



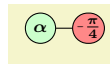
ZX-calculus with TikZ

Version 1.0 October 12, 2021

github.com/leo-colisson/zx

1 Introduction

This library (based on the great `TikZ` and `TikZ-cd` packages) allows you to typeset ZX-calculus directly in \LaTeX . It comes with a default—but customizable—style:



```
\begin{ZX}
  \zxZ{\alpha} \arrow[r] & \zxFracX-{\pi}{4}\\
\end{ZX}
```

The goal is to provide an alternative to the great `tikzit` package: we wanted a solution that does not require the creation of an additional file, the use of an external software, and which automatically adapts the width of columns and rows depending on the content of the nodes (in `tikzit` one needs to manually tune the position of each node, especially when dealing with large nodes). Our library also provides a default style and tries to separate the content from the style: that way it should be easy to globally change the styling of a given project without redesigning all diagrams. However, it should be fairly easy to combine `tikzit` and this library: when some diagrams are easier to design in `tikzit`, then it should be possible to directly load the style of this library inside `tikzit`.

This library is quite young, so feel free to propose improvements or report issues on github.com/leo-colisson/zx. We will of course try to maintain backward compatibility as much as possible, but we can't guarantee at 100% that small changes (spacing...) won't be changed later. The documentation is also a work in progress, so feel free to check the code to discover new functionalities.

2 Installation

If your CTAN distribution is recent enough, you can directly insert in your file:

```
\usepackage{zx}
```

or load `TikZ` and then use:

```
\usetikzlibrary{cd}
```

If this library is not yet packaged into CTAN (which is very likely in 2021), you must first download `tikzlibraryzx.code.tex`¹ and `zx.sty`² and copy them at the root of your project.

3 Usage

3.1 Add a diagram


```
\zx[options]{your diagram}
\begin{ZX}[options]
  environment contents
\end{ZX}
```

¹<https://github.com/leo-colisson/zx>

²<https://github.com/leo-colisson/zx>

`\end{ZX}`

You can create a new ZX-diagram either with a command (quicker for inline diagrams) or with an environment. The *options* can be used to locally change the style of the diagram, using the same options as the `{tikz-cd}` environment (from the `tikz-cd` package³). The *your diagram* argument, or the content of `{ZX}` environment is a TikZ matrix of nodes, exactly like in the `tikz-cd` package: columns are separated using `&`, columns using `\\`, and nodes are created using `|[tikz style]| node content` or with shortcut commands presented later in this document (recommended). Content is typeset in math mode by default, and diagrams can be included in any equation. Wires can be added like in `tikz-cd` (see more below) using `\arrow` or `\ar`: we provide later recommended styles to quickly create different kinds of wires which can change with the configured style.

Spider , equation $\circ = \bullet$ and custom diagram:



```
\Spider \zx{\zxZ{\alpha}}, equation $\zx{\zxZ{}} = \zx{\zxX{}}$ %
and custom diagram: %
\begin{ZX}[red]
  \zxZ{\beta} \arrow[r] & \zxZ{\alpha} \\
  |[fill=pink,draw]| \gamma \arrow[ru,bend right]
\end{ZX}
```

3.2 Nodes

The following commands are useful to create different kinds of nodes. Always add empty arguments like `\example{}` if none are already present, otherwise if you type `\example` we don't guarantee backward compatibility.

`\zxEmptyDiagram`

Create an empty diagram.

```
\begin{ZX}
  \zxEmptyDiagram{}
\end{ZX}
```

`\zxNone+-{<text>}`

Adds an empty node with `\zxNone{}`. Combine it with wires and the `wire centered` style for straight lines or C-like wires (alias `wc`, or you will get holes) or `between none` style (alias `bn`) for other curved lines. Moreover, you should also add column and row spacing `&[\zxWCol]` and `\\[\zxWRow]` to avoid too shrunked diagrams when only wires are involved. The `-|+` decorations are used to add a bit of (respectively) horizontal (`\zxNone-{}|`), vertical (`\zxNone|{}+`) and both (`\zxNone+{}|+`) spacing (I don't know how to add `|` in the documentation of the function).

```
\begin{ZX}
  \zxNone-{}| \ar[C,d,wc] \ar[rd,s,bn] &[\zxWCol] \zxNone-{}|\\[\zxWRow]
  \zxNone-{}| \ar[ru,s,bn] & \zxNone-{}|
\end{ZX}
```

`\zxNoneDouble+-{<text>}`

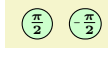
Like `\zxNone`, but the spacing for `+-|` is large enough to fake two lines in only one. Not extremely useful (or one needs to play with `start anchor=south,end anchor=north`).

```
\begin{ZX}
  \zxNoneDouble|{} \ar[r,s,start anchor=north,end
  anchor=south,l=1.2] \ar[r,s,start anchor=south,end
  anchor=north,l=1.2] &[\zxWCol] \zxNoneDouble|{}
\end{ZX}
```

³<https://www.ctan.org/pkg/tikz-cd>

`\zxFracZ-{\langle numerator \rangle}{\langle denominator \rangle}`

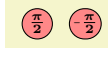
Adds a Z node with a fraction, use the minus decorator to add a small minus in front (the default minus is very big).



```
\begin{ZX}
  \zxFracZ{\pi}{2} & \zxFracZ-{\pi}{2}
\end{ZX}
```

`\zxFracX-{\langle numerator \rangle}{\langle denominator \rangle}`

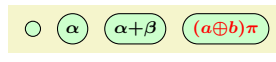
Adds an X node with a fraction.



```
\begin{ZX}
  \zxFracX{\pi}{2} & \zxFracX-{\pi}{2}
\end{ZX}
```

`\zxZ[\langle other styles \rangle]{\langle text \rangle}`

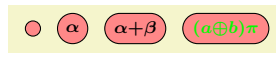
Adds a Z node.



```
\begin{ZX}
  \zxZ{} & \zxZ{\alpha} & \zxZ{\alpha + \beta} & \zxZ[text=red]{(a \oplus b)\pi}
\end{ZX}
```

`\zxX[\langle other styles \rangle]{\langle text \rangle}`

Adds an X node.



```
\begin{ZX}
  \zxX{} & \zxX{\alpha} & \zxX{\alpha + \beta} & \zxX[text=green]{(a \oplus b)\pi}
\end{ZX}
```

`\zxH[\langle other styles \rangle]`

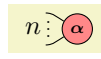
Adds an Hadamard node. See also H wire style.



```
\begin{ZX}
  \zxNone{} \rar & \zxH{} \rar & \zxNone{}
\end{ZX}
```

`\leftManyDots[\langle text scale \rangle][\langle dots scale \rangle]{\langle text \rangle}`

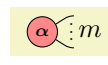
Shortcut to add a dots and a text next to it. It automatically adds the new column, see more examples below.



```
\begin{ZX}
  \leftManyDots{n} \zxX{\alpha}
\end{ZX}
```

`\leftManyDots[\langle text scale \rangle][\langle dots scale \rangle]{\langle text \rangle}`

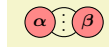
Shortcut to add a dots and a text next to it. It automatically adds the new column, see more examples below.



```
\begin{ZX}
  \zxX{\alpha} \rightManyDots{m}
\end{ZX}
```

`\middleManyDots`

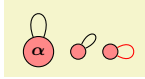
Shortcut to add a dots and a text next to it. It automatically adds the new column, see more examples below.



```
\begin{ZX}
\zxX{\alpha} \middleManyDots{} & \zxX{\beta}
\end{ZX}
```

`\zxLoop[⟨direction angle⟩][⟨opening angle⟩][⟨other styles⟩]`

Adds a loop in $\langle \text{direction angle} \rangle$ (defaults to 90), with opening angle $\langle \text{opening angle} \rangle$ (defaults to 20).



```
\begin{ZX}
\zxX{\alpha} \zxLoop{} & \zxX{} \zxLoop[45]{} & \zxX{} \zxLoop[0][30][red]{}
\end{ZX}
```

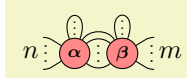
`\zxLoopAboveDots[⟨opening angle⟩][⟨other styles⟩]`

Adds a loop above the node with some dots.



```
\begin{ZX}
\zxX{\alpha} \zxLoopAboveDots{}
\end{ZX}
```

The previous commands can be useful to create this figure:



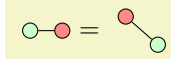
```
% Forces code/example on two lines.
\begin{ZX}
\leftManyDots{n} \zxX{\alpha} \zxLoopAboveDots{} \middleManyDots{} \ar[r,o'=60]
& \zxX{\beta} \zxLoopAboveDots{} \rightManyDots{m}
\end{ZX}
```

3.3 Wires

`\arrow[⟨options⟩]`

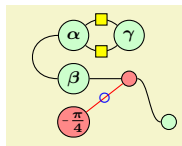
`\ar[⟨options⟩]`

These synonym commands (actually coming from `tikz-cd`) are used to draw wires between nodes. We refer to `tikz-cd` for an in-depth documentation, but what is important for our purpose is that the direction of the wires can be specified in the $\langle \text{options} \rangle$ using a string of letters `r` (right), `l` (left), `u` (up), `d` (down).



```
\zx{\zxZ{} \ar[r] & \zxX{}} = \zx{\zxX{} \arrow[rd] \& \zxZ{}}
```

$\langle \text{options} \rangle$ can also be used to add any additional style, either custom ones, or the ones defined in this library (this is recommended since it can be easily changed document-wise by simply changing the style). Multiple wires can be added in the same cell. Other shortcuts provided in `tikz-cd` like `\rar...` can be used.



```
\begin{ZX}
\zxZ{\alpha} \arrow[d, C] % C = Bell-like wire
\ar[r,H,o'] % o' = top part of circle
% H adds Hadamard, combine with \zxHCol
\ar[r,H,o.] & [\zxHCol] \zxZ{\gamma} \&
\zxZ{\beta} \rar & \zxX{} \ar[ld,red,"circ"
{marking,blue}] \ar[rd,s] \&
\zxFracX{-\pi}{4} & & \& \zxZ{}
\end{ZX}
```

As explained in `tikz-cd`, there are further shortened forms:

```
\rar[⟨options⟩]
\lar[⟨options⟩]
\dar[⟨options⟩]
\uar[⟨options⟩]
\drar[⟨options⟩]
\urar[⟨options⟩]
\dlar[⟨options⟩]
\ular[⟨options⟩]
```

The first one is equivalent to

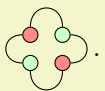
```
\arrow[⟨options⟩]{r}
```

and the other ones work analogously.

We give now a list of wire styles provided in this library (`/zx/wires definition/` is an automatically loaded style). We recommend using them instead of manual styling to ensure they are the same document-wise, but they can of course be customized to your need. Note that the name of the styles are supposed to graphically represent the action of the style, and some characters are added to precise the shape: typically `'` means top, `.` bottom, `X-` is right to `X` (or should arrive with angle 0), `-X` is left to `X` (or should leave with angle zero). These shapes are usually designed to work when the starting node is left most (or above of both nodes have the same column). But they may work both way for some of them.

```
/zx/wires definition/C (style, no value)
/zx/wires definition/C. (style, no value)
/zx/wires definition/C' (style, no value)
/zx/wires definition/C- (style, no value)
```

Bell-like wires with an arrival at “right angle”, `C` represents the shape of the wire, while `.` (bottom), `'` (top) and `-` (side) represent (visually) its position. Combine with `wire centered` (`wc`) to avoid holes when connecting multiple wires.

Bell pair  and funny graph

```
Bell pair \zx{\zxNone{} \ar[d,C,wc] \l[\zxWRow]
\zxNone{}}
and funny graph
\begin{ZX}
\zxX{} \ar[d,C] \ar[r,C'] & \zxZ{} \ar[d,C-] \l
\zxZ{} \ar[r,C.] & \zxX{}
\end{ZX}.
```

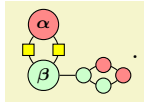
```
/zx/wires definition/o'=angle (style, default 40)
/zx/wires definition/o.=angle (style, default 40)
/zx/wires definition/o-=angle (style, default 40)
/zx/wires definition/-o=angle (style, default 40)
```

Curved wire, similar to `C` but with a soften angle (optionally specified via `⟨angle⟩`, and globally editable with `\zxDefaultLineWidth`). Again, the symbols specify which part of the circle (represented with `o`) must be kept.



```
\begin{ZX}
\zxX{} \ar[d,-o] \ar[d,o-] \l
\zxZ{} \ar[r,o'] \ar[r,o.] & \zxX{}
\end{ZX}.
```

Note that these wires can be combined with `H`, `X` or `Z`, in that case one should use appropriate column and row spacing as explained in their documentation:



```
\begin{ZX}
\zxX{\alpha} \ar[d,-o,H] \ar[d,o-,H] \\\[\\zxHRow]
\zxZ{\beta} \rar & \zxZ{} \ar[r,o',X] \ar[r,o.,Z] & \zxSCol \ \zxX{}
\end{ZX}.
```

`/zx/wires definition/(` (style, no value)
`/zx/wires definition/)` (style, no value)

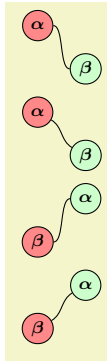
Curved wire, similar to `o` but can be used for diagonal items. The `(` and `)` symbols must be imagined as if the starting point was on top of the parens, and the ending point at the bottom.



```
\begin{ZX}
\zxX{} \ar[rd,[] \ar[rd,],red] \\\
& \zxZ{}
\end{ZX}.
```

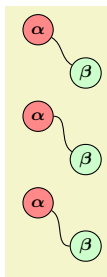
`/zx/wires definition/s` (style, no value)
`/zx/wires definition/s'=angle` (style, default 30)
`/zx/wires definition/s.=angle` (style, default 30)
`/zx/wires definition/-s'=angle` (style, default 30)
`/zx/wires definition/-s.=angle` (style, default 30)
`/zx/wires definition/s'=-angle` (style, default 30)
`/zx/wires definition/s.-=angle` (style, default 30)

`s` is used to create a `s`-like wire, to have nicer soften diagonal lines between nodes. Other versions are soften versions (the input and output angles are not as sharp, and the difference angle can be configured as an argument or globally using `\zxDefaultSoftAngleS`). Adding `'` or `.` specifies if the wire is going up-right or down-right.



```
\begin{ZX}
\zxX{\alpha} \ar[s,rd] \\\
& \zxZ{\beta} \\\
\zxX{\alpha} \ar[s.,rd] \\\
& \zxZ{\beta} \\\
& \zxZ{\alpha} \\\
\zxX{\beta} \ar[s,ru] \\\
& \zxZ{\alpha} \\\
\zxX{\beta} \ar[s',ru] \\\
\end{ZX}
```

`-` forces the angle on the side of `-` to be horizontal.

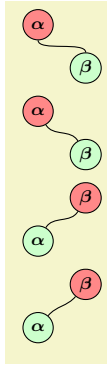


```
\begin{ZX}
\zxX{\alpha} \ar[s.,rd] \\\
& \zxZ{\beta} \\\
\zxX{\alpha} \ar[-s.,rd] \\\
& \zxZ{\beta} \\\
\zxX{\alpha} \ar[s.-,rd] \\\
& \zxZ{\beta} \\\
\end{ZX}
```

`/zx/wires definition/ss` (style, no value)
`/zx/wires definition/ss.=angle` (style, default 30)
`/zx/wires definition/.ss=angle` (style, default 30)

`/zx/wires definition/sIs.=angle` (style, default 30)
`/zx/wires definition/.sIs=angle` (style, default 30)
`/zx/wires definition/ss.I-=angle` (style, default 30)
`/zx/wires definition/I.ss-=angle` (style, default 30)

`ss` is similar to `s` except that we go from top to bottom instead of from left to right. The position of `.` says if the node is wire is going bottom right (`ss.`) or bottom left (`.ss`).

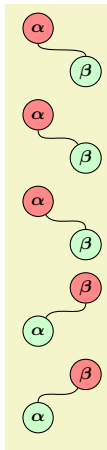


```

\begin{ZX}
  \zxX{\alpha} \ar[ss,rd] \\\
                                & \zxZ{\beta} \\\
  \zxX{\alpha} \ar[ss.,rd] \\\
                                & \zxZ{\beta} \\\
                                & \zxX{\beta} \ar[.ss,d1] \\\
  \zxZ{\alpha} \\\
                                & \zxX{\beta} \ar[.ss=40,d1] \\\
  \zxZ{\alpha} \\\
\end{ZX}

```

`I` forces the angle above (if in between the two `s`) or below (if on the same side as `.`) to be vertical.



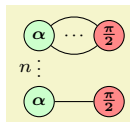
```

\begin{ZX}
  \zxX{\alpha} \ar[ss,rd] \\\
                                & \zxZ{\beta} \\\
  \zxX{\alpha} \ar[sIs.,rd] \\\
                                & \zxZ{\beta} \\\
  \zxX{\alpha} \ar[ss.I,rd] \\\
                                & \zxZ{\beta} \\\
                                & \zxX{\beta} \ar[.sIs,d1] \\\
  \zxZ{\alpha} \\\
                                & \zxX{\beta} \ar[I.ss,d1] \\\
  \zxZ{\alpha} \\\
\end{ZX}

```

`/zx/wires definition/3 dots=text` (style, default =)
`/zx/wires definition/3 vdots=text` (style, default =)

The styles put in the middle of the wire (without drawing the wire) `...` (for `3 dots`) or `\vdots` (for `3 vdots`). The dots are scaled according to `\zxScaleDots` and the text `\langle text \rangle` is written on the left. Use `&[\zxDotsRow]` and `\\[\zxDotsRow]` to properly adapt the spacing of columns and rows.



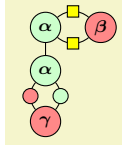
```

\begin{ZX}
  \zxZ{\alpha} \ar[r,o'] \ar[r,o.]
                                \ar[r,3 dots]
                                \ar[d,3 vdots={\ell n \ell},] & [\zxDotsCol] \zxFracX{\pi}{2} \\ [\zxDotsRow]
  \zxZ{\alpha} \rar                                & \zxFracX{\pi}{2}
\end{ZX}

```

`/zx/wires definition/H` (style, no value)
`/zx/wires definition/Z` (style, no value)
`/zx/wires definition/X` (style, no value)

Adds a H (Hadamard), Z or X node (without phase) in the middle of the wire. Width of column or rows should be adapted accordingly using `\zxNameRowcolFlatnot` where `Name` is replaced by H, S (for “spiders”, i.e. X or Z), HS (for both H and S) or W, `Rowcol` is either `Row` (for changing row sep) or `Col` (for changing column sep) and `Flatnot` is empty or `Flat` (if the wire is supposed to be a straight line as it requires more space). For instance:



```
\begin{ZX}
\zxZ{\alpha} \ar[d] \ar[r,o',H] \ar[r,o.,H] &[\zxHCol] \zxX{\beta} \\
\zxZ{\alpha} \ar[d,-o,X] \ar[d,o-,Z] & \\
\zxX{\gamma}
\end{ZX}
```

```
/zx/wires definition/wire centered (style, no value)
/zx/wires definition/wc (style, no value)
/zx/wires definition/wire centered start (style, no value)
/zx/wires definition/wcs (style, no value)
/zx/wires definition/wire centered end (style, no value)
/zx/wires definition/wce (style, no value)
```

When the wires are drawn, they start from the border of the node. However, it can make strange results if nodes do not have the same size, or if we connect none nodes (we will get holes). It is therefore possible to force the wire to start at the center of the node (`wire centered start`, alias `wcs`), to end at the center of the node (`wire centered end`, alias `wce`) or both (`wire centered`, alias `wc`). See also `between node` to also increase looseness when connecting only wires.



```
\begin{ZX}
\zxZ{} \ar[o',r] \ar[o.,r] & \zxX{\alpha} \\
\zxZ{} \ar[o',r,wc] \ar[o.,r,wc] & \zxX{\alpha}
\end{ZX}
```

Without `wc` (note that because there is no node, we need to use `&[\zxWCol]` (for columns) and `\\[\zxWRow]` (for rows) to get nicer spacing):



```
\zx{\zxNone{}} \rar &[\zxWCol] \zxNone{} \rar &[\zxWCol] \zxNone{} }
```

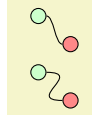
With `wc`:



```
\zx{\zxNone{}} \rar[wc] &[\zxWCol] \zxNone{} \rar[wc] &[\zxWCol] \zxNone{}}
```

```
/zx/wires definition/l=looseness (style, no default)
```

Shortcut for `looseness`, allows to quickly redefine looseness. Use with care (or redefine style directly). Note that you can also change yourself other values, like `in`, `out`...



```
\begin{ZX}
\zxZ{} \ar[rd,s] \\
& \zxX{} \\
\zxZ{} \ar[rd,s,l=3] \\
& \zxX{}
\end{ZX}
```

```
/zx/wires definition/between none (style, no value)
/zx/wires definition/bn (style, no value)
```

When drawing wires only, the default looseness may not be good looking and holes may appear in the line. This style (whose alias is `bn`) should therefore be used when curved wires (except `C` which already has a good looseness, use `wire centered` instead) are connected

together. In that case, also make sure to separate columns using `&[\zxWCol]` and row using `\\[\zxWRow]`.

A swapped Bell pair is 

```
\begin{ZX}
\zxNone{} \ar[C,d,wc] \ar[rd,s,bn] &[\zxWCol] \zxNone{} \\[\zxWRow]
\zxNone{} \ar[ru,s,bn] & \zxNone{}
\end{ZX}
```

4 Advanced styling

It is possible to arbitrarily customize the styling, create new styles... First, any option that can be given to a `tikz-cd` matrix can also be given to a `ZX` environment (we refer to the manual of `tikz-cd` for more details). Moreover, the user can also change the current style or add new style. While you should not use these styles directly (`\zxX` and `\zxFracX` will choose automatically the good style for you), you may want to use these styles to improve existing styles, or to create a `tikzit` style (make sure to load this library then in `*.tikzdefs`). We also encourage the advanced users too look at the code of the library.

/zx/default style nodes (style, no value)

This is where the default style must be loaded. By default, it simply loads the (nested) style packed with this library, `/zx/styles/rounded style`. You can change the style here if you would like to globally change a style.

/zx/user overlay nodes (style, no value)

If a user just wants to overlay some parts of the node styles, add your changes here.



```
{\tikzset{
  /zx/user overlay nodes/.style={
    zxH/.append style={dashed,inner sep=2mm}
  }
  \zx{\zxNone{} \rar & \zxH{} \rar & \zxNone{}}
}
```

You can also change it on a per-diagram basis:



```
\zx[text=yellow,/zx/user overlay nodes/.style={
  zxSpiders/.append style={thick,draw=purple}}
]{\zxX{} \rar & \zxX{\alpha} \rar & \zxFracZ-{\pi}{2}}
```

The list of keys that can be changed will be given below in `/zx/styles/rounded style/*`.

/zx/default style wires (style, no value)

Default style for wires. Note that `/zx/wires definition/` is always loaded by default, and we don't add any other style for wires by default. But additional styles may use this functionality.

/zx/user overlay wires (style, no value)

The user can add here additional styles for wires.



```
\begin{ZX}[/zx/user overlay wires/.style={thick,->,C/.append
style={dashed}}]
\zxNone{} \ar[d,C] \rar[] &[\zxWCol] \zxNone{} \\[\zxWRow]
\zxNone{} \rar[] & \zxNone{}
\end{ZX}
```

/zx/styles/rounded style (style, no value)

This is the style loaded by default. It contains internally other (nested) styles that must be defined for any custom style.

We present now all the properties that a new node style must have (and that can be overlaid as explained above).

`/zx/styles/rounded style/zxAllNodes` (style, no value)

Style applied to all nodes.

`/zx/styles/rounded style/zxEmptyDiagram` (style, no value)

Style to draw an empty diagram.

`/zx/styles/rounded style/zxNone` (style, no value)

`/zx/styles/rounded style/zxNone+` (style, no value)

`/zx/styles/rounded style/zxNone-` (style, no value)

`/zx/styles/rounded style/zxNoneI` (style, no value)

Styles for None wires (no inner sep, useful to connect to wires). The -,I,+ have additional horizontal, vertical, both spaces.

`/zx/styles/rounded style/zxNoneDouble` (style, no value)

`/zx/styles/rounded style/zxNoneDouble+` (style, no value)

`/zx/styles/rounded style/zxNoneDouble-` (style, no value)

`/zx/styles/rounded style/zxNoneDoubleI` (style, no value)

Like `zxNone`, but with more space to fake two nodes on a single line (not very used).

`/zx/styles/rounded style/zxSpiders` (style, no value)

Style that apply to all circle spiders.

`/zx/styles/rounded style/zxNoPhase` (style, no value)

Style that apply to spiders without any angle inside. Used by `\zxX{}` when the argument is empty.

`/zx/styles/rounded style/zxNoPhaseSmall` (style, no value)

Like `zxNoPhase` but for spiders drawn in between wires.

`/zx/styles/rounded style/zxShort` (style, no value)

Spider with text but no inner space. Used notably to obtain nice fractions.

`/zx/styles/rounded style/zxLong` (style, no value)

Spider with potentially large text. Used by `\zxX{\alpha}` when the argument is not empty.

`/zx/styles/rounded style/zxNoPhaseZ` (style, no value)

`/zx/styles/rounded style/zxNoPhaseX` (style, no value)

`/zx/styles/rounded style/zxNoPhaseSmallZ` (style, no value)

`/zx/styles/rounded style/zxNoPhaseSmallX` (style, no value)

`/zx/styles/rounded style/zxShortZ` (style, no value)

`/zx/styles/rounded style/zxShortX` (style, no value)

`/zx/styles/rounded style/zxLongZ` (style, no value)

`/zx/styles/rounded style/zxLongX` (style, no value)

Like above styles, but with colors of X and Z spider added. The color can be changed globally by updating the `colorZxX` color. By default we use:

```
\definecolor{colorZxZ}{RGB}{204,255,204}
\definecolor{colorZxX}{RGB}{255,136,136}
\definecolor{colorZxH}{RGB}{255,255,0}
```

as the second recommendation in zxcalculus.com/accessibility.html.

`/zx/styles/rounded style/zxH` (style, no value)

Style for Hadamard spiders, used by `\zxH{}` and uses the color `colorZxH`.

`/zx/styles/rounded style/zxHSmall`

(style, no value)

Like `zxH` but for Hadamard on wires, (see `H` style).

We also define several spacing commands that can be redefined to your needs:

`\zxHCol`
`\zxHRow`
`\zxHColFlat`
`\zxHRowFlat`
`\zxSCol`
`\zxSRow`
`\zxSColFlat`
`\zxSRowFlat`
`\zxHSCol`
`\zxHSRow`
`\zxHSColFlat`
`\zxHSRowFlat`
`\zxWCol`
`\zxWRow`
`\zxDotsCol`
`\zxDotsRow`

These are spaces, to use like `&[\zxHCol]` or `\\[\zxHRow]` in order to increase the default spacing of rows and columns depending on the style of the wire. `H` stands for Hadamard, `S` for Spiders, `W` for Wires only, `HS` for both Spiders and Hadamard, `Dots` for the 3 dots styles. And of course `Col` for columns, `Row` for rows.

`\zxDefaultSoftAngleS`
`\zxDefaultSoftAngleO`

Default opening angles of `S` and `o` wires. Defaults to respectively 30 and 40.

`\zxMinus`

The minus sign used in fractions.

5 Acknowledgement

I'm very grateful of course to everybody that worked on these amazing field which made diagramatic quantum computing possible, and to the many StackExchange users (sorry, I don't want to risk an incomplete list) that helped me to understand a bit more \LaTeX and \TikZ . I also thank Robert Booth for making me realize how my old style was bad, and for giving advices on how to improve it. Thanks to John van de Wetering, whose style has also been a source of inspiration.

Index

This index only contains automatically generated entries. A good index should also contain carefully selected keywords. This index is not a good index.

- (key, 6
-) key, 6
- o key, 5
- s' key, 6
- s. key, 6
- .sIs key, 7
- .ss key, 6
- 3 dots key, 7
- 3 vdots key, 7

- \ar, 4
- \arrow, 4

- between none key, 8
- bn key, 8

- C key, 5
- C' key, 5
- C- key, 5
- C. key, 5
- cd library, 1

- \dar, 5
- default style nodes key, 9
- default style wires key, 9
- \dlar, 5
- \drar, 5

- Environments
 - ZX, 2

- H key, 7

- I.ss- key, 7

- l key, 8
- \lar, 5
- \leftManyDots, 3
- Libraries
 - cd, 1

- \middleManyDots, 4

- o' key, 5
- o- key, 5
- o. key, 5

- Packages and files
 - tikz-cd, 1

- \rar, 5
- rounded style key, 9
- rounded style/zxAllNodes key, 10
- rounded style/zxEmptyDiagram key, 10
- rounded style/zxH key, 10
- rounded style/zxHSmall key, 11
- rounded style/zxLong key, 10
- rounded style/zxLongX key, 10
- rounded style/zxLongZ key, 10
- rounded style/zxNone key, 10
- rounded style/zxNone+ key, 10
- rounded style/zxNone- key, 10
- rounded style/zxNoneDouble key, 10
- rounded style/zxNoneDouble+ key, 10
- rounded style/zxNoneDouble- key, 10
- rounded style/zxNoneDoubleI key, 10
- rounded style/zxNoneI key, 10
- rounded style/zxNoPhase key, 10
- rounded style/zxNoPhaseSmall key, 10
- rounded style/zxNoPhaseSmallX key, 10
- rounded style/zxNoPhaseSmallZ key, 10
- rounded style/zxNoPhaseX key, 10
- rounded style/zxNoPhaseZ key, 10
- rounded style/zxShort key, 10
- rounded style/zxShortX key, 10
- rounded style/zxShortZ key, 10
- rounded style/zxSpiders key, 10

- s key, 6
- s' key, 6
- s'- key, 6
- s. key, 6
- s.- key, 6
- sIs. key, 7
- ss key, 6
- ss. key, 6
- ss.I- key, 7

- \uar, 5
- \ular, 5
- \urar, 5
- user overlay nodes key, 9
- user overlay wires key, 9

- wc key, 8
- wce key, 8
- wcs key, 8
- wire centered key, 8
- wire centered end key, 8
- wire centered start key, 8

- X key, 7

- Z key, 7
- ZX environment, 2
- \zx, 1
- zx package, 1
- /zx/
 - default style nodes, 9
 - default style wires, 9
 - styles/

rounded style, 9	wcs, 8
rounded style/zxAllNodes, 10	wire centered, 8
rounded style/zxEmptyDiagram, 10	wire centered end, 8
rounded style/zxH, 10	wire centered start, 8
rounded style/zxHSmall, 11	X, 7
rounded style/zxLong, 10	Z, 7
rounded style/zxLongX, 10	\zxDefaultSoftAngle0, 11
rounded style/zxLongZ, 10	\zxDefaultSoftAngleS, 11
rounded style/zxNone, 10	\zxDotsCol, 11
rounded style/zxNone+, 10	\zxDotsRow, 11
rounded style/zxNone-, 10	\zxEmptyDiagram, 2
rounded style/zxNoneDouble, 10	\zxFracX, 3
rounded style/zxNoneDouble+, 10	\zxFracZ, 3
rounded style/zxNoneDouble-, 10	\zxH, 3
rounded style/zxNoneDoubleI, 10	\zxHCol, 11
rounded style/zxNoneI, 10	\zxHColFlat, 11
rounded style/zxNoPhase, 10	\zxHRow, 11
rounded style/zxNoPhaseSmall, 10	\zxHRowFlat, 11
rounded style/zxNoPhaseSmallX, 10	\zxHSCol, 11
rounded style/zxNoPhaseSmallZ, 10	\zxHSColFlat, 11
rounded style/zxNoPhaseX, 10	\zxHSRow, 11
rounded style/zxNoPhaseZ, 10	\zxHSRowFlat, 11
rounded style/zxShort, 10	\zxLoop, 4
rounded style/zxShortX, 10	\zxLoopAboveDots, 4
rounded style/zxShortZ, 10	\zxMinus, 11
rounded style/zxSpiders, 10	\zxNone, 2
user overlay nodes, 9	\zxNoneDouble, 2
user overlay wires, 9	\zxSCol, 11
wires definition/	\zxSColFlat, 11
(, 6	\zxSRow, 11
), 6	\zxSRowFlat, 11
-o, 5	\zxWCol, 11
-s', 6	\zxWRow, 11
-s., 6	\zxX, 3
.sIs, 7	\zxZ, 3
.ss, 6	
3 dots, 7	
3 vdots, 7	
between none, 8	
bn, 8	
C, 5	
C', 5	
C-, 5	
C., 5	
H, 7	
I.ss-, 7	
l, 8	
o', 5	
o-, 5	
o., 5	
s, 6	
s', 6	
s'-, 6	
s., 6	
s.-, 6	
sIs., 7	
ss, 6	
ss., 6	
ss.I-, 7	
wc, 8	
wce, 8	