$$\{\mathbf{ZX}\}$$

ZX-calculus with Ti*k*Z

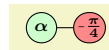Version 1.0   October 12, 2021
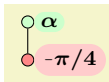
github.com/leo-colisson/zx-calculus

Contents

1 Introduction

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

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

Even if this has not yet been tested a lot, you can also use a "phase in label" style, without really changing the code:

```
\begin{ZX}[phase in label right]
    \zxZ{\alpha} \arrow[d] \\
    \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-calculus`. 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 made later. In case you want a completely unalterable style, just copy this library in your project forever! 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-calculus.code.tex`[1] and `zx-calculus.sty`[2] (right-click on "Raw" and "Save link as") and save 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}`

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` package[3]). 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 \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.

---

[1] `https://github.com/leo-colisson/zx-calculus/blob/main/tikzlibraryzx-calculus.code.tex`
[2] `https://github.com/leo-colisson/zx-calculus/blob/main/zx-calculus.sty`
[3] `https://www.ctan.org/pkg/tikz-cd`

\zxEmptyDiagram

Create an empty diagram.

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

\zxNone+-{⟨*text*⟩}
\zxN+-{⟨*text*⟩}

Adds an empty node with \zxNone{} (alias \zxN{}). 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 shrinked 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}
```
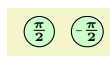
\zxNoneDouble+-{⟨*text*⟩}

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

\zxFracZ-{⟨*numerator*⟩}[⟨*numerator with parens*⟩][⟨*denominator with parens*⟩]{⟨*denominator*⟩}

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}
```

The optional arguments are useful when the numerator or the denominator need parens when they are written inline (in that case optional arguments must be specified): it will prove useful when using a style that writes the fraction inline, for instance the default style for labels:

Compare $\left(\frac{a+b}{c+d}\right)$ with ○ $(a + b)/(c + d)$

```
Compare
\begin{ZX}
   \zxFracZ{a+b}[(a+b)][(c+d)]{c+d}
\end{ZX} with %
\begin{ZX}[phase in label right]
   \zxFracZ{a+b}[(a+b)][(c+d)]{c+d}
\end{ZX}
```

\zxFracX-{⟨*numerator*⟩}{⟨*denominator*⟩}

Adds an X node with a fraction.

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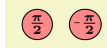
```
\begin{ZX}
    \zxFracX{\pi}{2} & \zxFracX-{\pi}{2}
\end{ZX}
```

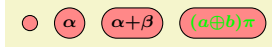**\zxZ**[⟨*other styles*⟩]{⟨*text*⟩}

Adds a Z node.

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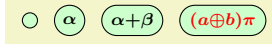
```
\begin{ZX}
    \zxZ{} & \zxZ{\alpha} & \zxZ{\alpha + \beta} & \zxZ[text=red]{(a \oplus b)\pi}
\end{ZX}
```

\zxX[⟨*other styles*⟩]{⟨*text*⟩}

Adds an X node.

● (α) (α+β) ((a⊕b)π)

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

\zxH[⟨*other styles*⟩]

Adds an Hadamard node. See also `H` wire style.

—▫—

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

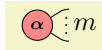**\leftManyDots**[⟨*text scale*⟩][⟨*dots scale*⟩]{⟨*text*⟩}

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

n ⋮ (α)

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

\leftManyDots[⟨*text scale*⟩][⟨*dots scale*⟩]{⟨*text*⟩}

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

(α) ⋮ m

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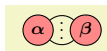
\middleManyDots

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

(α) ⋮ (β)

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

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

Adds a loop in ⟨*direction angle*⟩ (defaults to 90), with opening angle ⟨*opening angle*⟩ (defaults to 20).

```
\begin{ZX}
    \zxX{\alpha} \zxLoop{} & \zxX{} \zxLoop[45]{} & \zxX{} \zxLoop[0][30][red]{}
\end{ZX}
```
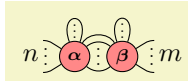
\zxLoopAboveDots[⟨*opening angle*⟩][⟨*other styles*⟩]

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   Phase in label style

We also provide styles to place the phase on a label next to an empty node (not yet very well tested):

/zx/styles rounded style/phase in content	(style, no value)
/zx/styles rounded style/phase in label=style	(style, default)
/zx/styles rounded style/pil=style	(style, default)
/zx/styles rounded style/phase in label above=style	(style, default)
/zx/styles rounded style/pila=style	(style, default)
/zx/styles rounded style/phase in label below=style	(style, default)
/zx/styles rounded style/pilb=style	(style, default)
/zx/styles rounded style/phase in label right=style	(style, default)
/zx/styles rounded style/pilr=style	(style, default)
/zx/styles rounded style/phase in label left=style	(style, default)
/zx/styles rounded style/pill=style	(style, default)

The above styles are useful to place a spider phase in a label outside the node. They can either be put on the style of a node to modify a single node at a time:
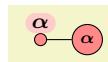
```
\zx{\zxX[phase in label]{\alpha} \rar & \zxX{\alpha}}
```
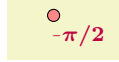
It can also be configured on a per-figure basis:
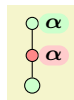
```
\zx[phase in label right]{
    \zxZ{\alpha} \dar \\
    \zxX{\alpha} \dar \\
    \zxZ{}}
```

or globally:

```
{
  \tikzset{
    /zx/user post preparation labels/.style={
      phase in label={label position=-45, text=purple,fill=none}
    }
  }
  \zx{
    \zxFracX-{\pi}{2}
  }
}
```

Note that we must use `user post preparation labels` and not `/zx/user overlay nodes` because this will be run after all the machinery for labels has been setup.

While `phase in content` forces the content of the node to be inside the node instead of inside a label (which is the default behavior), all other styles are special cases of `phase in label`. The ⟨*style*⟩ parameter can be any style made for a tikz label:



```
\zx{
  \zxX[phase in label={label position=45, text=purple}]{\alpha}
}
```

For ease of use, the special cases of label position `above`, `below`, `right` and `left` have their respective shortcut style. The `pil*` versions are shortcuts of the longer style written above. For instance, `pilb` stands for `phase in label below`. Note also that by default labels will take some space, but it's possible to make them overlay without taking space using the `overlay` label style... however do it at your own risks as it can overlay the content around (also the text before and after):
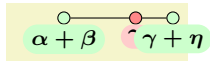


```
\zx{
  \zxZ[pilb]{\alpha+\beta} \rar & \zxX[pilb]{\gamma} \rar & \zxZ[pilb=overlay]{\gamma+\eta}
}
```

The above also works for fractions:
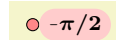
  `\zx{\zxFracX[pilr]-{\pi}{2}}`

For fractions, you can configure how you want the label text to be displayed, either in a single line (default) or on two lines, like in nodes. The function `\zxConvertToFracInLabel` is in charge of that conversion, and can be changed to your needs to change this option document-wise. To use the same notation in both content and labels, you can do:
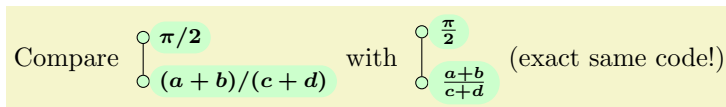

```
  Compare
  \begin{ZX}[phase in label right]
    \zxFracZ{\pi}{2} \dar \\
    \zxFracZ{a+b}[(a+b)][(c+d)]{c+d}
  \end{ZX} with
{\RenewExpandableDocumentCommand{\zxConvertToFracInLabel}{mmmmm}{
    \zxConvertToFracInContent{#1}{#2}{#3}{#4}{#5}%
  }
  \begin{ZX}[phase in label right]
    \zxFracZ{\pi}{2} \dar \\
    \zxFracZ{a+b}[(a+b)][(c+d)]{c+d}
  \end{ZX} (exact same code!)
}
```

Note that in `\zxFracZ{a+b}[(a+b)][(c+d)]{c+d}` the optional arguments are useful to put parens appropriately when the fraction is written inline.

6

Shortcut to add an `alias` to a wire. Will be practical later when using names instead of direction to connect wires in order to improve readability. This is added automatically in `/tikz/every node` style.

```
\begin{ZX}
  \zxX[alias=A]{} & \zxZ[a=B]{\beta}
  \ar[from=A,to=B]
\end{ZX}
```

3.4 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 ⟨*options*⟩ using a string of letters `r` (right), `l` (left), `u` (up), `d` (down). It's also possible to specify a node alias as a source or destination as shown below.
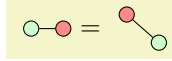
```
\zx{\zxZ{} \ar[r] & \zxX{}} = \zx{\zxX{} \arrow[rd] \\ & \zxZ{}}
```

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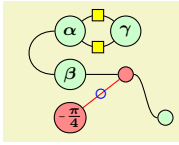
```
\begin{ZX}
  \zxZ{\alpha} \arrow[d, C] % C = Bell-like wire
               \ar[r,H,o'] % o' = top part of circle
               % H adds Hadamard, combine with \zxHCol
               \ar[r,H,o.] &[\zxHCol] \zxZ{\gamma}\\
  \zxZ{\beta}  \rar        & \zxX{} \ar[ld,red,"\circ" {marking,blue}] \ar[rd,s]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

Note that sometimes, it may be practical to properly organize big diagrams to increase read-ability. To that end, one surely wants to have a small and well indented matrix (emacs `M-x align` is very practical to indent matrices automatically). Unfortunately, adding wires inside the matrix can make the line really long and hard to read. Similarly, some nodes involving fractions or long expressions can also be quite long. It is however easy to increase readability (and maintainability) by moving the wires at the very end of the diagram, using `alias` (shortcut `a`) to connect nodes and `\def` to create shortcuts. Putting inside a macro with `\def` long node definitions can also be useful to keep small items in the matrix:
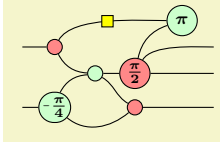

```
\begin{ZX}[row sep=1pt,
  execute at begin picture={%
    %%% Definition of long items (the goal is to have a small and readable matrix
    % (warning: macro can't have numbers in TeX)
    \def\Zpifour{\zxFracZ[a=Zpi4]-{\pi}{4}}%
    \def\mypitwo{\zxFracX[a=mypi2]{\pi}{2}}%
  }
  ]
  %%% Matrix: in emacs "M-x align" is practical to automatically format it. a is for 'alias'
                 &                 &                         &                     & \zxZ[a=toprightpi]{\pi} \\
  \zxN[a=in1]{} & \zxX[a=X1]{} &                         &                     & & \zxN[a=out1]{}         \\
                 &                 & \zxZ[a=xmiddle]{} & \mypitwo{}       & & \zxN[a=out2]{}         \\
  \zxN[a=in2]{} & \Zpifour{}   &                         & \zxX[a=Xdown]{} & & \zxN[a=out3]{}
  %%% Arrows
  % Column 1
  \ar[from=in1,to=X1]
  \ar[from=in2,to=Zpi4]
  % Column 2
  \ar[from=X1,to=xmiddle,(.]
  \ar[from=X1,to=toprightpi,<',H]
  \ar[from=Zpi4,to=xmiddle,(']
  \ar[from=Zpi4,to=Xdown,o.]
  % Column 3
  \ar[from=xmiddle,to=Xdown,s.]
  \ar[from=xmiddle,to=mypi2]
  % Column 4
  \ar[from=mypi2,to=toprightpi,(']
  \ar[from=mypi2,to=out1,<']
  \ar[from=mypi2,to=out2]
  \ar[from=Xdown,to=out3]
\end{ZX}
```

We give now a list of wire styles provided in this library (`/zx/wires definition/` is an auto-matically 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] \\[\zxWRow]
                \zxNone{}}
and funny graph
\begin{ZX}
    \zxX{} \ar[d,C] \ar[r,C'] & \zxZ{} \ar[d,C-]\\
    \zxZ{} \ar[r,C.]          & \zxX{}
\end{ZX}.
```
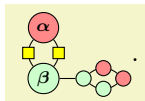
/zx/wires definition/o'=angle   (style, default 40)
/zx/wires definition/o.=angle   (style, default 40)
/zx/wires definition/o-=angle   (style, default 40)
/zx/wires definition/-o=angle   (style, default 40)

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-]\\
    \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/(=angle   (style, default 30)
/zx/wires definition/)=angle   (style, default 30)
/zx/wires definition/('=angle   (style, default 30)
/zx/wires definition/('=angle   (style, default 30)

Curved wire, similar to o but can be used for diagonal items. The angle is, like in `bend right`, the opening angle from the line which links the two nodes. For the first two commands, 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}.
```

Then, ('=(and (.=); this notation is, I think, more intuitive when linking nodes from left to right. (' is used when going to top right and (. when going to bottom right.
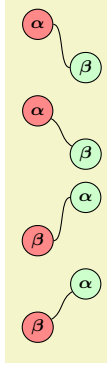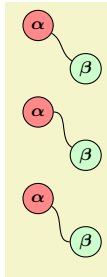


```
\begin{ZX}
    \zxN{}                        & \zxX{}\\
    \zxZ{} \ar[ru,('] \ar[rd,(.] & \\
                                  & \zxX{}
\end{ZX}
```

When the nodes are too far appart, the default angle of 30 may produce strange results as it will go above (for (') the vertical line. Either choose a shorter angle, or see <' instead.

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

`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}
```

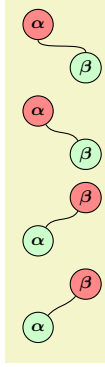`-` forces the angle on the side of `-` to be horizontal.

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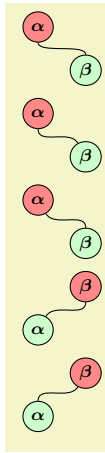
`ss` is similar to `s` except that we go from top to bottom instead of from left to right. The position of `.` says if the node is wire is going bottom right (`ss.`) or bottom left (`.ss`).

```
\begin{ZX}
  \zxX{\alpha} \ar[ss,rd] \\
                            & \zxZ{\beta}\\
  \zxX{\alpha} \ar[ss.,rd] \\
                            & \zxZ{\beta}\\
                            & \zxX{\beta} \ar[.ss,dl] \\
  \zxZ{\alpha}\\
                            & \zxX{\beta} \ar[.ss=40,dl] \\
  \zxZ{\alpha}\\
\end{ZX}
```
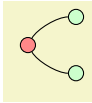
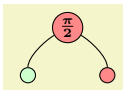I forces the angle above (if in between the two s) or below (if on the same side as .) to be vertical.


```
\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,dl] \\
  \zxZ{\alpha}\\
                            & \zxX{\beta} \ar[I.ss,dl] \\
  \zxZ{\alpha}\\
\end{ZX}
```

/zx/wires definition/<'=angle   (style, default 60)
/zx/wires definition/<.=angle   (style, default 60)
/zx/wires definition/'v=angle   (style, default 60)
/zx/wires definition/v'=angle   (style, default 60)
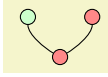
These keys are a bit like (' or (. but the arrival angle is vertical (or horizontal for the ^ (up-down) and v (down-up) versions). As before, the position of the decorator .,' denote the direction of the wire.


```
\begin{ZX}
  \zxN{}                    & & \zxZ{}\\
  \zxX{} \ar[rru,<'] \ar[rrd,<.] & \\
  \zxN{}                    & & \zxZ{}\\
\end{ZX}
```
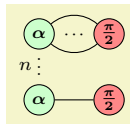


```
\begin{ZX}
  \zxN{} & \zxFracX{\pi}{2} \ar[ddl,.^] \ar[ddr,^.] & \\
       &                                & \\
  \zxZ{} &                                & \zxX{}
\end{ZX}
```

```
\begin{ZX}
  \zxZ{} &                                    & \zxX{}\\
         &                                    & \\
  \zxN{} & \zxX{} \ar[uul,'v] \ar[uur,v'] &
\end{ZX}
```

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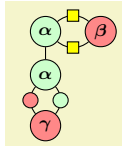/zx/wires definition/3 vdots=text   (style, default =)

The styles put in the middle of the wire (without drawing the wire) . . . (for `3 dots`) or ⋮ (for
`3 vdots`). The dots are scaled according to \zxScaleDots and the text ⟨*text*⟩ 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={£n£\,}] &[\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]                       \\[\zxHSRow]
  \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 loose-
ness 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)
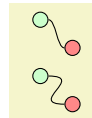and \\[\zxWRow] (for rows) to get nicer spacing):

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

With `wc`:

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

/zx/wires definition/ls=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,ls=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   ⟩⟨

```
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

4.1 Overlaying or creating styles

It is possible to arbitrarily customize the styling, create or update ZX or tikz 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). We also provide overlays to quickly modify the ZX style.

/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.

Note that a style must typically define at least zxZ2, zxX2, zxFracZ6, zxFracX6, \zxH, zxHSmall, zxNoPhaseSmallZ, zxNoPhaseSmallX, zxNone{,+,-,I}, zxNoneDouble{,+,-,I} and all the `phase in label*`, `pil*` styles (see code on how to define them). Because the above styles (notably zxZ* and zxFrac*) are slightly complex to define (this is needed in order to implement `phase in label`), it may be quite long to implement them all properly by yourself.

For that reason, it may be easier to load our default style and overlay only some of the styles we use (see example in `/zx/user overlay nodes` right after). You can check our code in `/zx/styles/rounded style` to see what you can redefine (intuitively, the styles like `my style name` should be callable by the end user, `myStyleName` may be redefined by users

or used in tikzit, and `my@style@name` are styles that should not be touched by the user). The styles that have most interests are `zxNoPhase` (for Z and X nodes without any phase inside), `zxShort` (for Z and X nodes for fractions typically), `zxLong` (for other Z and X nodes) and `stylePhaseInLabel` (for labels when using `phase in label`). These basic styles are extended to add colors (just add Z/X after the name) like `zxNoPhaseZ`... You can change them, but if you just want to change the color, prefer to redefine `colorZxZ`/`colorZxZ` instead (note that this color does not change `stylePhaseInLabelZ/X`, so you are free to redefine these styles as well). All the above styles can however be called from inside a tikzit style, if you want to use tikzit internally (make sure to load this library then in `*.tikzdefs`).

Note however that you should avoid to call these styles from inside `\zx{...}` since `\zx*` and `\zxFrac*` are supposed to choose automatically the good style for you depending on the mode (fractions, labels in phase...). For more details, we encourage the advanced users too look at the code of the library, and examples for simple changes will be presented now.

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

  If a user just wants to overlay some parts of the node styles, add your changes here.
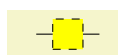
```
{\tikzset{
    /zx/user overlay nodes/.style={
        zxH/.append style={dashed,inner sep=2mm}
    }}
    \zx{\zxNone{} \rar & \zxH{} \rar & \zxNone{}}
}
```

  You can also change it on a per-diagram basis:
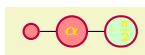
```
\zx[text=yellow,/zx/user overlay nodes/.style={
    zxSpiders/.append style={thick,draw=purple}}
]{\zxX{} \rar & \zxX{\alpha} \rar & \zxFracZ-{\pi}{2}}
```

  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 overlayed 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).

\zxConvertToFracInContent{⟨*sign*⟩}{⟨*num no parens*⟩}{⟨*denom no parens*⟩}{⟨*nom parens*⟩}{⟨*denom parens*⟩}
\zxConvertToFracInLabel

> These functions are not meant to be used, but redefined using something like (we use `\zxMinus` as a shorter minus compared to −):

```
\RenewExpandableDocumentCommand{\zxConvertToFracInLabel}{mmmmm}{%
  \ifthenelse{\equal{#1}{-}}{\zxMinus}{#1}\frac{#2}{#3}%
}
```

This is used to change how the library does the conversion between `\zxFrac` and the actual written text (either in the node content or in the label depending on the function). The first argument is the sign (string `-` for minus, anything else must be written in place of the minus), the second and third argument are the numerator and denominator of the fraction when used in `\frac{}{}` while the last two arguments are the same except that they include the parens which should be added when using an inline version. For instance, one could get a call `\zxConvertToFracInLabel{-}{a+b}{c+d}{(a+b)}{(c+d)}`. See part on labels to see an example of use.

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`
`\zxDefaultSoftAngleChevron`

Default opening angles of `S`, `o` and `v`/`<` wires. Defaults to respectively 30, 40 and 45.

`\zxMinus`

The minus sign used in fractions.

4.2 Further customization

You can further customize your drawings using any functionality from Ti*k*Z and `tikz-cd` (but it is of course at your own risk). For instance, we can change the separation between rows and/or columns for a whole picture using:

```
\begin{ZX}[row sep=1mm]
              &                                          &                        & & \zxZ{\pi} \\
  \zxN{} \rar & \zxX{} \ar[rd,(.] \ar[urrr,(',H]         &                        & & & \zxN{} \\
              &                                          & \zxZ{} \ar[rd,s.] \rar &
    \zxFracX{\pi}{2} \ar[uur,('] \ar[rru,<'] \ar[rr] &  & \zxN{} \\
  \zxN{} \rar & \zxFracZ-{\pi}{4} \ar[ru,('] \ar[rr,o.] &          & \zxX{} \ar[rr] &  & \zxN{}
\end{ZX}
```
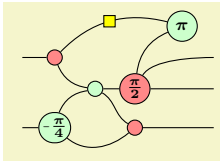
Or we can define our own style to create blocks:
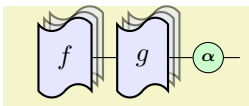

```
{ % \usetikzlibrary{shadows}
  \tikzset{
    my bloc/.style={
      anchor=center,
      inner sep=2pt,
      inner xsep=.7em,
      minimum height=3em,
      draw,
      thick,
      fill=blue!10!white,
      double copy shadow={opacity=.5},tape,
    }
  }
  \zx{|[my bloc]| f \rar &[1mm] |[my bloc]| g \rar &[1mm] \zxZ{\alpha} \rar & \zxNone{}}
}
```

We can also use for instance `fit`, `alias`, `execute at end picture` and layers (the user can use `background` for things behind the drawing, `box` for special nodes above the drawings (like multi-column nodes, see below), and `foreground` which is even higher) to do something like that:
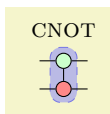

```
% \usetikzlibrary{fit}
\begin{ZX}[
    execute at end picture={
      \node[
        inner sep=2pt,
        node on layer=background, %% Ensure the node is behind the drawing
        rounded corners,
        draw=blue,
        dashed,
        fill=blue!50!white,
        opacity=.5,
        fit=(cnot1)(cnot2), %% position of the node, thanks fit library
        "\textsc{cnot}" above %% Adds label, thanks quote library
      ] {};
    }
  ]
  \zxNone{} \rar & \zxZ[alias=cnot1]{} \dar \rar & \zxNone{}\\
  \zxNone{} \rar & \zxX[alias=cnot2]{} \rar     & \zxNone{}\\
\end{ZX}
```

This can also be used to fake multi-columns nodes (I need to check later if I can facilitate this kind of operation from the library directly):
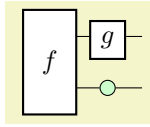
```latex
% \usetikzlibrary{fit}
\tikzset{
  my box/.style={inner sep=4pt, draw, thick, fill=white,anchor=center},
}
\begin{ZX}[
  execute at end picture={
    \node[
      my box,
      node on layer=box, %% Ensure the node is above the wires
      fit=(f1)(f2), %% position of the node, thanks fit library
      label={[node on layer=box]center:£f£},
    ] {};
  }
  ]
  \zxNoneDouble+[alias=f1]{} \rar &[1mm] |[my box]| g \rar & \zxNone{}\\
  \zxNoneDouble+[alias=f2]{} \rar &[1mm] \zxZ{} \rar     & \zxNone{}\\
\end{ZX}
```

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 LaTeXand 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.