Qasm exporter creates invalid Qasm for subcircuits with classical registers #7750

Edit New issue

Closed **ANONYMOUS AUTHOR** opened this issue on Mar 9, 2022 · 3 comments

ANONYMOUS AUTHOR commented on Mar 9, 2022

Environment

- **Qiskit Terra version**: 0.19.1
- **Python version**: 3.8
- **Operating system**: Ubuntu 18.04.6 LTS

What is happening?

Converting a circuit with a subcircuit to qasm fails, when this subcircuit contains also classical registers.

How can we reproduce the issue?

Run this:

```
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister
qr = QuantumRegister(2, name='qr')
cr = ClassicalRegister(2, name='cr')
qc = QuantumCircuit(qr, cr, name='qc')
subcircuit = QuantumCircuit(qr, cr, name='subcircuit')
subcircuit.x(qr[0])
qc.append(subcircuit, qargs=qr, cargs=cr)
qc.measure(qr, cr)
qc.draw(fold=-1)
```

Output:

```
      ┌──────────────┐┌─┐
qr_0: ┤0             ├┤M├──
      │              │└╥┘┌─┐
qr_1: ┤1             ├─╫─┤M├
      │  subcircuit  │ ║ └╥┘
cr_0: ┤0             ├─╫──╫─
      │              │ ║  ║
cr_1: ┤1             ├─╫──╫─
      └──────────────┘ ╚══╩═
```

Convert to qasm:

```
qc.qasm(formatted=True)
```

Output:

```
OPENQASM 2.0;
include "qelib1.inc";
gate subcircuit q0,q1 { x q0; }
qreg qr[2];
creg cr[2];
subcircuit qr[0],qr[1],cr[0],cr[1];
measure qr[0] -> cr[0];
measure qr[1] -> cr[1];
```

Now, if we want to reuse this qasm and load it:

```
qc = QuantumCircuit.from_qasm_str(qc.qasm())
```

We get the following error:

```
qiskit/circuit/quantumcircuit.py in from_qasm_str(qasm_str)
   2362         """
   2363         qasm = Qasm(data=qasm_str)
-> 2364         return _circuit_from_qasm(qasm)
   2365
   2366     @property
```

Assignees

No one assigned

Labels

bug help wanted mod: qasm2

Projects

Contributor Monitoring

Status: Done +5 more

Milestone

No milestone

Development

No branches or pull requests

Notifications Customize

Unsubscribe

You're receiving notifications because you authored the thread.

2 participants

```
qiskit/circuit/quantumcircuit.py in _circuit_from_qasm(qasm)
   4695     from qiskit.converters import dag_to_circuit
   4696
-> 4697     ast = qasm.parse()
   4698     dag = ast_to_dag(ast)
   4699     return dag_to_circuit(dag)

qiskit/qasm/qasm.py in parse(self)
     51         with QasmParser(self._filename) as qasm_p:
     52             qasm_p.parse_debug(False)
---> 53             return qasm_p.parse(self._data)

qiskit/qasm/qasmparser.py in parse(self, data)
   1138     def parse(self, data):
   1139         """Parse some data."""
-> 1140         self.parser.parse(data, lexer=self.lexer, debug=self.parse_deb)
   1141         if self.qasm is None:
   1142             raise QasmError("Uncaught exception in parser; " + "see previous messages for details.")

ply/yacc.py in parse(self, input, lexer, debug, tracking, tokenfunc)
    331             return self.parseopt(input, lexer, debug, tracking, tokenfunc)
    332         else:
--> 333             return self.parseopt_notrack(input, lexer, debug, tracking, tokenfunc)
    334
    335

ply/yacc.py in parseopt_notrack(self, input, lexer, debug, tracking, tokenfunc)
   1118                           del symstack[-plen:]
   1119                           self.state = state
-> 1120                           p.callable(pslice)
   1121                           del statestack[-plen:]
   1122                           symstack.append(sym)

qiskit/qasm/qasmparser.py in p_unitary_op_2(self, program)
    704         """
    705         program[0] = node.CustomUnitary([program[1], program[2]])
--> 706         self.verify_as_gate(program[1], program[2])
    707         self.verify_reg_list(program[2], "qreg")
    708         self.verify_distinct([program[2]])

qiskit/qasm/qasmparser.py in verify_as_gate(self, obj, bitlist, arglist)
    158
    159         if g_sym.n_bits() != bitlist.size():
--> 160             raise QasmError(
    161                 "Gate or opaque call to '" + obj.name + "' uses",
    162                 str(bitlist.size()),

QasmError: "Gate or opaque call to 'subcircuit' uses 4 qubits but is declared for 2 qubits line 6 file "
```

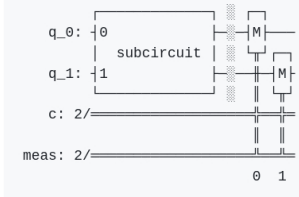
What should happen?

I would have expected the exported qasm to be a valid qasm (aka to be read again) and also give me back the original exported circuit if I decide to store my circuit on a file as qasm and pass it to someone else.

Any suggestions?

I would have expected the following QASM:

```
qasm_expected = """
OPENQASM 2.0;
include "qelib1.inc";
gate subcircuit q0,q1 { x q0; }
qreg q[2];
creg c[2];
creg meas[2];
subcircuit q[0],q[1];
barrier q[0],q[1];
measure q[0] -> meas[0];
measure q[1] -> meas[1];
"""
qc = QuantumCircuit.from_qasm_str(qasm_expected)
qc.draw(fold=-1)
```



But this is also inaccurate, since in the original `subcircuit` might as well use the classical registers to perform some measurement, whereas with this qasm representation it is not possible to express this.
Thus, I am not sure on how we should proceed and I very curious to listen to your feedback on this.
Thanks in advance

Add tasklist

ANONYMOUS AUTHOR added the bug label on Mar 9, 2022

jakelishman commented on Mar 9, 2022 • edited — Member

There is no possible valid OpenQASM 2 programme for the circuit you're describing; there's no subroutine-like construct that can take classical parameters. The error message could be better, but there is no way to produce valid OpenQASM 2 in this situation, because the object has no way of being represented. The bug here is that the QASM 2 exporter should have rejected the circuit out-of-hand.

I haven't looked at the QASM 2 exporter code for a while, but I think it tries to export all `Instruction` instances currently, whereas it should fail on anything other than a special case or a gate, due to limitations in the QASM 2 language. Probably the fix involves modifying the exporter step that tries to find the definition of each object in a circuit so that it goes through the following steps:

- if a `Gate` instance, proceed as it currently does
- if `Barrier`, `Measure` or `Reset`, output the specialist QASM 2 statements
- if not a `Gate` (i.e. an unknown `Instruction`), then fail with a message saying that QASM 2 cannot represent non-unitary operations (except for the built-in `measure` and `reset`).

The way you have added the subcircuit to your circuit, it's as an `Instruction` not a `Gate`, so even if it had *no* classical registers, in my new scheme it would still be rejected as non-unitary (that's the difference between `Instruction` and `Gate`). You should call `QuantumCircuit.to_gate` to avoid that. This is an important note for API stability in the bug fix, though - people may be relying on the QASM 2 exporter working in similar situations, so we should take care to check that we don't break anybody's workflow.

OpenQASM 3 can represent non-unitary subroutines, but at the moment Terra's QASM 3 exporter is quite limited in what it can support, since that language spec is still evolving and so is Terra's capability to represent dynamic circuits.

1

jakelishman added help wanted mod: qasm2 labels on Mar 9, 2022

jakelishman mentioned this issue on May 16, 2022

A translation about `initialize` from qiskit in OpenQASM cannot translate back into qiskit #8048 — Open

javabster added this to **Contributor Monitoring** on Jun 21, 2022

javabster moved this to **Tagged but unassigned** in **Contributor Monitoring** on Aug 16, 2022

jakelishman mentioned this issue on Apr 20

Snapshot definition not included in generated qasm #2195 — Closed

ANONYMOUS AUTHOR commented 4 days ago — Author

I attach this bug report here, since closely related (instead of opening a new one)

Environment

- **Qiskit Terra version**: 0.43.1 meta package, terra 0.24.1
- **Python version**: 3.10
- **Operating system**: docker continuumio/miniconda3

What is happening?

Exporting a circuit with a sub-circuit via `append()` and that uses `measure()` leads to an invalid qasm file, generating error when imported again.

How can we reproduce the issue?

Run this python script:

```
from qiskit import QuantumCircuit
q = QuantumCircuit(2)
q.initialize('00')
print(q.qasm(filename="my.qasm"))
round_trip = QuantumCircuit.from_qasm_str(q.qasm())
```

Produces this output and error:

```
Converting to qasm and back for circuit circuit-114
     ┌───┐     ┌─┐   ┌─────────────┐
q_0: ┤ H ├──■──┤M├───┤0            ├
     └───┘┌─┴─┐└╥┘┌─┐│             │
q_1: ─────┤ X ├─╫─┤M├┤1            ├
          └───┘ ║ └╥┘│  circuit-115 │
c_0: ═══════════╩══╬═╡0            ├
                   ║ │             │
c_1: ══════════════╩═╡1            ├
                     └─────────────┘
OPENQASM 2.0;
include "qelib1.inc";
gate circuit_115 q0,q1 { h q0; cx q0,q1; measure q0; measure q1; }
qreg q[2];
creg c[2];
h q[0];
cx q[0],q[1];
measure q[0] -> c[0];
measure q[1] -> c[1];
circuit_115 q[0],q[1],c[0],c[1];

Error near line 3 Column 15
Traceback (most recent call last):
  File "myfile.py", line 16, in <module>
    QuantumCircuit().from_qasm_str(circuit2.qasm())
  File "...qiskit/circuit/quantumcircuit.py", line 2529, in from_qasm_str
    return _circuit_from_qasm(qasm)
  File "...qiskit/circuit/quantumcircuit.py", line 4964, in _circuit_from_qasm
    ast = qasm.parse()
  File "...qiskit/qasm/qasm.py", line 53, in parse
    return qasm_p.parse(self._data)
  File "...qiskit/qasm/qasmparser.py", line 1137, in parse
    self.parser.parse(data, lexer=self.lexer, debug=self.parse_deb)
  File "...ply/yacc.py", line 333, in parse
    return self.parseopt_notrack(input, lexer, debug, tracking, tokenfunc)
  File "...ply/yacc.py", line 1120, in parseopt_notrack
    p.callable(pslice)
  File "...qiskit/qasm/qasmparser.py", line 397, in p_id_e
    raise QasmError("Expected an ID, received '" + str(program[1].value) + "'")
qiskit.qasm.exceptions.QasmError: "Expected an ID, received 'measure'"
```

The qasm contains also a second error leading to an error, thus the fact that `circuit_115` is defined with 2 qubits, but then it is used with 4 arguments (see: #7750 (comment))

What should happen?

I would expect a valid qasm since the circuit is theoretically representable in qasm.

Any suggestions?

What about expanding the definition and injecting the basic instructions directly in the main code, without a separate definition, thus avoiding the problem with the `measure` and number of arguments?
This would clearly lead to a bigger qasm file, but it would be valid.

jakelishman commented 4 days ago — Member

Thanks for the bug report - the second one is a duplicate of #8048. The first one was fixed already by #10438, so I'll close this issue now.

jakelishman closed this as completed 4 days ago

github-project-automation bot moved this from **Tagged but unassigned** to **Done** in **Contributor Monitoring** 4 days ago

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Comment

Remember, contributions to this repository should follow its contributing guidelines, security policy, and code of conduct.