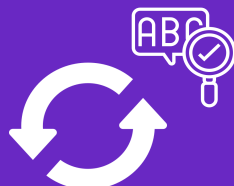
Use the newly introduced classical control Instruction in `QuantumCircuit`



Package our translator to easy installation and use



Update of the typing system with the latest specifications

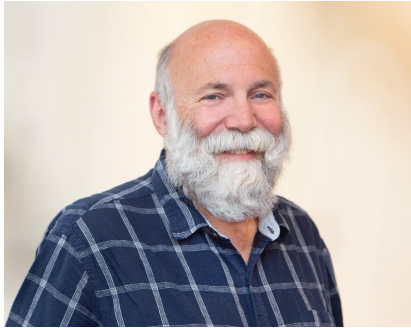


Better error reporting for easy debugging



QAMP #23 - Team

- Jack Woehr (Mentor)
IBM Champion 2021



- Abeer Vaishnav



- Adrien Suau



- Vishal Bajpe

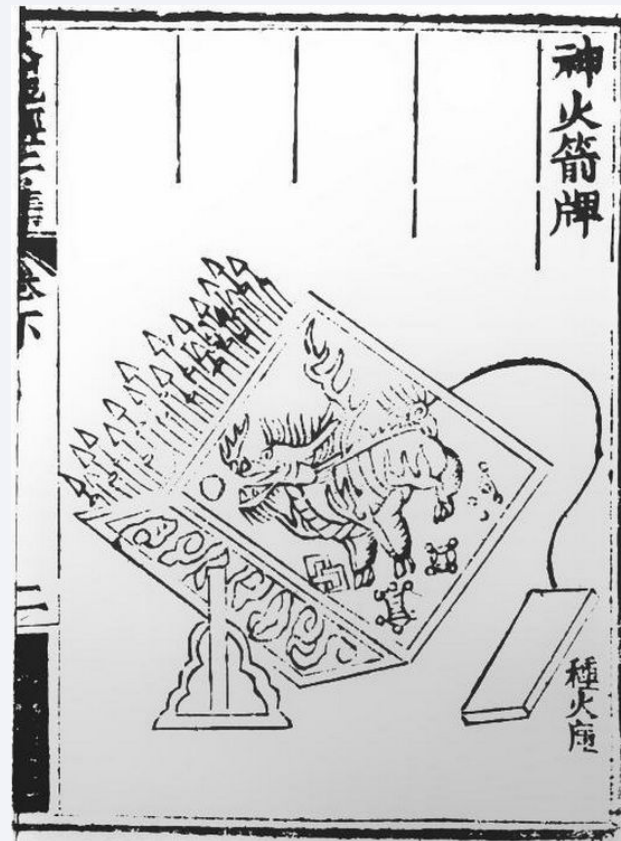
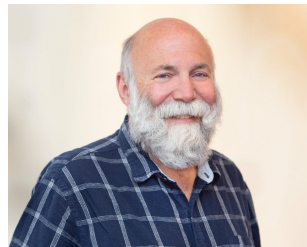


Today is Yesterday's Tomorrow

In the course of 20 years' dilettante interest in Quantum Computing, I have seen the field progress from what Nobel laureate [Bill Phillips](#) called “A 50-50 chance: 50% chance in 50 years” to a world-wide collaborative research project involving the brightest young minds of every habitable continent.

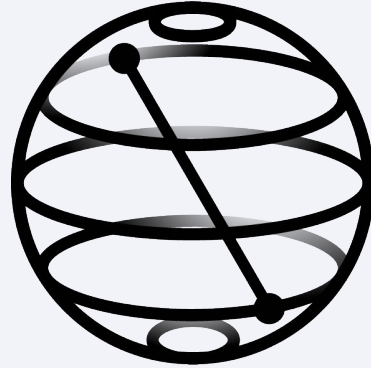
It has been a pleasure and a privilege to participate in QAMP Fall 2021 with this stellar team! Best of luck in your careers!

- Jack Woehr



*Depiction of a
fire arrow
launcher, or
shen huo chien
pai, from the
Ming Dynasty
book Huo Long
Jing
Wikimedia
Public Domain*

Thank you for watching!



Merci! धन्यवाद!