xbondgraphs[*]– drawing bond graphs using Ti*k*Z

Marcus J.W. Snippe[†]

May 4, 2018

Abstract

When using the xbondgraphs-package, the user is able to draw visually pleasing bond graphs[1], while mostly maintaining the standard notation of Ti*k*Z drawings. It defines two new PGF arrows, an accompanying decoration to ensure the direction of the barb, as well as a PGF shape for power (de-)mux elements. This package is based on the bondgraphs package by G. Folkertsma[2], but does not (yet) cover all its functions. It *might* result in more appealing bond graphs.

Contents

[*]This document corresponds to xbondgraphs v0.0.1, dated 2018/05/02.
[†]E-mail: m.j.w.snippe@saxion.nl
[1]`https://en.wikipedia.org/wiki/Bond_graph`
[2]`https://ctan.org/pkg/bondgraphs`

1 Introduction

1.1 Motivation

This package is a by-product of a project in which I was in need of a convenient way to draw bond graphs. At first, the bondgraphs package was sufficient, but as the delivery date of the final report approached, I became less and less satisfied by the aesthetic end result of my bond graphs, especially when using multi-bonds. Figure 1 shows a simple comparison between the bondgraphs- and the xbondgraphs package.

(a) Using the bondgraphs package **(b)** Using the xbondgraphs package

Figure 1 – Comparison of multi bond graph drawing.

Figure 1 shows the main motivation for this package. Although of course subjective, most of the differences between the bondgraphs- and the xbondgraphs package can be argued to be improvements. The drawing in figure 1b is overall more consistent. The causality stroke of figure 1a with flow-out causality is overdrawn by the inner line of the multi bond. This is fixed in figure 1b. Most flaws of the drawing in figure 1a can be traced back to the decoration being a `postaction`. This however is needed to inherit other options from the `\draw`-command, e.g. color.

Due to these reasons, I wrote the xbondgraphs package from scratch, re-using some parts but in a completely different setup.

1.2 Alternatives

As already mentioned, this package is based on the bondgraphs package, but does not (yet) cover all its functions. A comparison of main package functions is shown in table 1.

[3] See figure 1.
[4] This is optional.

Table 1 – Function comparison between bondgraphs and xbondgraphs.

	bondgraphs	xbondgraphs
Automatic arrow barb direction	✓	✓
Single bond drawings	✓	✓
Multi bond drawings[3]	✓	✓
Power (de-)mux element	✗	✓
Multi-segment bonds	✗	✓
Curly bond barb	✓	✗
Colon between element and variable[4]	✗	✓

A second alternative is the `bondgraph`[5] package, but because it has nearly no documentation and an incomprehensible example file, I have never tried it personally.

1.3 Known issues

- None yet, but please submit issues to `https://github.com/MaxSnippe/xbondgraphs/issues`.

2 Basic usage

2.1 Installation

This package has not yet been included in popular LaTeX distributions, and therefore can be installed only by downloading the source (`xbondgraphs.sty`) from the GitHub repository[6] to your local TEXMF tree. It should be placed under `$TEXMF$/tex/latex/local`.

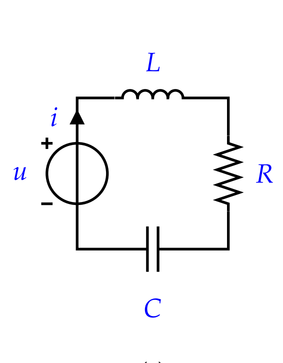
2.2 Including the package

The package can be included with the well-known `\usepackage[<options>]{xbondgraphs}`, where ⟨*options*⟩ can be any of the options mentioned in section 3.1. Options that set the same keys to different values are treated in the order in which they are provided. The package works fine straight-out-of-the-box without setting any options.

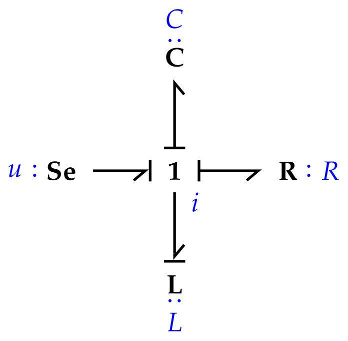
2.3 Simple example

A simple example of an electric domain dynamic model shown as an iconic diagram, and its domain independent equal model shown as a bond graph.

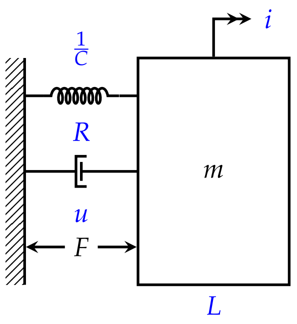
[5]`https://ctan.org/pkg/bondgraph`
[6]`https://github.com/MaxSnippe/xbondgraphs`



(a)



(b)



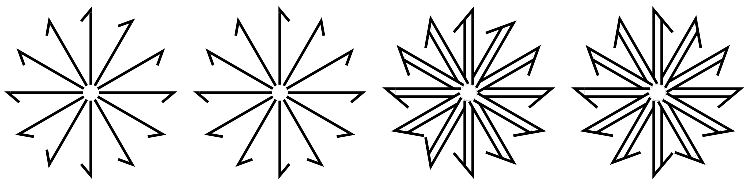
(c)

Figure 2 – Electric domain dynamic model and its bond graph representation.

3 Options

3.1 Global (package) options

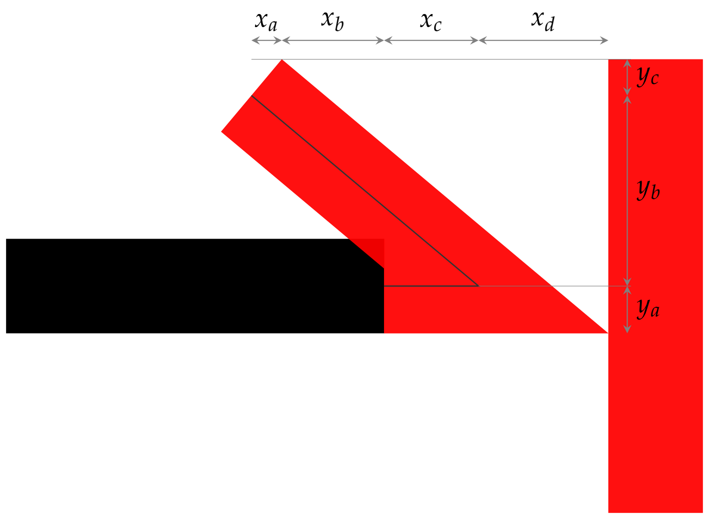
`barbdirection`



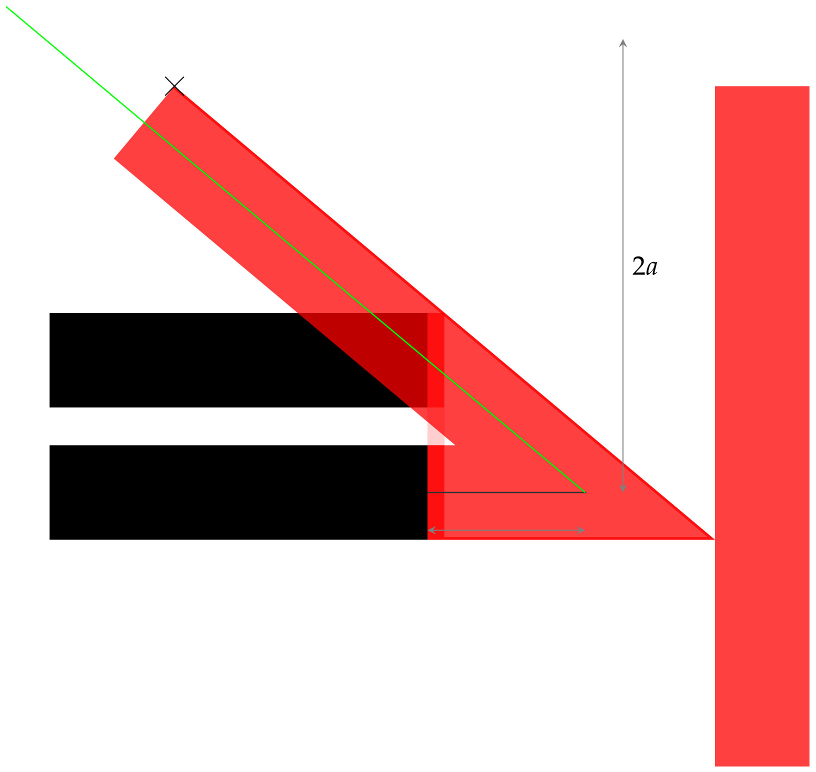
3.2 Local (Ti*k*Z) options

4 Arrow tips

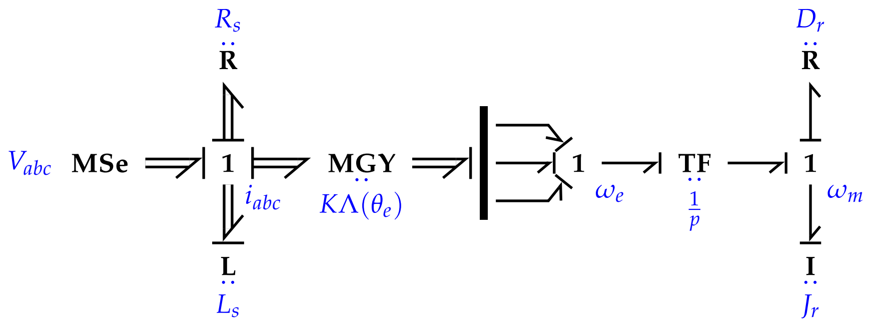
4.1 Single bond arrow tip



4.2 Multi bond arrow tip



5 Examples



6 Implementation

6.1 Package definition

```
⟨*package⟩
\NeedsTeXFormat{LaTeX2e}[2017/04/15]
\ProvidesPackage{xbondgraphs}
    [2018/05/02 v0.0.1 Bond graph drawing using TikZ]
```

6.2 Required packages and libraries

This package uses the pgfopts package to be able to use `pgfkeys` as package options. All the actual drawing is done by Ti*k*Z. The amsfonts package is used for the `\mathbb` font.

```
\RequirePackage{pgfopts}
\RequirePackage{tikz}
\RequirePackage{amsfonts}

\usetikzlibrary{arrows.meta,decorations.markings,shapes}
```

6.3 Arrow tip definitions

The arrow tips are defined using `\pgfdeclarearrow`.

`Single Bond Barb` First the single bond barb is defined. The definition of this arrow is elaborated in section 4.1.

```
\pgfdeclarearrow{
  name = {Single Bond Barb},
  setup code = {
```

First locally define the line width of a single bond, a multibond, and the (absolute) angle the barb makes with the bond.

```
    \pgfmathsetlengthmacro{\sbw}{\pgflinewidth}
    \pgfmathsetlengthmacro{\mbw}{\xbondgraphs@multibondwidth}
    \pgfmathsetlengthmacro{\ba}{\xbondgraphs@barbangle}
```

Calculate the x- and y position of the points that the barb will follow. If one was walking along the bond from startpoint to endpoint, the origin of this scope would be the endpoint, the x direction would be forward, and the y direction would be leftward.

```
    \pgfmathsetlengthmacro{\tipx}{\sbw}
    \pgfmathsetlengthmacro{\tipy}{0pt}
    \pgfmathsetlengthmacro{\backx}{-1/tan(\ba)*(\mbw-0.5*cos(\ba)*\sbw)%
      + \sbw}
    \pgfmathsetlengthmacro{\backy}{\mbw - 0.5*cos(\ba)*\sbw}
```

PGF needs the outer points of the arrow tip to accurately determine the bounding box. Also, the actual tip of the arrow is needed, so the drawn bond will end exactly at the endpoint (the bond Ti*k*Z styles use a `shorten > =` ⟨*dimen*⟩ and `shorten > =` ⟨*dimen*⟩ so they will not end exactly at the endpoint).

```
    \pgfmathsetlengthmacro{\hullpointx}{\backx + 0.5*\sbw*sin(\ba)}
    \pgfmathsetlengthmacro{\hullpointy}{\mbw}
    \pgfmathsetlengthmacro{\tipendx}{0.5*\sbw/tan(\ba/2) + \tipx}
    \pgfmathsetlengthmacro{\tipendy}{-0.5*\sbw}
```

These commands are used to set the outer dimensions that Ti*k*Z/PGF needs.

```
    \pgfarrowssettipend{\tipendx}
    \pgfarrowssetbackend{\backx}
    \pgfarrowshullpoint{\hullpointx}{\hullpointy}
    \pgfarrowshullpoint{\tipendx}{\tipendy}
  },
  drawing code = {
```

The actual drawing of the arrow.

```
    \pgfpathmoveto{\pgfpointorigin}
    \pgfpathlineto{\pgfpoint{\tipx}{\tipy}}
    \pgfpathlineto{\pgfpoint{\backx}{\backy}}
    \pgfusepathqstroke
  },
}
```

`Multi Bond Barb` Repeat all for the multi bond barb. The definition of this arrow is elaborated in section 4.2.

```
\pgfdeclarearrow{
  name = {Multi Bond Barb},
  setup code = {
```

Note that the single bond line width is now read from its PGF key and not from `\pgflinewidth`. The latter now holds the multi bond line width.

```
    \pgfmathsetlengthmacro{\sbw}{\xbondgraphs@singlebondwidth}
    \pgfmathsetlengthmacro{\mbw}{\pgflinewidth}
    \pgfmathsetlengthmacro{\ba}{\xbondgraphs@barbangle}
```

The starting point of the drawing of the actual arrow tip is now were the 'bottom' line ends. The tip end location is calculated such that the centerline of the barb passes through the endpoint of the 'top' double line.

```
    \pgfmathsetlengthmacro{\startx}{0pt}
    \pgfmathsetlengthmacro{\starty}{-0.5*\mbw+0.5*\sbw}
    \pgfmathsetlengthmacro{\tipx}{(\mbw-\sbw)/tan(\ba)}
    \pgfmathsetlengthmacro{\tipy}{-0.5*\mbw + 0.5*\sbw}
    \pgfmathsetlengthmacro{\backy}{1.5*\mbw - 0.5*\sbw*cos(\ba)}
    \pgfmathsetlengthmacro{\backx}{-(\backy+\tipy)/tan(\ba)}
```

The outer dimensions of the arrow are slightly different than for the `Single Bond Barb`, but not much.

```
    \pgfmathsetlengthmacro{\hullpointx}{\backx + 0.5*\sbw*sin(\ba)}
    \pgfmathsetlengthmacro{\hullpointy}{1.5*\mbw}
    \pgfmathsetlengthmacro{\tipendx}{0.5*\sbw/tan(\ba/2) + \tipx}
    \pgfmathsetlengthmacro{\tipendy}{-0.5*\mbw}
```

Again set the PGF dimensions needed for the definition of the arrow tip.

```
    \pgfarrowssettipend{\tipendx}
    \pgfarrowssetbackend{\backx}
    \pgfarrowshullpoint{\hullpointx}{\hullpointy}
    \pgfarrowshullpoint{\tipendx}{\tipendy}
  },
  drawing code = {
```

The drawing is the same as for the `Single Bond Barb`, except for the `\pgfsetlinewidth` that sets the line width to the single bond line width.

```
    \pgfpathmoveto{\pgfpoint{\startx}{\starty}}
    \pgfpathlineto{\pgfpoint{\tipx}{\tipy}}
    \pgfpathlineto{\pgfpoint{\backx}{\backy}}
    \pgfsetlinewidth{\sbw}
    \pgfusepathqstroke
  }
}


% BOND DECORATION
\pgfdeclaredecoration{bond}{initial}{
  \state{initial}[width=\pgfdecoratedinputsegmentlength+1pt]{
    \pgfpathlineto{\pgfpointdecoratedinputsegmentlast}
  }
  \state{final}{
    %
    \pgfmathparse{int((\pgfdecoratedangle+\xbondgraphs@bond@barbdirectionflipangle)/90)}
    \ifcase\pgfmathresult
    \pgfkeys{/xbondgraphs/bond/barb direction=right}
    \or
    \pgfkeys{/xbondgraphs/bond/barb direction=left}
    \or
    \pgfkeys{/xbondgraphs/bond/barb direction=left}
    \else
    \pgfkeys{/xbondgraphs/bond/barb direction=right}
    \fi
    \ifxbondgraphs@bond@causality@eout
    \tikzset{-{\xbondgraphs@bond@barbarrowhead[\xbondgraphs@bond@barbdirection].|[/tikz/causal
    \else
    \ifxbondgraphs@bond@causality@fout
    \tikzset{{|[/tikz/causal stroke style]}-{\xbondgraphs@bond@barbarrowhead[\xbondgraphs@bond@
    \else
    \tikzset{-{\xbondgraphs@bond@barbarrowhead[\xbondgraphs@bond@barbdirection]}}
    \fi
    \fi
    \path[/xbondgraphs/bond/template]\pgfextra{\pgfpathlineto{\pgfpointdecoratedinputsegmentlas
  }
}

% ifs for the bond options
\newif\ifxbondgraphs@bond@causality@eout
```

```
\newif\ifxbondgraphs@bond@causality@fout

% ifs for the element options
\newif\ifxbondgraphs@element@word
\newif\ifxbondgraphs@element@multiport

% Define 'xbondgraphs' as key family for this package
\pgfkeys{
  xbondgraphs/.is family,
  xbondgraphs,
  % Two key families are mainly used, first is 'bond':
  bond/.is family,
  bond,
  template/.style={
    shorten < = 3pt,
    shorten > = 3pt,
    draw,
    line width = \xbondgraphs@singlebondwidth,
  },
  barb direction/.store in=\xbondgraphs@bond@barbdirection,
  barb direction flip angle/.store in=\xbondgraphs@bond@barbdirectionflipangle,
  eout/.is if=xbondgraphs@bond@causality@eout,
  eout=false,
  fout/.is if=xbondgraphs@bond@causality@fout,
  fout=false,
  effort out/.code=\pgfkeys{
    /xbondgraphs/bond/.cd,
    eout=true,
    fout=false,
    /tikz/causal stroke style/.append style={#1}
  },
  flow out/.code=\pgfkeys{
    /xbondgraphs/bond/.cd,
    eout=false,
    fout=true,
    /tikz/causal stroke style/.append style={#1}
  },
  effort in/.code=\pgfkeys{/xbondgraphs/bond/flow out={#1}},
  flow in/.code=\pgfkeys{/xbondgraphs/bond/effort out={#1}},
  multi/.code=\pgfkeys{
    /xbondgraphs/bond/causality stroke scale=3,
    /xbondgraphs/bond/barb arrow head={Multi Bond Barb},
    /xbondgraphs/bond/template/.append style={
      double,double distance={\xbondgraphs@multibondwidth-2*\xbondgraphs@singlebondwidth}
    },
    /tikz/line width = \xbondgraphs@multibondwidth,
  },
  causality stroke scale/.store in=\xbondgraphs@causalitystrokescale,
  causality stroke scale=2,
  barb arrow head/.store in=\xbondgraphs@bond@barbarrowhead,
  barb arrow head={Single Bond Barb},
  label/.style = {
    \xbondgraphs@bondlabelcolor,
  },
  /xbondgraphs,
  % Second key family is 'element':
  element/.is family,
  element,
  n/.store in=\xbondgraphs@element@n,
  n=1,
  word/.is if=xbondgraphs@element@word,
  word=false,
  multiport boolean/.is if=xbondgraphs@element@multiport,
  multiport boolean=false,
  multiport/.code=\pgfkeys{
    /xbondgraphs/element/multiport boolean=true,
  },
  label/.style={
    \xbondgraphs@bgelementlabelcolor,
  },
  % The 'XBG' keys are used as package options
  /XBG/.cd,
  barbangle/.store in=\xbondgraphs@barbangle,
  barbangle=40,
  singlebondwidth/.store in=\xbondgraphs@singlebondwidth,
  singlebondwidth=1pt,
  multibondwidth/.store in=\xbondgraphs@multibondwidth,
  multibondwidth=4pt,
  bgelementlabelcolor/.store in=\xbondgraphs@bgelementlabelcolor,
  bgelementlabelcolor=blue,
  bondlabelcolor/.store in=\xbondgraphs@bondlabelcolor,
  bondlabelcolor=green!50!black,
  gray/.code={
    \pgfkeys{
      /XBG/.cd, bondlabelcolor=gray, bgelementlabelcolor=gray
    }
    \colorlet{diff}{white!60!black}
    \colorlet{error}{white!30!black}
  },
  barbdirection/.is choice,
  barbdirection/leftbelow/.code={\pgfkeys{/xbondgraphs/bond/barb direction flip angle=45}},
  barbdirection/alwaysbelow/.code={\pgfkeys{/xbondgraphs/bond/barb direction flip angle=-1}},
  barbdirection/alwaysbelow,
  /tikz/.cd,
  bond/.style={
    /xbondgraphs/bond,
    #1,
    /tikz,
    draw = none,
    decoration={bond},
    postaction=decorate,
  },
  bond graph element/.code 2 args={
    \pgfkeys{
      /xbondgraphs/element,
      #2
    }
    \tikzset{
      shape=rounded rectangle,
      inner sep = 1.5pt,
      node contents = {%
        \ifxbondgraphs@element@multiport%
        \ifnum\xbondgraphs@element@n=1
        \ensuremath{\mathbb{#1}}%
        \else
        \ensuremath{\mathbb{#1}_{\xbondgraphs@element@n}}
        \fi
        \else%
        \ifnum\xbondgraphs@element@n=1
        \ensuremath{\mathbf{#1}}%
        \else
        \ensuremath{\mathbf{#1}_{\xbondgraphs@element@n}}%
        \fi
        \fi%
      },
      prefix after command={
        \pgfextra{
          \tikzset{
            every pin/.style={
              /xbondgraphs/element/label,
              pin distance = 2pt,
              pin edge={
                draw = none,
                decoration={
                  markings,
                  mark = at position 0.5 with {
                    \node[rotate=\pgfdecoratedangle,inner sep = 0pt,/xbondgraphs/element/label]
                  },
                },
                decorate,
              },
            },
            every label/.style={
              /xbondgraphs/element/label,
            },
          },
        }
      },
    }
    \ifxbondgraphs@element@word
    \tikzset{draw,line width = 0.75\xbondgraphs@singlebondwidth,shape=ellipse}
    \fi
  },
  bond label/.style={
    font=\small,
    /xbondgraphs/bond/label,
    sloped,
  },
  effort/.style={
    edge node={node [bond label,above]{#1}}
  },
  flow/.style={
    edge node={node [bond label,below]{#1}}
  },
  causal stroke style/.style={
    width=\xbondgraphs@causalitystrokescale*\xbondgraphs@multibondwidth,
  },
  }

  % MUX SHAPE
  \pgfkeys{
    /tikz/mux/.code={
    \pgfkeys{
      %      /tikz/shape=mux,
      /mux/.cd,
      #1
    }
    \tikzset{
      outer sep = 0pt,
      inner sep = 0pt,
      minimum width = \pgfkeysvalueof{/mux/width},
      node contents = {},
      fill=black,
      shape=mux,
    }
    },
    /mux/.is family,
    mux,
    inputs/.initial=2,
    outputs/.initial=2,
    io spacing/.initial=5mm,
    width/.initial=3pt,
  }
  \pgfdeclareshape{mux}{
    \savedanchor\centerpoint{%
    \pgf@x=0%
    \pgf@y=0%
    }%
    \inheritsavedanchors[from=rectangle]
    \inheritanchorborder[from=rectangle]
```

```
\inheritanchor[from=rectangle]{north}
\inheritanchor[from=rectangle]{north west}
\inheritanchor[from=rectangle]{north east}
\inheritanchor[from=rectangle]{center}
\inheritanchor[from=rectangle]{west}
\inheritanchor[from=rectangle]{east}
\inheritanchor[from=rectangle]{mid}
\inheritanchor[from=rectangle]{mid west}
\inheritanchor[from=rectangle]{mid east}
\inheritanchor[from=rectangle]{south}
\inheritanchor[from=rectangle]{south west}
\inheritanchor[from=rectangle]{south east}
\savedmacro\inputs{\pgfmathtruncatemacro\inputs{\pgfkeysvalueof{/mux/inputs}}}%
\savedmacro\outputs{\pgfmathtruncatemacro\outputs{\pgfkeysvalueof{/mux/outputs}}}%
\savedmacro\numio{\pgfmathparse{max(\inputs,\outputs)}\pgfmathtruncatemacro\numio\pgfmathresu
\saveddimen\height{%
  \pgfmathparse{max(\pgfkeysvalueof{/mux/inputs},\pgfkeysvalueof{/mux/outputs})}
  \pgfmathparse{(\pgfmathresult) * \pgfkeysvalueof{/mux/io spacing}}
  \pgfmathsetlength\pgf@x{\pgfmathresult}
}
\saveddimen\halfwidth{\pgfmathsetlength\pgf@x{\pgfkeysvalueof{/mux/width}/2}\pgfmathresult}
\saveddimen\iospacing{\pgfmathsetlength\pgf@x{\pgfkeysvalueof{/mux/io spacing}}\pgfmathresult
\backgroundpath{
  \pgfpathrectanglecorners{
    \pgfpointadd{\centerpoint}{\pgfpoint{-\halfwidth}{\height/2}}
  }{
    \pgfpointadd{\centerpoint}{\pgfpoint{\halfwidth}{-\height/2}}
  }
}
\pgfutil@g@addto@macro\pgf@sh@s@mux{%
  % Start with the maximum input number and go backwards.
  % If the anchor is undefined, create it. Otherwise stop.
  \c@pgf@counta=\pgfkeysvalueof{/mux/inputs}\relax%
  \pgfmathloop%
  \ifnum\c@pgf@counta>0\relax%
  \pgfutil@ifundefined{pgf@anchor@mux@input\the\c@pgf@counta}{%
    \expandafter\xdef\csname pgf@anchor@mux@input\the\c@pgf@counta\endcsname{%
      \noexpand\ioanchor{\the\c@pgf@counta}{-1}{((\inputs-\numio)/2+0.5)}%
    }%
  }{\c@pgf@counta=0\relax}%
  \advance\c@pgf@counta-1\relax%
  \repeatpgfmathloop%
}%
\pgfutil@g@addto@macro\pgf@sh@s@mux{%
  % Start with the maximum output number and go backwards.
  % If the anchor is undefined, create it. Otherwise stop.
  \c@pgf@counta=\pgfkeysvalueof{/mux/outputs}\relax%
  \pgfmathloop%
  \ifnum\c@pgf@counta>0\relax%
  \pgfutil@ifundefined{pgf@anchor@mux@output\the\c@pgf@counta}{%
```

```
      \expandafter\xdef\csname pgf@anchor@mux@output\the\c@pgf@counta\endcsname{%
        \noexpand\ioanchor{\the\c@pgf@counta}{1}{((\outputs-\numio)/2+0.5)}%
      }%
    }{\c@pgf@counta=0\relax}%
    \advance\c@pgf@counta-1\relax%
    \repeatpgfmathloop%
  }%
}

\def\ioanchor#1#2#3{%
  \pgfpointadd{\centerpoint}{\pgfpoint{#2*\halfwidth}{\height/2-#1*\iospacing+#3*\iospacing}}%
}

\colorlet{diff}{orange}
\colorlet{error}{red}

% Proces all /XBG keys as package options
\ProcessPgfPackageOptions{/XBG}
⟨/package⟩
```

7 Change History

v0.0.1
General: Initial version 1

8 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.