

A PGF-based Library for Reo Circuits

Nuno Oliveira

HASLab/INESC TEC
Universidade do Minho

August 13, 2014

Abstract

This article briefly presents the package for Reo Circuits which is an *extension* of the PGF/Tikz package, thoroughly documented in the manual “Tikz & PGF Manual for Version 2.10”.

Herein, the several commands to build a Reo circuit are presented and its *API* explained.

For a good usage of this package, L^AT_EX 2_ε version is required. Some commands may work as well with L^AT_EX version, but it is left as your responsibility.

Moreover, an excellent introduction to Reo can be found in [?], and for more information and tools on this theme the following website may be visited:

<http://reo.project.cwi.nl/>

1 How to Make it Work

Ok, say you want to use this package to draw beautiful Reo Circuits. You start by loading your latex template. Then, in your preamble, you need to have the following:

- `\usepackage{reotex}`

Obviously, you need to have `pgf` and `tikz` packages installed in your environment, otherwise the commands provided in `reotex` will not be interpreted.

2 Drawing a Reo Channel

Reo circuits are the composition of several Reo channels. In this package all the channels have the same API, thus it is easy to explain a generic way of drawing such channels.

2.1 The Generic Format

The following pseudo-code shows how to draw any channel.

```
\tikz \<channel_name>{\<start>}
      {\<target>}
      {\<labels>} ;
```

In detail,

- `\tikz`: refers to the Tikz environment to draw a tikz-based picture. In fact, the `\tikz` command is to draw an inline picture. If you desire to draw in a more controlled environment you may use:

```
\begin{tikzpicture}
  <channel name>{\<start>}
                {\<target>}
                {\<labels>}
\end{tikzpicture}
```

- `<channel name>`: refers to the command (representing the name of a channel) as presented in Section 2.2.
- `<start>` refers to the coordinates of the starting point of the channel. It may be written either as (x, y) or as a reference for a previous defined node: $(name)$. The second format should only be used in a `tikzpicture` environment.
- `<target>`: refers to the coordinates of the target point of the channel. It is used as the `<start>` coordinate.
- `<labels>`: refer to the labels that may be desired to add to the channel. More than one label may be expressed; whenever it happens, a space should separate them. Each label should have the following format:

```
node[<options>] {\<label>} ...
```


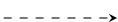



where

- *<options>* are simple coma-separated graphic directives that may be used to transform the label. These directives are those used in normal `pgf` or `tikz` pictures: `yshift=number`, `xshift=number`, `shift=number`, `color name`, `above`, `left`, `below` and `right`, to list some but a few.
- *<label>* is the text to put on the channel. It may be used both simple text and mathematical text.



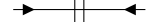
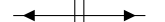




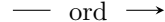
2.2 Reo Channels

The following list presents the commands to draw the 22 defined Reo channels.

2.2.1 Synchronous Channels

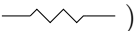
<code>\sync</code>		<code>\sync{(0,0)}{(1.5,0)}{}</code>
<code>\lossysync</code>		<code>\lossysync{(0,0)}{(1.5,0)}{}</code>
<code>\syncdrain</code>		<code>\syncdrain{(0,0)}{(1.5,0)}{}</code>
<code>\syncspout</code>		<code>\syncspout{(0,0)}{(1.5,0)}{}</code>
<code>\filter</code>		<code>\filter{(0,0)}{(1.5,0)}{}</code>

2.2.2 Asynchronous Channels


<code>\fifoe</code>		<code>\fifoe{(0,0)}{(2,0)}{}</code>
<code>\fifof</code>		<code>\fifof{(0,0)}{(2,0)}{}</code>
<code>\asyncdrain</code>		<code>\asyncdrain{(0,0)}{(2,0)}{}</code>
<code>\asynspout</code>		<code>\asynspout{(0,0)}{(2,0)}{}</code>
<code>\fifon</code>		<code>\fifon{(0,0)}{(2,0)}{}</code>
<code>\shiftofifon</code>		<code>\shiftofifon{(0,0)}{(2,0)}{}</code>
<code>\lossyfifon</code>		<code>\lossyfifon{(0,0)}{(2,0)}{}</code>
<code>\timer</code>		<code>\timer{(0,0)}{(2,0)}</code>
<code>\ordered</code>		<code>\ordered{(0,0)}{(2,0)}</code>

<code>\orderedn</code>	— ord_n →	<code>\orderedn{(0,0)}{(2,0)}{}</code>
<code>\bag</code>	— (...) →	<code>\bag{(0,0)}{(2,0)}{}</code>
<code>\bagn</code>	— $(\dots)_n$ →	<code>\bagn{(0,0)}{(2,0)}{}</code>
<code>\set</code>	— {...} →	<code>\set{(0,0)}{(2,0)}{}</code>
<code>\setn</code>	— $\{\dots\}_n$ →	<code>\setn{(0,0)}{(2,0)}{}</code>
<code>\delayset</code>	— DSet →	<code>\delayset{(0,0)}{(2,0)}{}</code>
<code>\delaysetn</code>	— DSet_n →	<code>\delaysetn{(0,0)}{(2,0)}{}</code>
<code>\keyedset</code>	— KSet →	<code>\keyedset{(0,0)}{(2,0)}{}</code>
<code>\keyedsetn</code>	— KSet_n →	<code>\keyedsetn{(0,0)}{(2,0)}{}</code>

Notes:

- The minimum acceptable length (that is, the distance between start and target points) for a channel is 1cm.
- Notice that it is only shown the $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\varepsilon}$ code used to draw the channel. Obviously, this code should be inside a Tikz environment as referred previously).
- All the commands have a starred version that makes them to lose the arrow: (e.g. this code: `\filter*{(0,0)}{(1.5,0)}{}` draws: )

This version is, for instance, used internally for channels that need to be represented in L or U format, for instance (code explained in Section 4):

	<code>\begin{tikzpicture}</code>
	<code>\Uchannel{fifof}{(0,0)}{(2,0)}</code>
	<code>{0.5}{v}{-}{}</code>
	<code>\end{tikzpicture}</code>

3 Drawing a Node

The following code shows how a node is drawn.

3.1 The Generic Format

There are two sorts of nodes in Reo circuits: the boundary nodes and the mixed nodes. In this package they differ only in their colour, as can be seen in Section 3.2. However, the API is the same for both sorts of nodes:

```
\tikz \<node name>{\<reference>}{point}{\<labels>}
```

In detail,

- `\tikz` is the Tikz environment. See the details as described in Section 2.1
- `<node name>` refers to the command (representing the name of the node) as presented in Section 3.2.
- `<reference>` refers to an alias to the x, y position of this node. It should be formatted as `(name)`, where name may be composed of more than one word.
- `<point>` refers to the x and y coordinates of the point where the node is to be placed.
- `<labels>` refer to the text that may be added to the node. It follows the same notation as described in Section 2.1. More than one label may be defined; whenever this happens, a space should separate them.

3.2 Nodes

The following list presents the commands to draw the two sorts of Reo nodes. (Notice that it is only shown the $\text{\LaTeX} 2_{\epsilon}$ code used to draw the node. Obviously, this code should be inside a Tikz environment as referred previously).

<code>\ionode</code>	○	<code>\ionode{(n1)}{(0,0)}{}</code>
<code>\mixednode</code>	●	<code>\mixednode{(n2)}{(0,0)}{}</code>

A third node may be used in Reo Circuits. It is named the Exclusive Router node which is a graphic-alias for the Reo Circuit with the same

name. It follows exactly the same notation as the other two nodes previously presented:

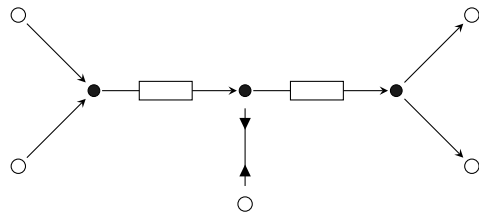
`\xrouter` \otimes `\xrouter{(xr)}{(0,0)}{}`

4 Drawing Reo Circuits

It is strongly recommended to use the `tikzpicture` environment when drawing Reo circuits. The use of the inline command (`\tikz`) may deliver unexpected results.

4.1 A Basic Circuit

So, the main strategy underlying the drawing of Reo Circuits with this package is twofold: first we should place the nodes, so we can name them, and then connect the channels to these nodes, taking advantage of the nodes references.



```

\begin{tikzpicture}
  \ionode{(io1)}{(0,1)}{}
  \ionode{(io2)}{(0,3)}{}
  \ionode{(io3)}{(3,0.5)}{}
  \ionode{(io4)}{(6,1)}{}
  \ionode{(io5)}{(6,3)}{}
  \mixednode{(m1)}{(1,2)}{}
  \mixednode{(m2)}{(3,2)}{}
  \mixednode{(m3)}{(5,2)}{}

  \sync{(io1)}{(m1)}{}
  \sync{(io2)}{(m1)}{}
  \fifoe{(m1)}{(m2)}{}
  \syncdrain{(io3)}{(m2)}{}
  \fifoe{(m2)}{(m3)}{}
  \sync{(m3)}{(io4)}{}
  \sync{(m3)}{(io5)}{}
\end{tikzpicture}

```

If it may become complex to draw the nodes and then understand which node is which, it is strongly recommended to add some labels to the nodes and, afterwards, remove them if they are not desired.

4.2 Drawing U- and L-shaped Channels

Two new commands (`\Lchannel` and `\Uchannel`) are added to this package, as utilities, to draw these shapes.

The `\Lchannel` command is used to define the L-shaped channels. Its notation is:

```
\Lchannel[<dashed>]
    {<channel name>}
    {<start>}
    {<target>}
    {<length>}
    {<orientation>}
    {<polarisation>}
    {<labels>}
```

where,

- *<dashed>* is an optional argument that determines if the *non-channel* part of the L-shape is dashed. If omitted, the result would be a solid line.
- *<channel name>* is the name of the channel that will be L-shaped. These names are exactly the same as the names of the commands presented in Section 2.2, but without the `'\'`
- *<start>* and *<target>* refer to the start and target points of these lines. They may be a reference to a previously defined node or it may be a concrete point. IMPORTANT: *<start>* is always the initial point of the non-channel part of the L-shape, even if, in fact, it is the target point in the drawn figure.
- *<length>* refers to the length of the non-channel part. A desirable value for this argument would be 0.5.
- *<orientation>* refers to the orientation of the non-channel part. IMPORTANT: it must be one of *v* or *h* for vertical or horizontal orientation, respectively.
- *<polarization>* refers to the side to where the non-channel part will grow *length* units. IMPORTANT: it must be one of *+* or *-*. The combination of the following values (respectively the orientation and polarization) mean:

- v and $+$: the non-channel part grows up (positive in the y-axis);
 - v and $-$: the non-channel part grows down (negative in the y-axis);
 - h and $+$: the non-channel part grows to the right (positive in the x-axis);
 - h and $-$: the non-channel part grows to the left (negative in the y-axis);
- `<labels>` refers to the labels that may be added to the channel. Its notation is exactly the same as presented in Section 2.2.
 - This channel has a starred version (`\Lchannel*`) that makes the arrow tip to be placed in the non-channel part of the L-shape. Obviously, the arrow tip, is by default, placed at the channel part of the L-shape.

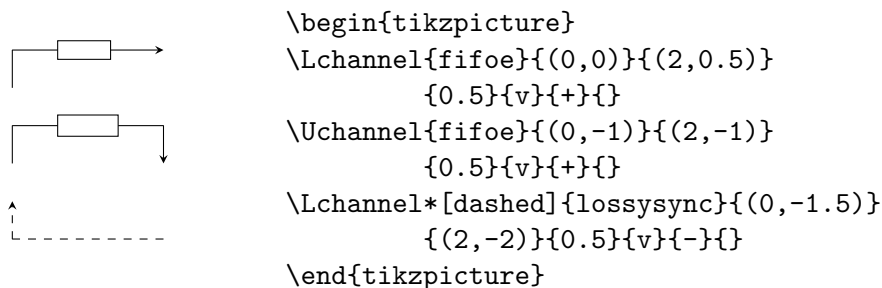
The `\Uchannel` command is used to define the U-shaped channels. Its notation is:

```
\Uchannel [<dashed>]
  {<channel name>}
  {<start>}
  {<target>}
  {<length>}
  {<orientation>}
  {<polarisation>}
  {<labels>}
```

where,

- `<dashed>` is an optional argument that determines if the *non-channel* parts of the U-shape are dashed. If omitted, the result would be a solid line. **IMPORTANT:** this option is sensible to the `lossyfifon` and `shiftfifon` channels, meaning that it creates a dashed and a solid non-channel part.
- `<channel name>` is the name of the channel that will be L-shaped. These names are exactly the same as the names of the commands presented in Section 2.2, but without the `'\'`
- `<start>` and `<target>` refer to the start and target points of these lines. They may be a reference to a previously defined node or it may be a concrete point. **IMPORTANT:** the `<start>` point is always the point of the non-channel part which will not have the arrow tip.

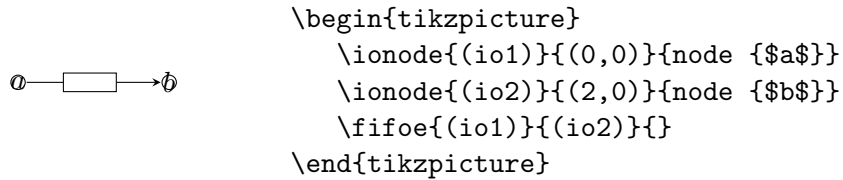
- $\langle length \rangle$ refers to the length of the non-channel parts. A desirable value for this argument would be 0.5.
- $\langle orientation \rangle$ refers to the orientation of the non-channel part. IMPORTANT: it must be one of v or h for vertical or horizontal orientation, respectively.
- $\langle polarization \rangle$ refers to the side to where the non-channel part will grow $length$ units. IMPORTANT: it must be one of $+$ or $-$. The combination of the following values (respectively the orientation and polarization) mean:
 - v and $+$: the non-channel part grows up (positive in the y-axis);
 - v and $-$: the non-channel part grows down (negative in the y-axis);
 - h and $+$: the non-channel part grows to the right (positive in the x-axis);
 - h and $-$: the non-channel part grows to the left (negative in the y-axis);
- $\langle labels \rangle$ refers to the labels that may be added to the channel. Its notation is exactly the same as presented in Section 2.2.



4.3 Adding Labels to Channels or Nodes in a Circuit

Very often it is necessary to add some information to a node or to a channel. In this package, all the channels and nodes allow for this feature. The following examples show how this can be achieved.

Here is an example that adds some label to some nodes.

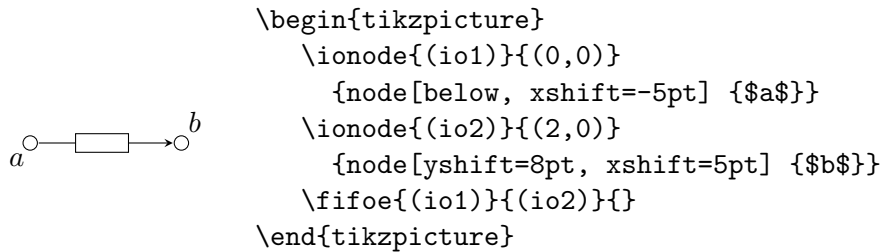


Ups! That is not what we desired!

In fact, the code `node {<text>}` creates a label right in the centre of the node to which it will be stick. This would result in almost invisible labels. The solution is to use position options. Such position options are thoroughly documented in the tikz & pgf manual. Those presented next are easily understandable, and are enough to place the labels in desired positions:

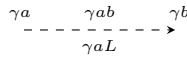
- `yshift` or `xshift`: these options require a dimension d and they make the label to move d units in the vertical or horizontal axes, respectively. They should be used as: `yshift=10pt`, for instance (the same notation is used for for the `xshift`, as it would be expected).
- `above`, `below`, `left` `right`: these options do not require any value as they actually place the label in the expected positions. It may be, however, needed that these options are used with the `y` or `xshift` to place the label in a more suitable position.

Take a look at the next example to understand how these options are used.



It may be interesting to have some (read: more than one) labels in a node or in a channel. The next example shows how more than one label may be added to a channel, defining the stochastic LossySync channel with the stochastic annotations.

Note. Since nodes and channels use the same notation to define labels, the code (the label parts) shown here may be applied to nodes too.

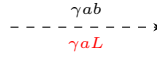


```

\begin{tikzpicture}
  \lossysync{(0,0)}
  {(2,0)}
  {node[above]
  {\tiny$\gamma$ ab$}
  node[below]
  {\tiny$\gamma$ aL$}
  node[above, xshift=-30pt]
  {\tiny$\gamma$ a$}
  node[above, xshift=30pt]
  {\tiny$\gamma$ b$}
  }
\end{tikzpicture}

```

By the way it is possible to change the colour of the label, by adding to the node options the name of a colour. See how in the next example.



```

\begin{tikzpicture}
  \lossysync{(0,0)}{(2,0)}
  {node[above]{\tiny$\gamma$ ab$}
  node[below, red]{\tiny$\gamma$ aL$}
  }
\end{tikzpicture}

```

5 Summary

In this document you learned how to use the reotex package, which allows for the construction of Reo Circuits, using Reo Channels and Nodes.

All the channels follow the same Channel API: receive a start point or a reference to such a start point (always between parentheses), then they receive a list of space-separated labels, and finally they receive the target point or the reference to that point.

The channels have a starred version that defines the existence, or not, of an arrow tip.

The nodes (there are three different) also follow the same Node API: receive a reference name (always between parentheses, composed of one

or more words), then they receive the point and finally the list of space-separated labels.

Moreover, there are some special utility commands that allow for the construction of U- and L-shaped Channels.