# yquant

Typesetting quantum circuits in a human-readable language

Benjamin Desef

2025-09-08

# qasm2circ (2004)

The Wayback Machine - https://web.archive.org/web/20191009061927/https://www.media.mit.edu/quanta/qasm2circ/

Quantum Architectures

Quantum Computer Architecture Resources [ home ]

**Quantum circuit viewer: qasm2circ**

QASM is a simple text-format language for describing acyclic quantum circuits composed from single qubit, multiply controlled single-qubit gates, multiple-qubit, and multiple-qubit controlled multiple-qubit gates.

*qasm2circ* is a package which converts a QASM file into a graphical depiction of the quantum circuit, using standard quantum gate symbols (and other user-defined symbols). This is done using latex (specifically, xypic), to produce high-quality output in epsf, pdf, or png formats.

Figures of quantum circuits in the book "Quantum Computation and Quantum Information," by Nielsen and Chuang, were produced using an earlier version of this package.

- Download distribution here (zip) or (tgz) (the distribution includes all the examples).
- Current version = 1.4 (Released 14-Mar-05)

[ home | Example 1 | Example 2 | Example 3 | Example 4 | Example 5 | Example 6 | Example 7 | Example 8 | Example 9 | Example 10 | Example 11 | Example 12 | Example 13 | Example 14 | Example 15 | Example 16 | Example 18 | Example 17 | qasm specification | Installation instructions ]

Live demo (MIT only)

---

## Example 1

[ test1.qasm |test1.png |test1.pdf |test1.eps |test1.tex ]

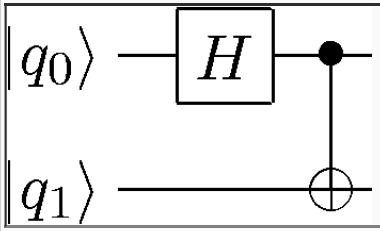
```
#
# File:   test1.qasm
# Date:   22-Mar-04
# Author: I. Chuang
```

## Example 1

[ test1.qasm |test1.png |test1.pdf |test1.eps |test1.tex ]

```
#
# File:   test1.qasm
# Date:   22-Mar-04
# Author: I. Chuang
#
# Sample qasm input file - EPR creation
#
        qubit   q0
        qubit   q1

        h       q0      # create EPR pair
        cnot    q0,q1
```




## Example 2

[ test2.qasm |test2.png |test2.pdf |test2.eps |test2.tex ]

```
#
# File:   test2.qasm
# Date:   29-Mar-04
# Author: I. Chuang
#
# Sample qasm input file - simple teleportation circuit
#
        qubit   q0
        qubit   q1
        qubit   q2
```

## Example 9

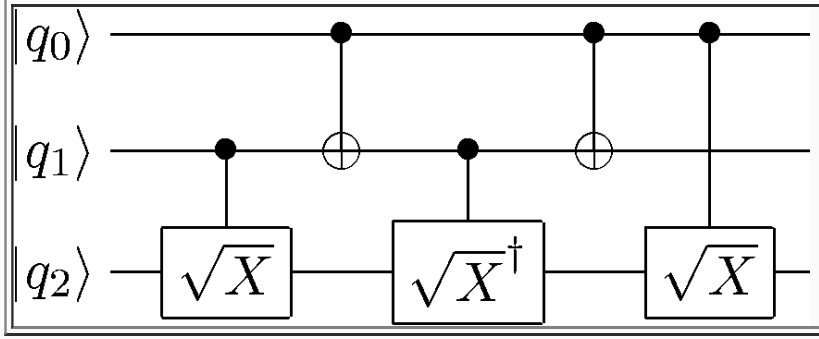
[ test9.qasm |test9.png |test9.pdf |test9.eps |test9.tex ]

```
#
# File:   test9.qasm
# Date:   22-Mar-04
# Author: I. Chuang
#
# Sample qasm input file - two-qubit gate circuit
# implementation of Toffoli

        def     c-X,1,'\sqrt{X}'
        def     c-Xd,1,'{\sqrt{X}}^\dagger'

        qubit   q0
        qubit   q1
        qubit   q2

        c-X     q1,q2
        cnot    q0,q1
        c-Xd    q1,q2
        cnot    q0,q1
        c-X     q0,q2
```



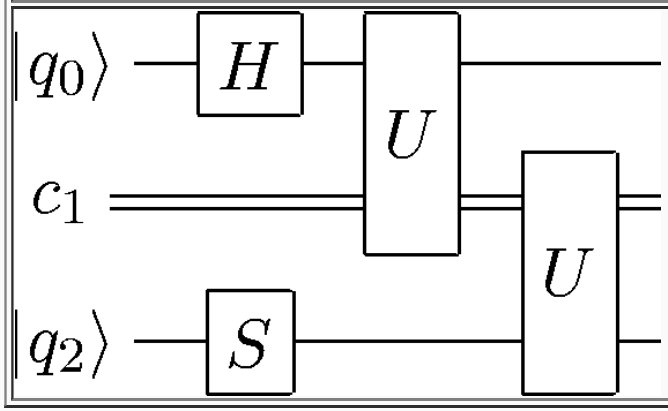
## Example 10

[ test10.qasm |test10.png |test10.pdf |test10.eps |test10.tex ]

```
#
# File:   test10.qasm
# Date:   22-Mar-04
# Author: I. Chuang
#
# Sample qasm input file - multi-qubit gates
# also demonstrates use of classical bits

        qubit   q0
        cbit    c1
        qubit   q2

        h       q0
        Utwo    q0,c1
        S       q2
        Utwo    c1,q2
```



# qasm2circ (2004)

## QASM Specification

QASM instructions are as follows. Lines begining with '#' are comments. All other lines should be of the form **op args** where op-args pairs are:

```
qubit   name,initval
cbit    name,initval
measure qubit
H       qubit
X       qubit
Y       qubit
Z       qubit
S       qubit
T       qubit
nop     qubit
zero    qubit
discard qubit
slash   qubit
dmeter  qubit
cnot    ctrl,target
c-z     ctrl,target
c-x     ctrl,target
toffoli ctrl1,ctrl2,target
ZZ      b1,b2
SS      b1,b2
swap    b1,b2
Utwo    b1,b2
space   qubit
def     opname,nctrl,texsym
defbox  opname,nbits,nctrl,texsym

Where:

def     - define a custom controlled single-qubit operation, with
          opname  = name of gate operation
          nctrl   = number of control qubits
```

# qasm2circ (2004)

```
------------------------------------------------------------------------------
REQUIREMENTS:

- latex2e with xypic (included in tetex)
- python version 2.3 or greater
- ghostscript (and epstopdf)
- netpbm (for creation of png files)

------------------------------------------------------------------------------
LIST OF FILES:

 README          - this file
 meter.epsf      - picture of measurement meter
 qasm2pdf        - script to create PDF from QASM file
 qasm2png        - script to create PNG from QASM file
 qasm2tex.py     - main python program to convert QASM to latex file
 samples         - directory containing examples
 xyqcirc.tex     - latex macros necessary to compile the latex files

------------------------------------------------------------------------------
INSTALLATION INSTRUCTIONS:

untar/unzip this distribution; it creates the directory qasm2circ with
all of the files.  Copy your qasm file into this directory, and run
qasm2pdf or qasm2png to create the desired output.  You may also run
"python qasm2tex.py foo.qasm" to generate just the tex file (it will
appear on stdout).

<!------------------------------------------------------------------------ ->
```

last version: 2013; originally DVI/PS workflow

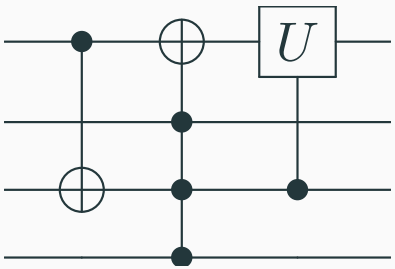
probably works with automatic conversion

PDF backend since 2010, some differences

Accessed  times

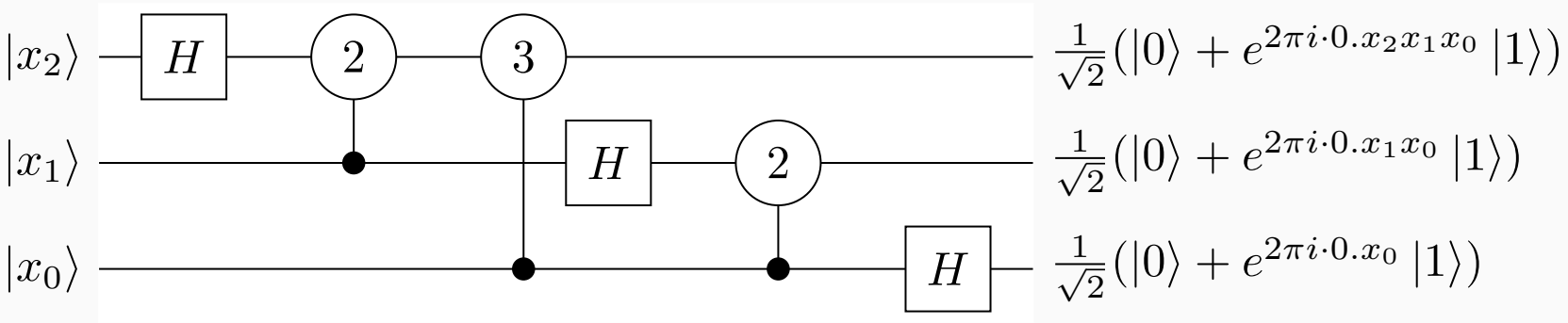
[ awcross at mit.edu ]

# qcircuit (2004 – 2021)



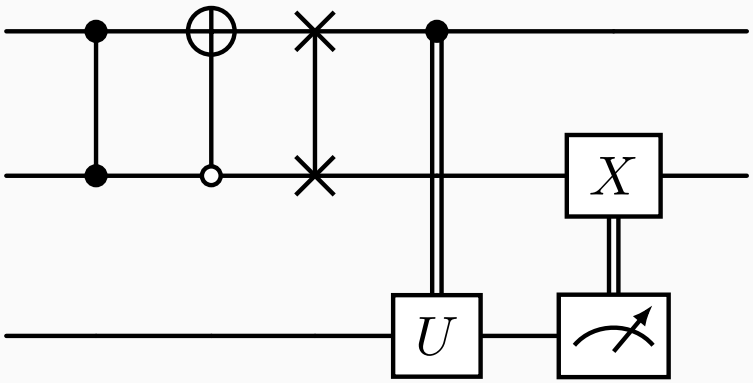

```
\Qcircuit @C=1em @R=.7em {
   & \ctrl{2} & \targ & \gate{U} & \qw \\
   & \qw & \ctrl{-1} & \qw & \qw \\
   & \targ & \ctrl{-1} & \ctrl{-2} & \qw \\
   & \qw & \ctrl{-1} & \qw & \qw
}
```

# qpic (2016 – 2023)




```
PREAMBLE \providecommand{\ket}[1]{\left|#1\right\rangle}
PREAMBLE \providecommand{\phase}[1]{e^{2{\pi}i\cdot#1}}
SCALE 1.5
a2 W \ket{x_2} \frac{1}{\sqrt{2}}(\ket{0}+\phase{0.x_2x_1x_0}\ket{1})
a1 W \ket{x_1} \frac{1}{\sqrt{2}}(\ket{0}+\phase{0.x_1x_0}\ket{1})
a0 W \ket{x_0} \frac{1}{\sqrt{2}}(\ket{0}+\phase{0.x_0}\ket{1})
a2 H
a2 P $2$ a1
a2 P $3$ a0
a1 H
a1 P $2$ a0
a0 H
```

# quantikz (2018 – 2023)




```
\begin{quantikz}
& \ctrl{1} & \targ{} & \swap{1} & \ctrl[vertical wire=c]{2} &&\\
& \control{} & \ctrl[open]{-1} & \targX{} && \gate{X} &\\
&&&& \gate{U} & \meter{} \wire[u][1]{c}
\end{quantikz}
```

This can be achieved by writing

```
qubit {$\ket{<name>_{\idx}}$} <name>[<len>];
```

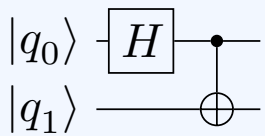
but if you want to realize this naming scheme for all circuits in your document, it is more convenient to say

```
\yquantset{register/default name=$\ket{\reg_{\idx}}$}
```

in the preamble, as is done here.

Note that `yquant` also directly supports the `qasm` syntax, see section 7.2.

**test1 (create an EPR pair)**

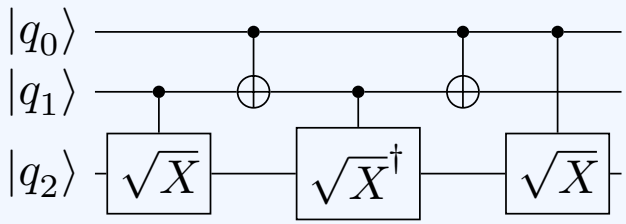



```
\begin{tikzpicture}
   \begin{yquant}
      qubit q[2];

      h q[0];
      cnot q[1] | q[0];
   \end{yquant}
\end{tikzpicture}
```

# yquant (2019 – 2025)

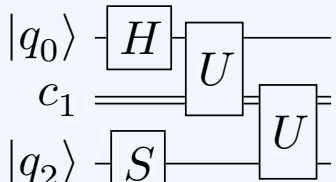
## test9 (two-qubit gate circuit implementation of Toffoli)




```
\begin{tikzpicture}
   \begin{yquant}
      qubit q[3];

      box {$\sqrt X$} q[2] | q[1];
      cnot q[1] | q[0];
      box {$\sqrt X^\dagger$} q[2] |
       ↪  q[1];
      cnot q[1] | q[0];
      box {$\sqrt X$} q[2] | q[0];
   \end{yquant}
\end{tikzpicture}
```

## test10 (multi-qubit gates also demonstrates use of classical bits)

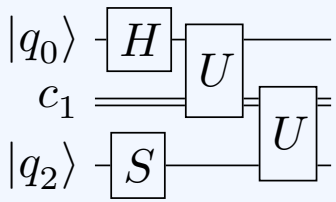



```
\begin{tikzpicture}
   \begin{yquant}
      qubit {$\ket{q_0}$} q;
      cbit {$c_1$} c;
      qubit {$\ket{q_2}$} q[+1];

      h q[0];
      box {$U$} (q[0], c);
```

```
   \end{yquant}
\end{tikzpicture}
```

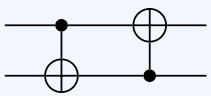
## test10 (multi-qubit gates also demonstrates use of classical bits)



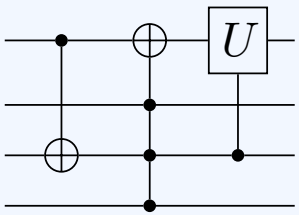

```
\begin{tikzpicture}
   \begin{yquant}
      qubit {$\ket{q_0}$} q;
      cbit {$c_1$} c;
      qubit {$\ket{q_2}$} q[+1];

      h q[0];
      box {$U$} (q[0], c);
      box {$S$} q[1];
      box {$U$} (c, q[1]);
   \end{yquant}
\end{tikzpicture}
```

Instead of a discontiguous vector register, we could also have used three scalar registers. The labels chosen for `qasm` do not fit well to the indices `yquant` assigns. We might also have used a three-register vector and used the `settype` pseudo-gate to immediately change the second register into a classical one, which would give indices matching the labels—but still, the registers would have a common name, which would make this a very unnatural approach.

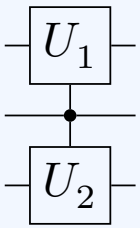


```
      cnot a[1] | a[0];
      cnot a[0] | a[1];
   \end{yquant*}
\end{tikzpicture}
```




```
\begin{tikzpicture}
   \begin{yquant*}
      cnot q[2] | q[0];
      cnot q[0] | q[1-3];
      box {$U$} q[0] | q[2];
   \end{yquant*}
\end{tikzpicture}
```

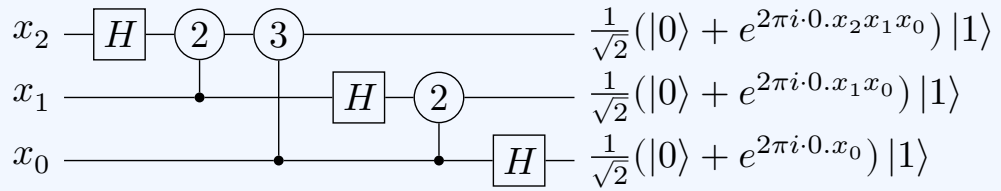
### C. Vertical wires




```
\begin{tikzpicture}
   \begin{yquant*}
      box {$U_{\The\numexpr\idx+1}$} q[0, 2] | q[1];
   \end{yquant*}
\end{tikzpicture}
```

There is no direct support for this construction, but as with the initialization of

### 2.2 Example 2: Quantum Fourier Transform



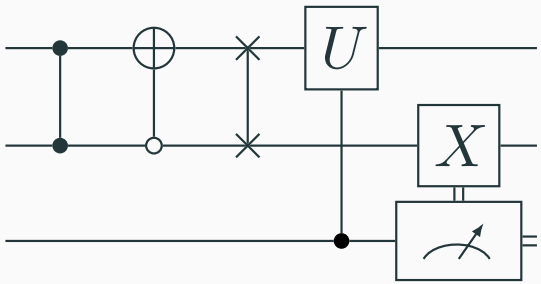

```
\begin{tikzpicture}
   \begin{yquant}[operators/every box/.append style={shape=yquant-circle,
    ↪  radius=2.5mm}]
      qubit {$x_2$} x2;
      qubit {$x_1$} x1;
      qubit {$x_0$} x0;

      h x2;
      box {$2$} x2 | x1;
      box {$3$} x2 | x0;
      h x1;
      box {$2$} x1 | x0;
      h x0;

      output {$\frac{1}{\sqrt2} (\ket0 + e^{2\pi i \cdot 0.x_2 x_1 x_0})
       ↪  \ket1$} x2;
      output {$\frac{1}{\sqrt2} (\ket0 + e^{2\pi i \cdot 0.x_1 x_0})
       ↪  \ket1$} x1;
      output {$\frac{1}{\sqrt2} (\ket0 + e^{2\pi i \cdot 0.x_0}) \ket1$}
       ↪  x0;
   \end{yquant}
\end{tikzpicture}
```

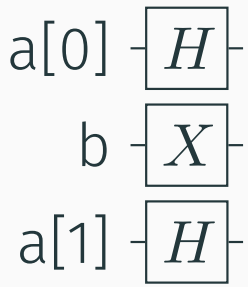
In this example, we opted to use three distinct registers instead of one vector

# yquant (2019 – 2025)




```
\begin{tikzpicture}
   \begin{yquant*}
      zz (q[0, 1]);
      cnot q[0] ~ q[1];
      swap (q[0, 1]);
      box {$U$} q[0] | q[2];
      [direct control] measure q[2];
      x q[1] | q[2];
   \end{yquant*}
\end{tikzpicture}
```

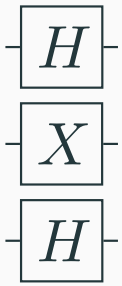
pre- and automatically-created registers (at any time)




```
\begin{yquant}
    qubit a;
    h a;
    qubit b;
    x b;
    qubit a[+1];
    h a[1];
\end{yquant}
```

# yquant • Feature overview

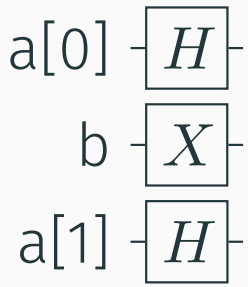
pre- and automatically-created registers (at any time)




```
\begin{yquant*}
    h a;
    x b;
    h a[1];
\end{yquant*}
```

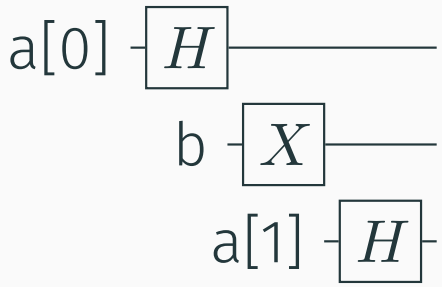
# yquant • Feature overview

pre- and automatically-created registers (at any time)



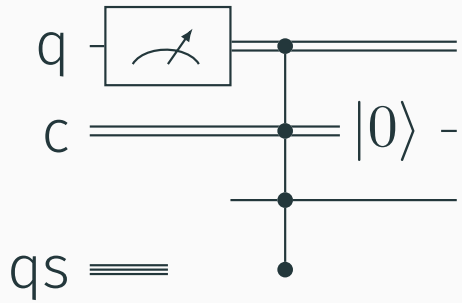

```
\begin{yquant*}[register/default lazy name=\regidx]
    h a;
    x b;
    h a[1];
\end{yquant*}
```

pre- and automatically-created registers (at any time)




```
\begin{yquant}
   qubit a;
   h a;
   [after=a]
   QuBit b;
   X b;
   [after=b] qubit a[+1];
   h a[1];
\end{yquant}
```

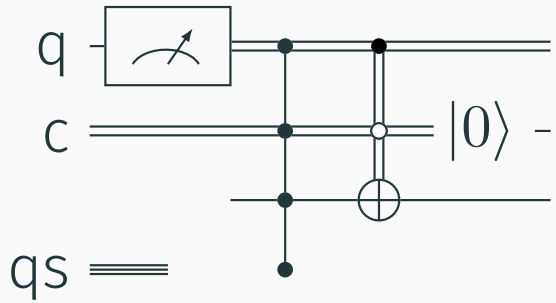
quantum, classical, bundle, empty—change at any time




```
\begin{yquant}
   qubit q;
   cbit c;
   nobit n;
   qubits qs;

   measure q;
   align n, q;
   settype {qubit} n;
   hspace {5mm} qs;
   discard qs;
   zz (-);
   [type=qubit]
   init {$\ket0$} c;
\end{yquant}
```

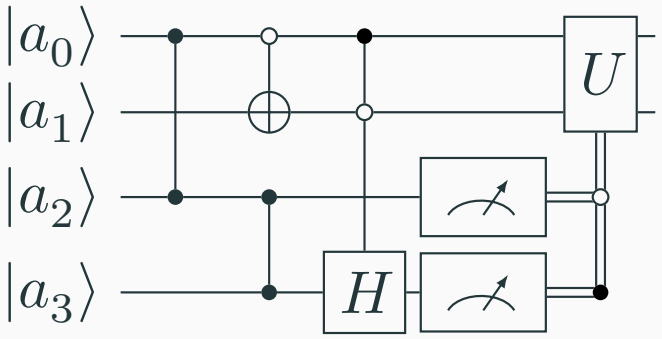
positive and negative controls: set/boolean notation




```
\begin{yquant}
   qubit q;
   cbit c;
   nobit n;
   qubits qs;

   measure q;
   align n, q;
   settype {qubit} n;
   hspace {5mm} qs;
   discard qs;
   zz (-);
   cnot n | q ~ c;
   [type=qubit] init {$\ket0$} c;
\end{yquant}
```

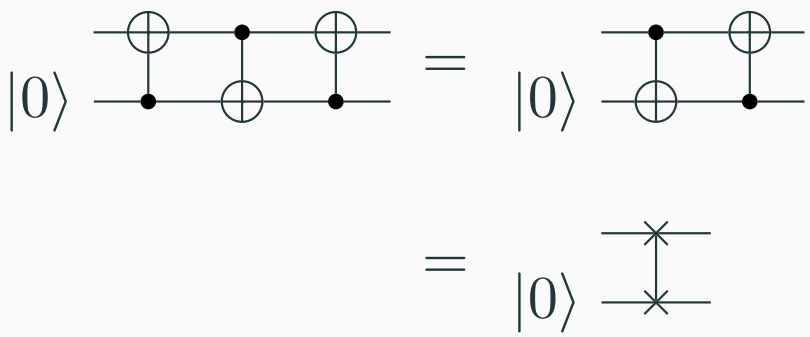
vector registers; single/multi registers; versatile indices



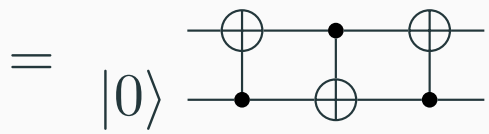

```
\begin{yquant}
   qubit {$\ket{\reg_{\idx}}$} a[4];

   zz (a[0, 2]);
   cnot a[1] ~ a[0];
   zz a[(2, 3)];
   h a[3] | a[0] ~ a[1];
   measure a[2], a[3];
   box {$U$} (a[-1]) | a[3] ~ a[2];
   discard a[2, 3];
\end{yquant}
```

circuit equations

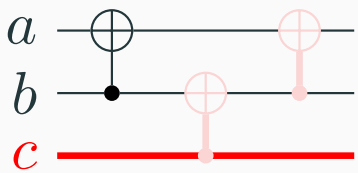


and since equality is transitive,

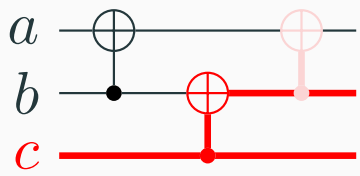


```
\begin{yquantgroup}[group/aligned]
   \registers{
      qubit {} q;
      qubit {$\ket0$} q[+1];
   }
   \circuit{
      cnot q[0] | q[1];
      cnot q[1] | q[0];
      cnot q[0] | q[1];
   }
   \equals*
   \circuit{
      cnot q[1] | q[0];
      cnot q[0] | q[1];
   } \\
   \equals
   \circuit{
      swap (q);
   }
   \intertext{and since equality is
   ↪  transitive,}
   \equals
   \circuit{
      cnot q[0] | q[1];
      cnot q[1] | q[0];
      cnot q[0] | q[1];
   }
\end{yquantgroup}
```

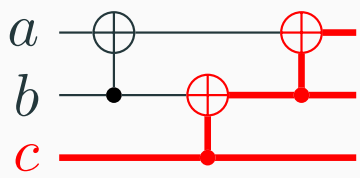
integration with `beamer`



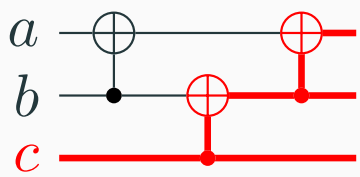
integration with `beamer`



integration with `beamer`

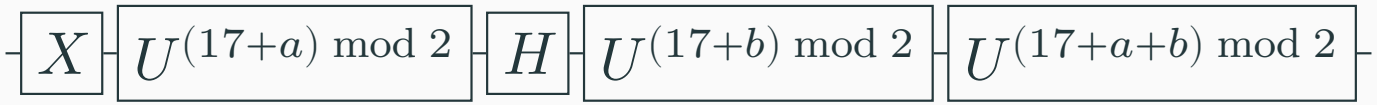


integration with **beamer**; styling via **pgfkeys**




```
\begin{tikzpicture}[on/.style={red, very
↪  thick}]
  \begin{yquant}[on/.style={style=red, control
  ↪  style={very thick}}]
    qubit {$a$} a;
    qubit {$b$} b;
    qubit {\color{red}$c$} c;
    cnot a | b;
    setstyle {on} c;
    \uncover<2->{ [on] cnot b | c; }
    \only<2->{ setstyle {on} b; }
    \uncover<3->{ [on] cnot a | b; }
    \only<3->{ setstyle {on} a; }
  \end{yquant}
\end{tikzpicture}
```

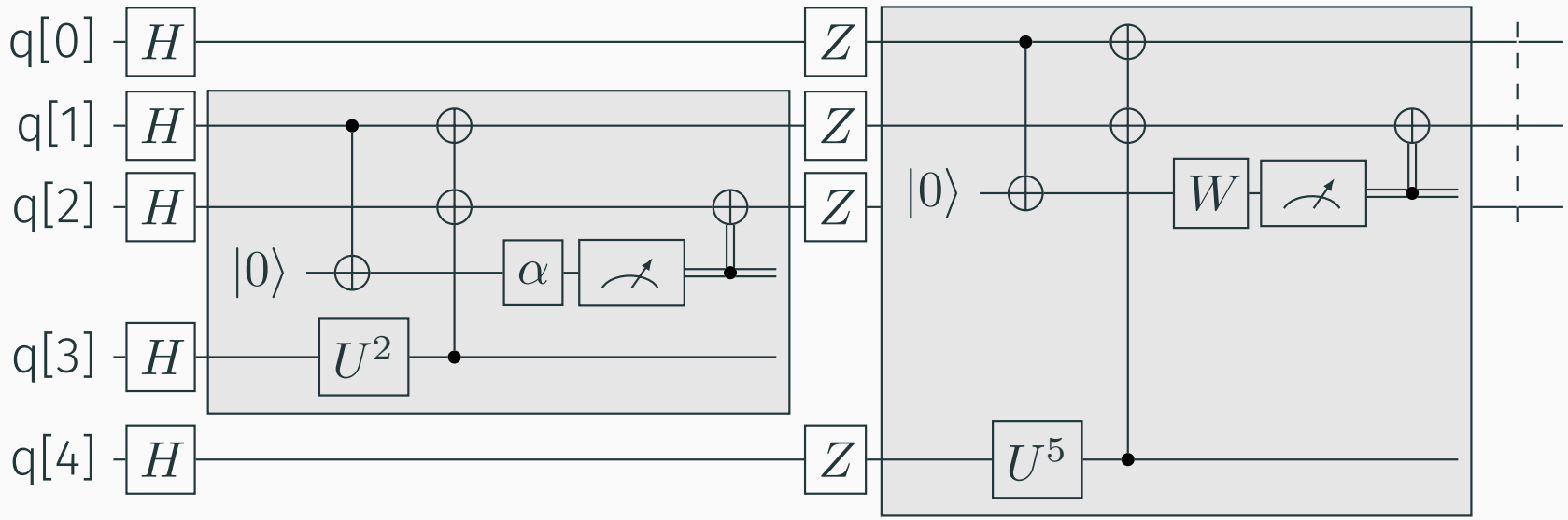
## custom gates




```
% somewhere (probably preamble):
\yquantdefinebox{mygate}{$U^{(17 + #1) \bmod 2}$}

\begin{yquant*}
   x q;
   mygate {a} q;
   h q;
   mygate {b} q;
   mygate {a + b} q;
\end{yquant*}
```

custom gates; subcircuits



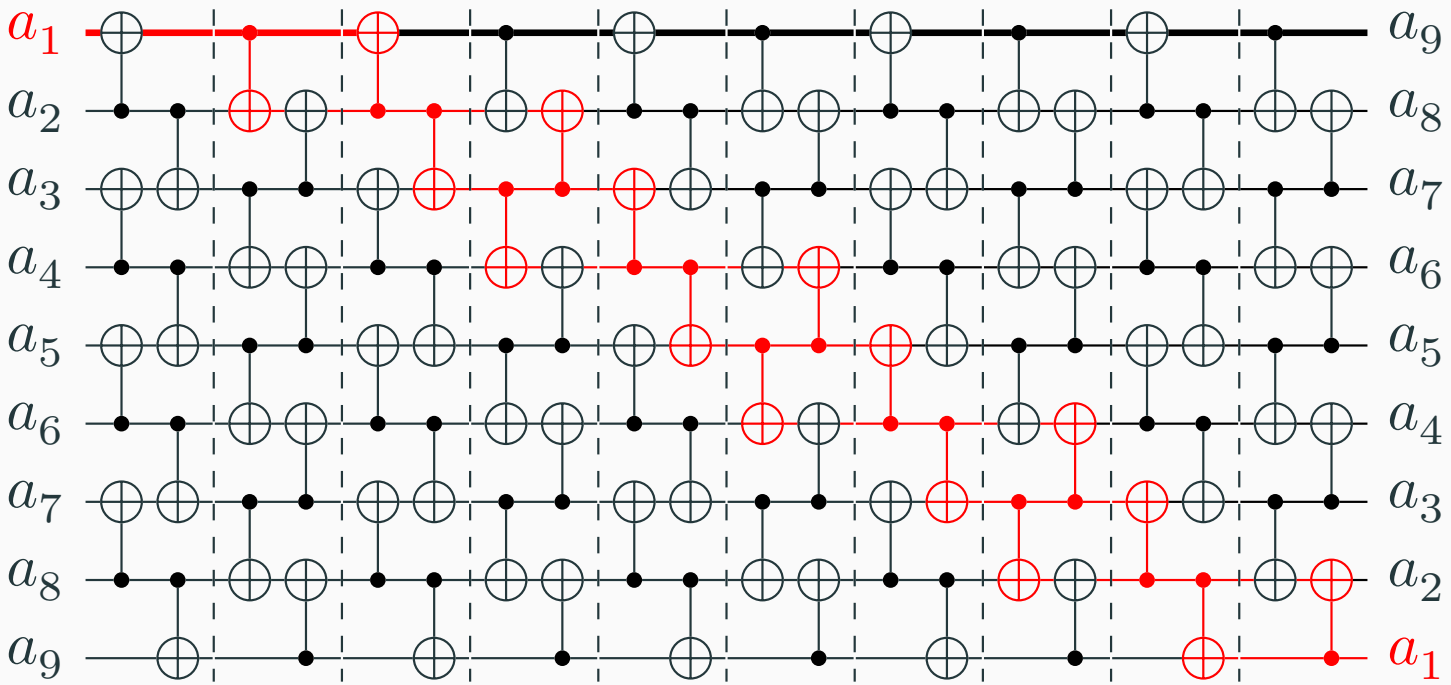

```
\begin{yquant}
   qubit q[5];
   h -;
   longgate {\alpha} (q[1-3]);
   z q[-2, 4];
   [power=5] longgate {W} (q[0, 1, 4]);
   barrier (-q[2]);
\end{yquant}
```

## custom gates; subcircuits

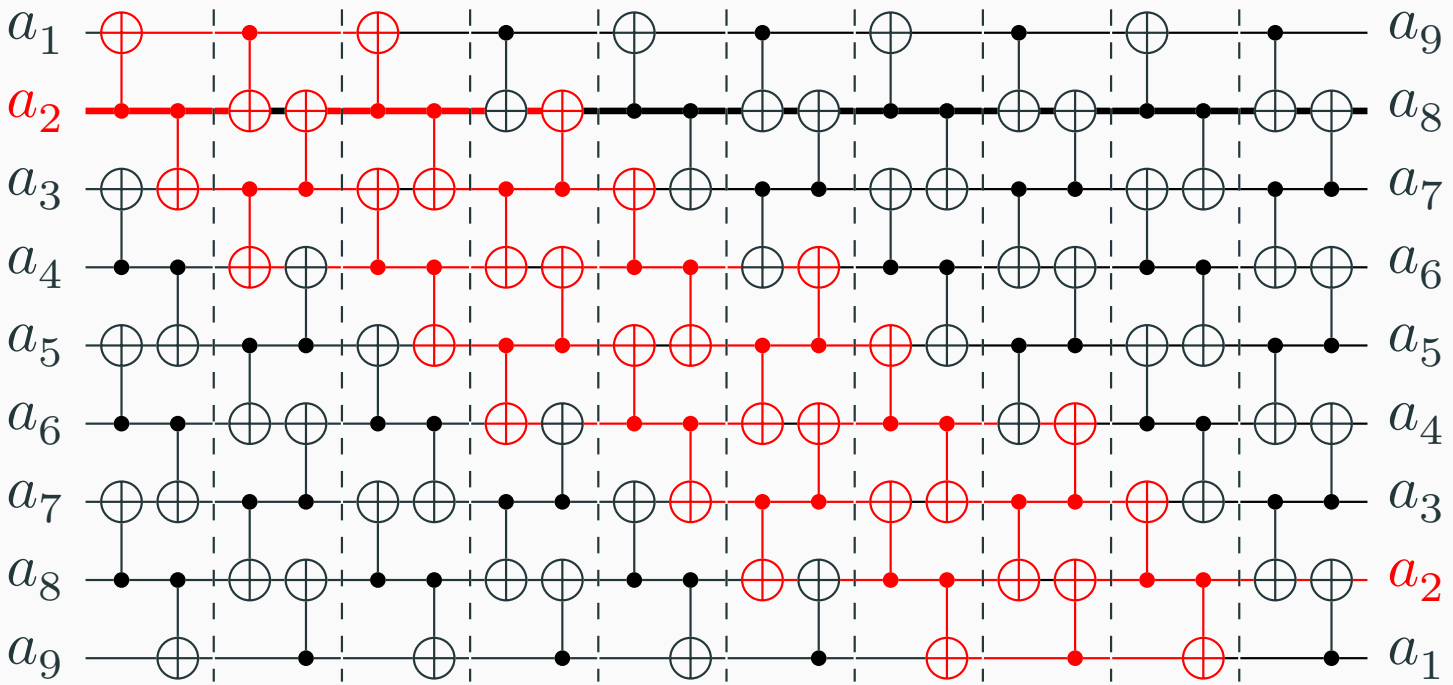
```
% somewhere (probably preamble):
\def\longgatepower{2}
\yquantdeclareattr{power/.store in=\longgatepower}
\yquantdefinegate[optional attrs={power}]{longgate}
                 [/yquant/this subcircuit box style=
                  ↪ {fill=gray!20!white}]{
   qubit {} reg[2];
   [ancilla] qubit {$\ket0$} anc;
   [in] qubit {} cons;

   cnot anc | reg[0];
   box {$U^{\longgatepower}$} cons;
   cnot reg | cons;
   box {$#1$} anc;
   measure anc;
   cnot reg[1] | anc;
}
```

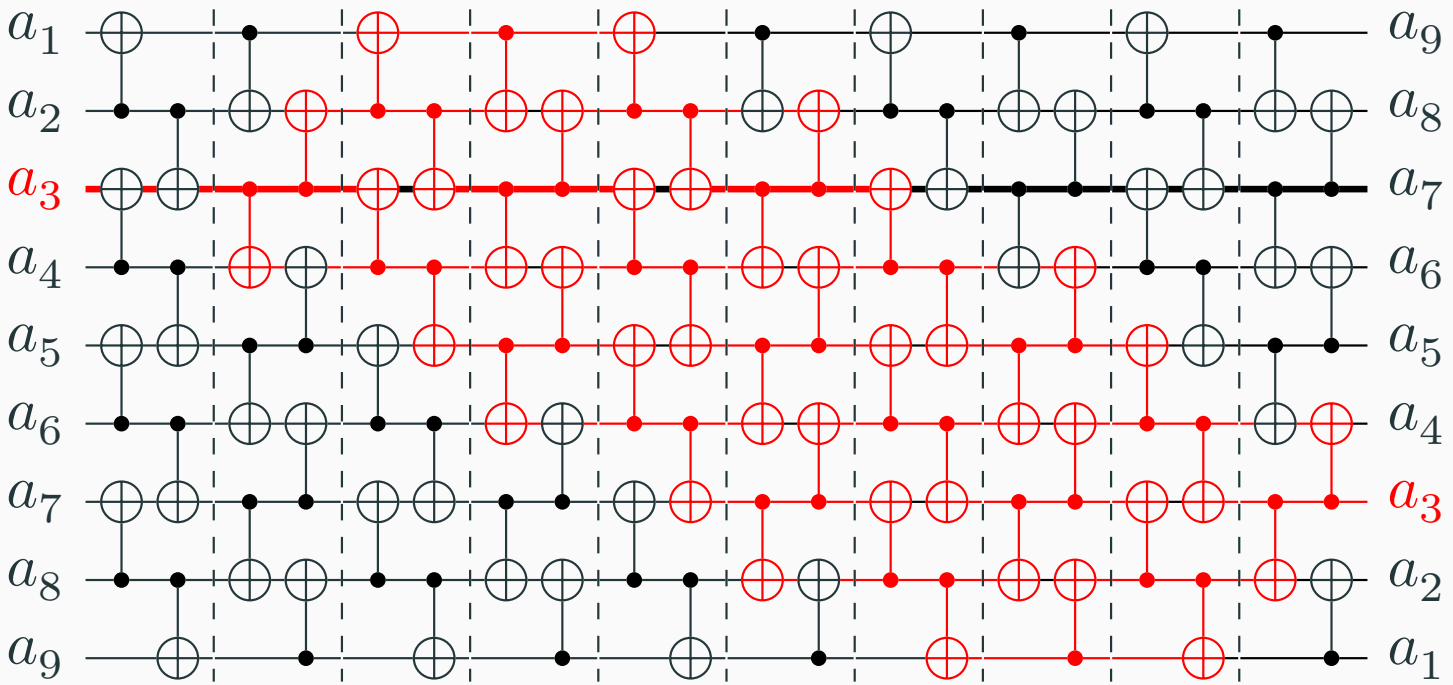
## TEX programming



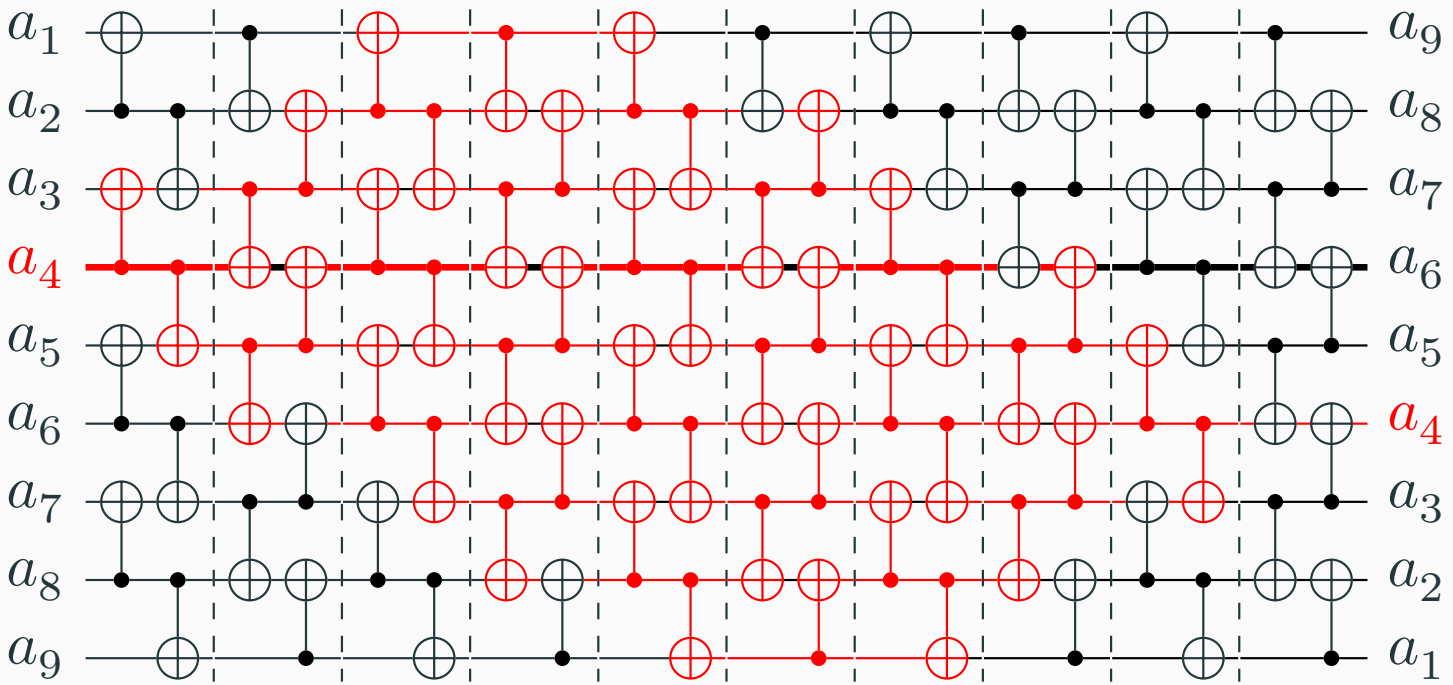
## TEX programming



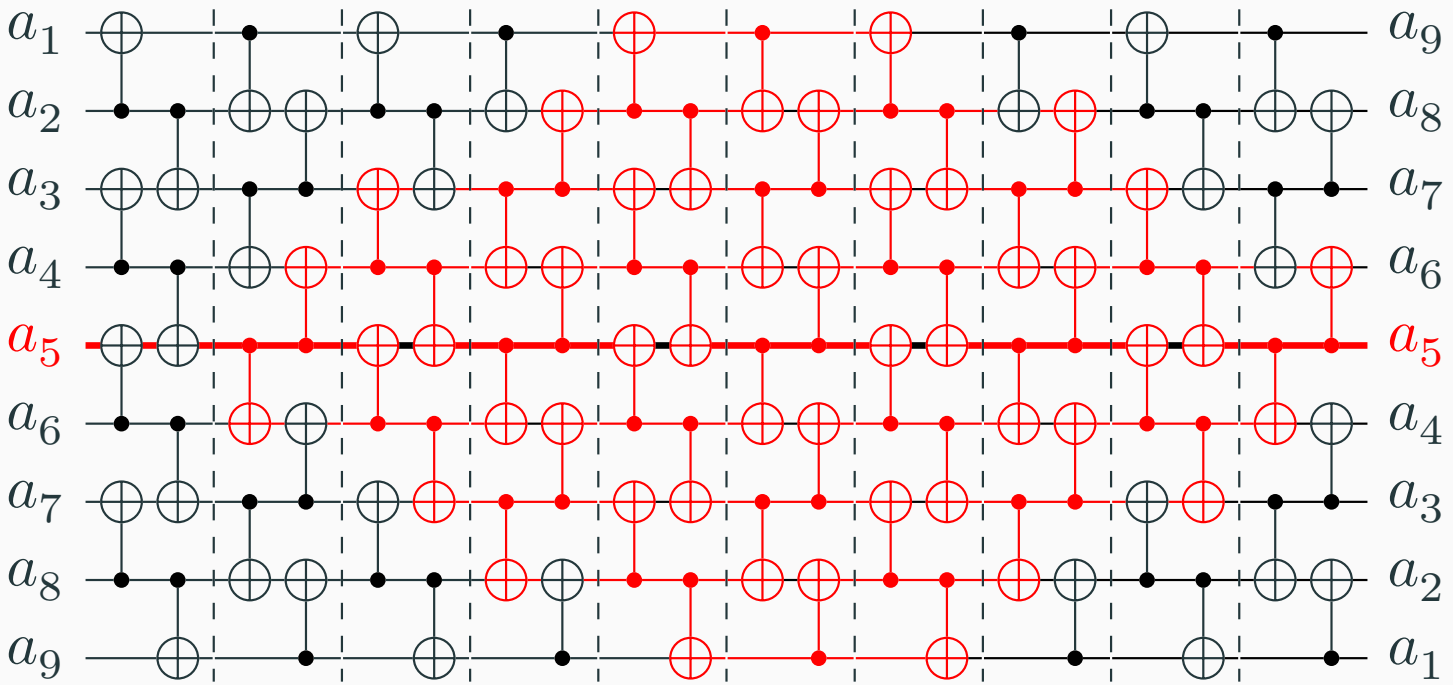
## TEX programming



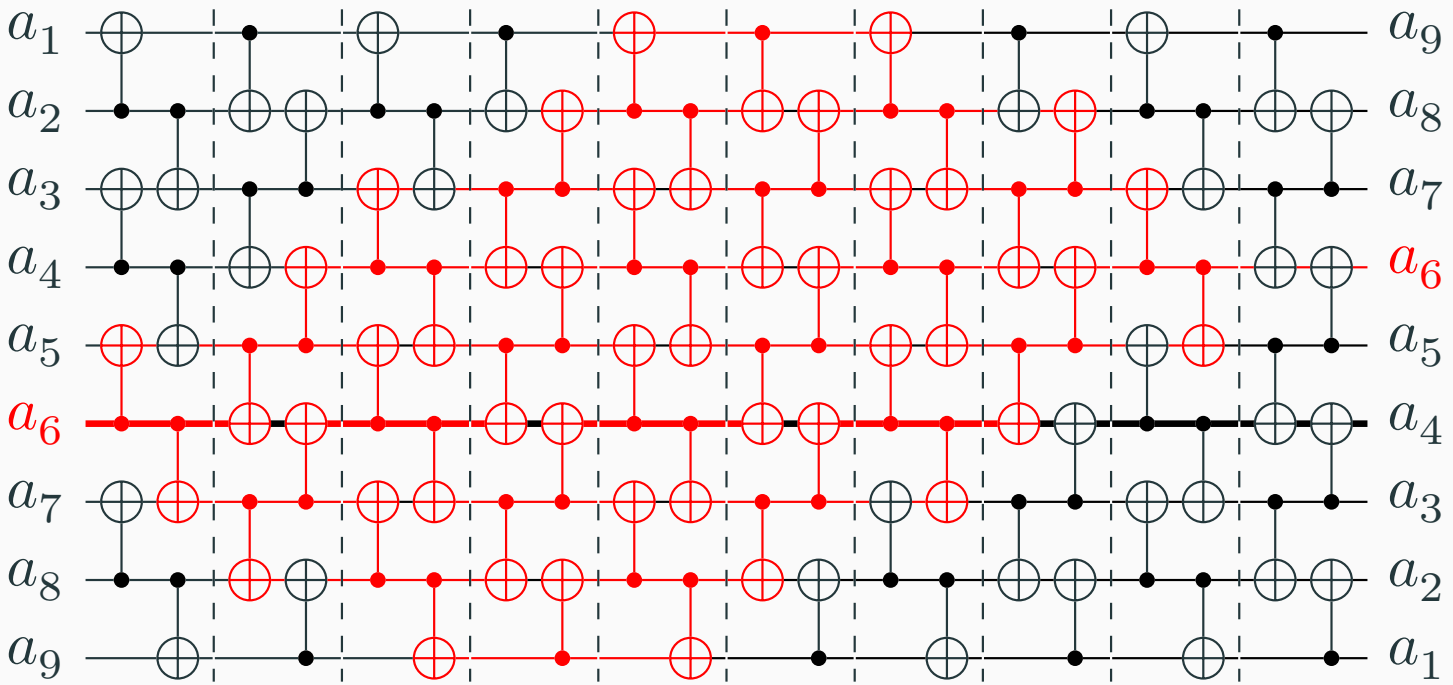
## TEX programming



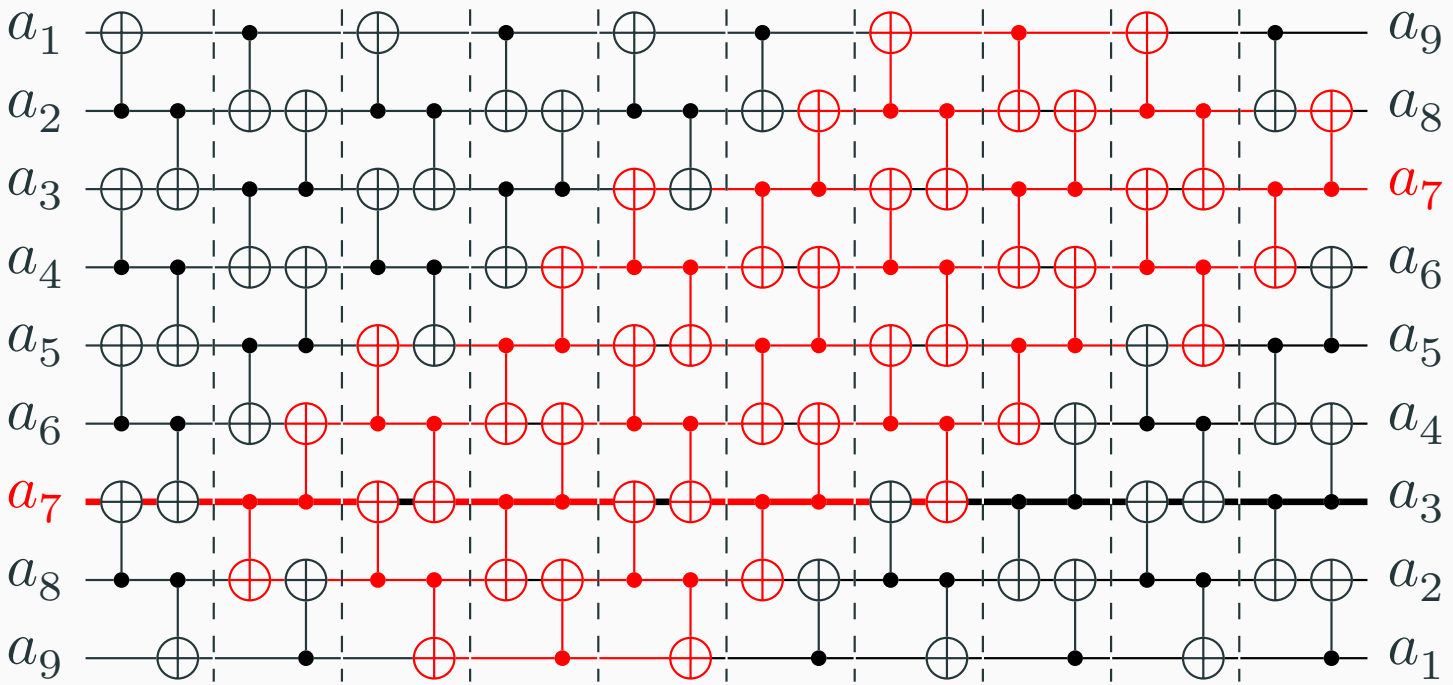
## TEX programming



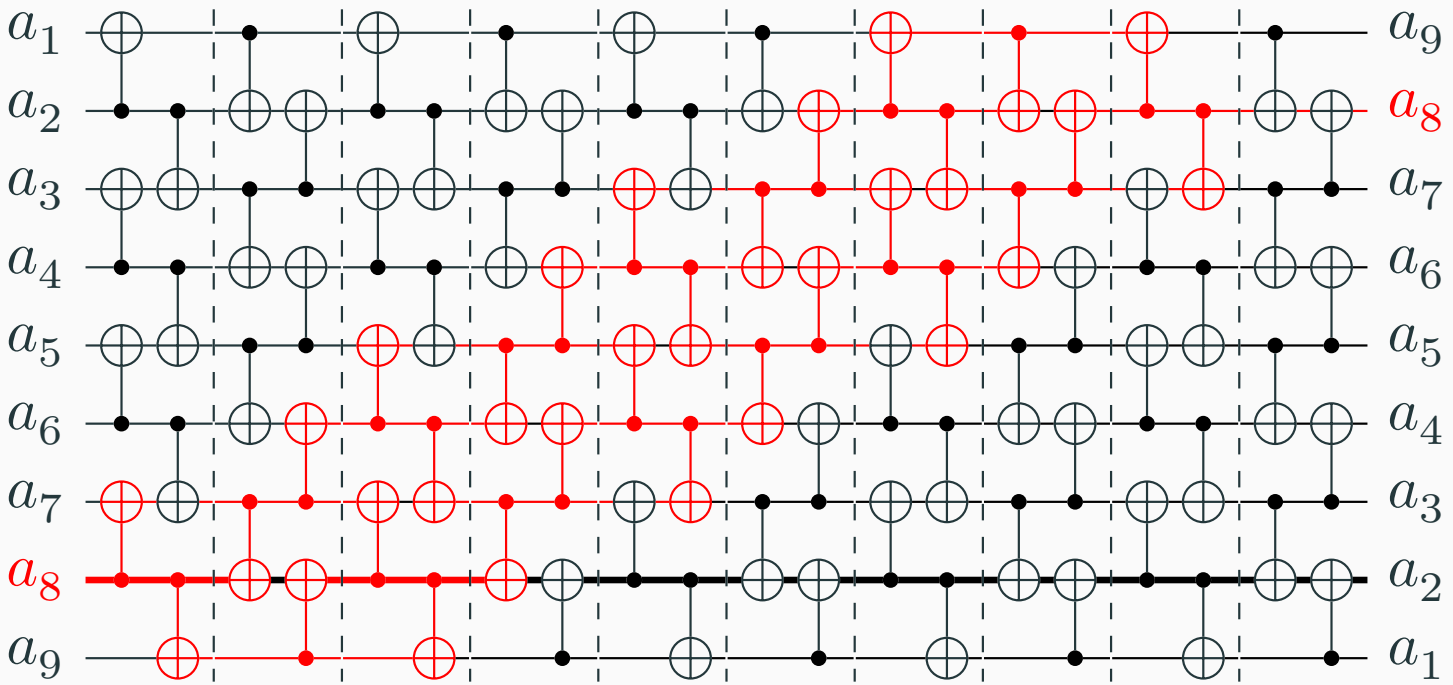
## TEX programming



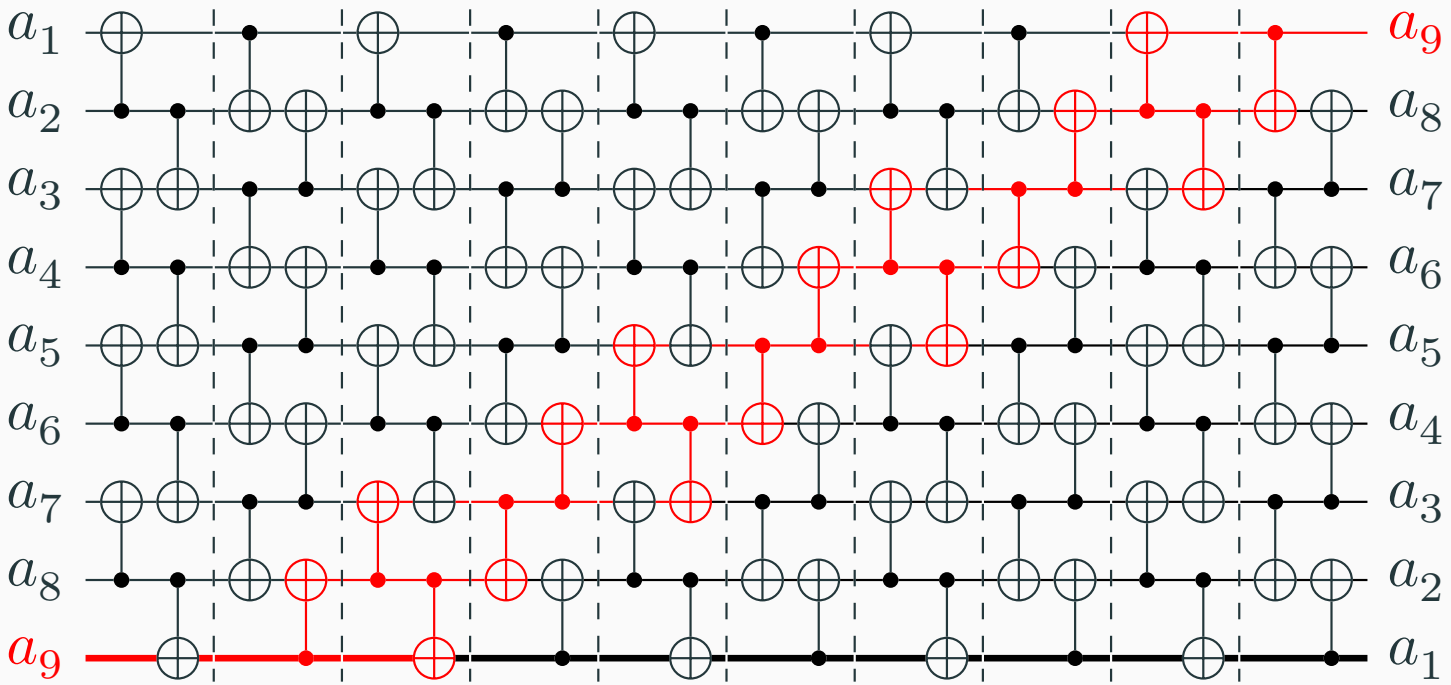
## TEX programming



## TEX programming



## TEX programming



## TEX programming

```
\def\reversecircuit#1{%
   \begin{tikzpicture}
      \let\high=\empty
      \listeadd\high{#1}
      \def\cnot##1|##2;{%
         \ifinlist{##2}\high{
            \yquant [style=red] cnot a[##1] | a[##2];
            \ifinlist{##1}\high{
               \listremove\high{##1}
               \yquant addstyle {black} a[##1];
            }{
               \listadd\high{##1}
               \yquant addstyle {red} a[##1];
            }
         }{
            \yquant cnot a[##1] | a[##2];
         }
      }
      \def\cnotA{\cnot 0|1; \cnot 2|3; \cnot 4|5; \cnot 6|7;}
      \def\cnotB{\cnot 2|1; \cnot 4|3; \cnot 6|5; \cnot 8|7;}
      \def\cnotC{\cnot 1|0; \cnot 3|2; \cnot 5|4; \cnot 7|6;}
      \def\cnotD{\cnot 1|2; \cnot 3|4; \cnot 5|6; \cnot 7|8;}
      \def\cnotBlock{\cnotA \cnotB barrier (-); \cnotC \cnotD}
      % ...
```

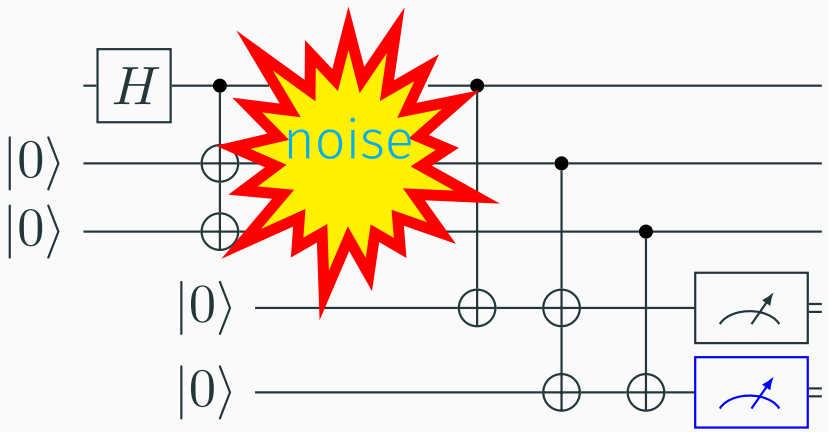
## TEX programming

```
      % ...
      \begin{yquant}[operator/minimum width=0pt, register/minimum height=2mm,
                     register/minimum depth=2mm]
         qubit {\Ifnum\idx=#1\color{red}\Fi$\reg_{\The\numexpr\idx+1}$} a[9];
         addstyle {very thick, red} a[#1];

         \cnotBlock barrier (-);
         \cnotBlock barrier (-);
         \cnotBlock barrier (-);
         \cnotBlock barrier (-);
         \cnotBlock
         output {\protect\xifinlist{\idx}{\high}{\color{red}}\relax
                 $a_{\The\numexpr9-\idx}$} -;
      \end{yquant}
   \end{tikzpicture}%
}

\foreach \i in {1, ..., 9} {%
   \only<\i>{\expandafter\reversecircuit\the\numexpr\i-1\relax}%
}%
```

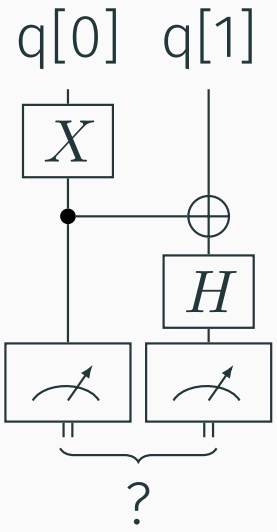
## TikZ integration




```
\begin{tikzpicture}
   \begin{yquant}
      qubit {} data;
      qubit {$\ket0$} anc1[2];

      h data;
      cnot anc1 | data;
      [after=data]
      qubit {$\ket0$} anc2[2];
      [name=noise]
      text {\phantom{noise}} (data,
      ↪  anc1);
      cnot anc2[0] | data;
      cnot anc2 | anc1[0];
      cnot anc2[1] | anc1[1];
      measure anc2[0];
      [blue] measure anc2[1];
   \end{yquant}
   \node[starburst, cyan, fill=yellow,
   ↪  draw=red, line width=2pt,
         inner xsep=-4pt, inner
         ↪  ysep=-5pt, fit=(noise)]
         ↪  {noise};
\end{tikzpicture}
```

vertical layout




```
\begin{yquant}[vertical]
   qubit q[2];
   x q[0];
   cnot q[1] | q[0];
   h q[1];
   measure q;
   output {?} (q);
\end{yquant}
```

# yquant

- LaTeX-only package for typesetting quantum circuits
- input in a human-readable language
- builds on Ti*k*Z
- extensive documentation with countless examples
- available on CTAN since 2019
- `https://github.com/projekter/yquant`