The \texttt{ForSyDe-LATEX} utilities

George Ungureanu  
Department of Electronics, School of EECS,  
KTH Royal Institute of Technology  
Stockholm, SWEDEN  
March 2, 2018

Abstract
This is the reference manual for the LATEX utilities used in ForSyDe documentations. This manual presents the featured packages, their API, as well as some examples of usage.

1 Introduction
This library was developed as an effort to standardize symbols and graphical primitives in documents related to ForSyDe, but also to provide tools and utilities for user convenience. ForSyDe is a high-level design methodology aiming at correct-by-construction system design flows through formal means. For more information check \url{https://forsyde.github.io}.

The library contains the following main packages:

- \texttt{forsyde-tikz}: is a collection of PGF and TikZ styles, graphical primitives and commands for drawing ForSyDe process networks;
- \texttt{forsyde-math}: is a collection of math symbols used in the ForSyDe formal notation. It is mainly focused on the ongoing ForSyDe–ATOM project;
- \texttt{forsyde-plot}: provides utilities for plotting ForSyDe signals;
- a set of additional TikZ picture libraries commonly used in ForSyDe documentation.

2 Installation & usage
The most straightforward way to install \texttt{ForSyDe-LATEX} is to use the provided GNU Make script command:

\texttt{make install}

which installs the packages in \texttt{TEXMFLOCAL} if you have write access or \texttt{TEXMFHOME} otherwise. Refer to \url{https://en.wikibooks.org/wiki/LaTeX/InstallingExtraPackages} for more information about the two environment variables.

Alternatively, there are three main alternatives to manually install the libraries:

\footnotesize{This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License}
1. copy the contents of `forsyde-latex/src` and `forsyde-latex/fonts` in their appropriate path under `TEXMFHOME` or any standard loading path, as specified by your \LaTeX compiler.

2. compile your document with the variable `TEXINPUTS` set to `/path/to/forsyde-latex/src/`. If you intend to use `forsyde-math` characters, you need to generate the fonts under `forsyde-latex/fonts` using a METAFONT tool suite, and afterwards compile your document with the variable `TEXFONTS` set to `/path/to/forsyde-latex/fonts/`;

3. copy the contents of `forsyde-latex/src` and `forsyde-latex/fonts` in the same folder as your document and compile normally.

To include any of the packages enumerated in the introduction, you can load the `forsyde` package with the appropriate option:

```
\usepackage[option]{forsyde}
```

where `option` is

- `tikz` for loading the `forsyde-tikz` package.
- `math` for loading the `forsyde-math` package.
- `plot` for loading the `forsyde-plot` package.
- `legacy` for using commands defined in previous versions of the library (v0.3 or earlier).

When loaded without an option, this package only provides some general commands for typesetting and logos:

<table>
<thead>
<tr>
<th>Command</th>
<th>Expands to</th>
</tr>
</thead>
<tbody>
<tr>
<td>\ForSyDe</td>
<td>\textsc{Forsyde}</td>
</tr>
<tr>
<td>\ForSyDeLaTeX</td>
<td>\textsc{Forsyde-La\TeX}</td>
</tr>
<tr>
<td>\ForSyDeAtom</td>
<td>\textsc{Forsyde-Atom}</td>
</tr>
<tr>
<td>\ForSyDeShallow</td>
<td>\textsc{Forsyde-Shallow}</td>
</tr>
</tbody>
</table>
# Contents

1 Introduction ................................. 1

2 Installation & usage ....................... 1

3 The forsyde-tikz package ................. 5
   3.1 Main nodes ................................ 5
      3.1.1 Draw commands ....................... 5
      3.1.2 Shapes ................................ 9
      3.1.3 Helper options ....................... 10
   3.2 Paths .................................... 11
      3.2.1 Helper commands ..................... 13
   3.3 Utility commands ....................... 13
   3.4 Further customizing the environment .... 14

4 The forsyde-math package ................. 16
   4.1 Operator symbols ....................... 16
   4.2 Math commands ......................... 17
   4.3 Font map ................................ 17

5 The forsyde-plot package .................... 19
   5.1 The signalsSY environment .............. 19
   5.2 The signalsDE environment .............. 20
   5.3 The signalsCT environment .............. 21

6 Libraries of additional pictures .......... 23
   6.1 The forsyde.pictures.layered library ... 23

7 The forsyde-legacy package ................. 25
   7.1 forsyde-tikz v0.3 or prior ............ 25
   7.2 forsyde-pc v0.3 or prior .............. 25
3 The forsyde-tikz package

This library is used to draw ForSyDe systems modeled as process networks. The package is imported using the command \usepackage[tikz]{forsyde} in the document preamble.

A ForSyDe process network is drawn inside a \tikzpicture environment, like any other TikZ figure. The options provided are key variables defined by the tikz package.

\begin{tikzpicture}[options]
  content...
\end{tikzpicture}

Apart from the standard keys, forsyde-tikz provides the following options variables:

no moc color: disables process coloring according to their MoC.
no moc label: disables process labeling according to their MoC.
label style=: font style for the process type labels. Default is \textbf.
type style=: font style for the process name labels. Default is \scriptsize\textit.
function style=: font style for the functions. Default is \scriptsize.
constructors=[shallow|atom]: switches between the process constructor naming convention of either ForSyDe–Shallow or ForSyDe–Atom. Default is atom.

3.1 Main nodes

Process networks are drawn using a combination of commands and style options for customizing shapes.

3.1.1 Draw commands

Inside the \tikzpicture environment, one can use a set of draw commands\footnote{All draw commands are defined in the \texttt{forsyde.nodes} TikZ library, which is loaded by default by the \texttt{forsyde-tikz} package, but can also be loaded independently using the command \texttt{\usetikzlibrary{forsyde.nodes}} in the document preamble.} which are used as templates for placing shapes and information provided as arguments. In the following listing, the gray arguments are optional while the black ones are mandatory. The library also provide as set of helper styles for setting several options at once, presented in 3.1.3.

\begin{verbatim}
\basic[shape=atom shape, nf=1, f1=+$+1$] (bas) <0,0> {$\oplus$};
\end{verbatim}

Figure 1: The \texttt{basic} draw command. The red dots are custom anchors, accessed with \texttt{name}-f.[anchor].

\begin{verbatim}
\basic[options](name)<position>{label};
\end{verbatim}

The basic node is the simplest depiction of a function (e.g., process, atom) with maximum one argument. It is the most lightweight graphical primitive, and does not contain port anchors. It can be customized with the following keys (check 3.1.3 for composite options):
shape= any TikZ style option for the node shape. Check section 3.1.2 for a list of library-provided shapes. Default is rectangle.

nf=[0..1] the number of passed functions. Default is 0. If nf>1 then nf=1. 
f1= function label. Appears as a box above the main shape in case nf > 0. Default is f1. 
anchor=anchor center point for shape. Default is center.
xshift=[distance] shift position in X direction. Default is 0pt.
yshift=[distance] shift position in Y direction. Default is 0pt.

\standard\{shape=leaf shape, hasmoc, moc=de, ni=8, no=10, type=mealy, nf=4, f1=+$+1$, f3=$\sin$, inner ysep=15pt\}(std)<0,0> {Proc};

Figure 2: The \standard draw command. The function anchors are accessed with [name]_f.[anchor] and the port anchors are accessed with [name].[anchor]

\standard\{options\}(name)<position>\{label\};

The standard node is a more complex depiction of a ForSyDe block. It contains anchors for each port, shows a label and the constructor name, and its field is colored according to a MoC (if this is the case). It can be customized with the following keys (check 3.1.3 for composite options):

class=[sy|de|ct|sdf|sadf|blackbox|none] The class of the node. This option affects the field color and the constructor label. Default is none.

hasmoc flag for saying that this node is a process, therefore it is associated with a MoC. It need to be provided otherwise the global flags will ignore MoC-related options (e.g. nomoccolor).

shape= any TikZ style option for the node shape. Check section 3.1.2 for a list of library-provided shapes. Default is rectangle.

type= the type / constructor of that particular ForSyDe block. It shows below the main label and it might be affected by the moc argument.

npw=[0..10] number of ports on the left side of the node, i.e. the number of left port anchors. ni=[0..10] alias for npw, stands for “number of inputs”.

npe=[0..10] number of ports on the right side of the node, i.e. the number of right port anchors. no=[0..10] alias for npe, stands for “number of outputs”.

nf=[0..4] the number of passed functions. Default is 0.

f1= first function label. Appears as a box in the upper part in case nf > 0. Default is f1.

f2= second function label. Appears as a box in the upper part in case nf > 1. Default is f2.

f3= third function label. Appears as a box in the upper part in case nf > 2. Default is f3.

f4= fourth function label. Appears as a box in the upper part in case nf > 3. Default is f4.
anchor=anchor center point for shape. Default is center.
xshift=dist shift position in X direction. Default is 0pt.
**\interface{}** (if) <0,0> {sy}{sdf};

Figure 3: The \trans draw command. The function anchors are accessed with [name]-f.[anchor] and the port anchors are accessed with [name].[anchor]

\interface[options](name)<position>{domain left}{domain right};

This command draws an interface from one domain to another (e.g. a MoC interface). It can be customized with the following keys (check 3.1.3 for composite options):

- **anchor=anchor** center point for shape. Default is **center**.
- **xshift=dist** shift position in X direction. Default is **0pt**.
- **yshift=dist** shift position in Y direction. Default is **0pt**.
- **rotate=num** the rotation angle (in degrees) of the node. Default is 0.

\trans[shape=trans shape v1v3, ni=1, no=3, nf=1, f1=n, type=splitevery, inner ysep=5pt] (tran) <0,0> {};

Figure 4: The \trans draw command. The function anchors are accessed with [name]-f.[anchor] and the port anchors are accessed with [name].[anchor]

\trans[options](name)<position>{};

This command draws a transversal node over a path (usually 90 degrees), which symbolizes a structural transition of the path’s type (e.g. permutation). The label is ignored, but the brackets are necessary to mark the end of the arguments. It can be customized with the following keys (check 3.1.3 for composite options):

- **shape=** any TikZ style option for the node shape. Check section 3.1.2 for a list of library-provided shapes. Default is **rectangle**.
- **type=** the type / constructor of that particular ForSyDe block. It is shown as the main label, rotated at 90 degrees.
- **npw=[0..10]** number of ports on the left side of the node, i.e. the number of left port anchors.
ni=\{0..10\} alias for npw, stands for “number of inputs”.
npe=\{0..10\} number of ports on the right side of the node, i.e. the number of right port anchors.
no=\{0..10\} alias for npe, stands for “number of outputs”.

nf=\{0..4\} the number of passed functions. Default is 0.
f1= first function label. Appears as a box in the upper part in case nf > 0. Default is $f_1$.
f2= second function label. Appears as a box in the upper part in case nf > 1. Default is $f_2$.
f3= third function label. Appears as a box in the upper part in case nf > 2. Default is $f_3$.
f4= fourth function label. Appears as a box in the upper part in case nf > 3. Default is $f_4$.

anchor=anchor center point for shape. Default is center.
xshift=dist shift position in X direction. Default is 0pt.
yshift=dist shift position in Y direction. Default is 0pt.

\rotate=\num the rotation angle (in degrees) of the node. Default is 0.
\rotate shape=\num the rotation angle (in degrees) of the shape. Default is 0.

\inner ysep=dist the distance between the function node, the label node and the type node. Default is 3pt.

\node[draw] (n1) at (0,0) {n1};
\node[draw] (n2) at (2,1) {n2};
\cluster[shape=generic skel shape, ni=4, no=5, type=mesh, nf=4, f1=+1, f3=$\sin$, inner sep=15 pt] (clus) <(n1)(n2)> {Cluster};

Figure 5: The \cluster draw command. The function anchors are accessed with \[name]-f.[anchor], the port anchors are accessed with \[name].[anchor] and the cluster outer shape edges are named \[name]-[position]
type= the type / constructor of that particular ForSyDe block. It shows below the main label and it is affected by the `class` argument.

- npw=\([0..10]\) number of ports on the left side of the node, i.e. the number of left port anchors.
- ni=\([0..10]\) alias for `npw`, stands for “number of inputs”.
- npe=\([0..10]\) number of ports on the right side of the node, i.e. the number of right port anchors.
- no=\([0..10]\) alias for `npe`, stands for “number of outputs”.
- nf=\([0..4]\) the number of passed functions. Default is \(\text{nf} = 0\).
- f1= first function label. Appears as a box in the upper part in case `nf > 0`. Default is \(f_1\).
- f2= second function label. Appears as a box in the upper part in case `nf > 1`. Default is \(f_2\).
- f3= third function label. Appears as a box in the upper part in case `nf > 2`. Default is \(f_3\).
- f4= fourth function label. Appears as a box in the upper part in case `nf > 3`. Default is \(f_4\).
- anchor=anchor center point for shape. Default is `center`.
- xshift=dist shift position in \(X\) direction. Default is `0pt`.
- yshift=dist shift position in \(Y\) direction. Default is `0pt`.
- rotate=\(num\) the rotation angle (in degrees) of the node. Default is `0`.
- rotate shape=\(num\) the rotation angle (in degrees) of the `shape`. Default is `0`.
- inner xsep=dist the inner separation in \(X\) direction. Default is `3pt`.
- inner ysep=dist the inner separation in \(Y\) direction. Default is `3pt`.

\begin{verbatim}
\node[ports e8w8, inner ysep=18pt] (inside) at (2,0) {}; 
\cluster[shape=trans shape v1v1,inner sep=2pt]<(inside)>{};
\drawconduit{inside}{8}{8};
\foreach \i in {1,3,...,8} \{\draw[very thin, ->] (inside.w\i) -- (inside.e\i);\}
\end{verbatim}

Figure 6: The `\drawconduit` draw command.

\begin{verbatim}
\drawconduit[length]{node with ports}{num. ports west}{num. ports east};
\end{verbatim}

This command draws short lines outside a node with port anchors, that might be interpreted as “conduits” or “ports”. It is used as helper for drawing, for example, custom-shaped transition patterns, like in the figure above.

### 3.1.2 Shapes

ForSYDE-TikZ provides a collection of raw shapes\(^2\) either as TikZ styles or as low-level PGF drawings. They are often used as arguments for drawing commands. Below is a list with them and their usage:

<table>
<thead>
<tr>
<th>Shape</th>
<th>Drawing</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>atom shape</td>
<td><img src="image" alt="atom shape" /></td>
<td>meant to be used with the <code>\basic</code> command, it is a good candidate for depicting ForSyDe atoms or other primitive blocks.</td>
</tr>
</tbody>
</table>

\(^2\)All shapes are defined in the `forsyde.shapes` TikZ library, which is loaded by default by the `forsyde-tikz` package, but can also be loaded independently using the command `\usetikzlibrary{forsyde.shapes}` in the document preamble.
nary atom shape
leaf shape
nary leaf shape
comp shape
nary comp shape
ports

These are multiple shapes that have no drawing, but define additional anchors for ports on the east (.e[index]) and west (.w[index]) sides of the node. It is meant to fit around an existing shape.

func[0..4]
dp shape
pipe shape
merge shape
generic skel shape
trans shape

These shapes create a series of boxes with text and for each box two anchors: north (.n[index]), south (.s[index]). They are used to show functions as input arguments.

dp shape
pipe shape
merge shape
generic skel shape
trans shape

These are meant to be used with the \cluster command, it is a good candidate for depicting exposed hierarchical blocks such as composite processes.

nary leaf shape
leaf shape
comp shape
nary comp shape
ports
e[0..10]w[0..10]

These are multiple shapes that have no drawing, but define additional anchors for ports on the east (.e[index]) and west (.w[index]) sides of the node. It is meant to fit around an existing shape.

3.1.3 Helper options

The forsyde-tikz library provides a large set of styles or function keys that can be used for user convenience as options in the draw commands shown in section 3.1.1.

<table>
<thead>
<tr>
<th>Key</th>
<th>Description/Expands to</th>
</tr>
</thead>
<tbody>
<tr>
<td>moc=x</td>
<td>hasmoc, class=x</td>
</tr>
<tr>
<td>inner sep=x</td>
<td>inner xsep=x, inner ysep=x</td>
</tr>
<tr>
<td>left of=x</td>
<td>positions left of node x, with an edge-to-edge distance = 1cm</td>
</tr>
</tbody>
</table>

3 apart from the ones inherited from rectangle
4 The helpers are defined in the forsyde-tikz package itself, separated from the TikZ libraries.
3.2 Paths

Data flow in Forsyde-Tikz is represented as (directed) edges between components. These are regular \textit{TikZ paths}
 customized with a series of keys provided by the library. There are obvious advantages for using the TikZ infrastructure, such as being able to further customize paths outside the scope of this library, or reuse existing keys (e.g. the arrow tips, shapes). The custom port or function anchors presented in the previous section come in handy when considering port-to-port connections.

The library provides the following custom keys:

\begin{itemize}
  \item \texttt{s=[MoC]} draws a signal. If the global options allow it, passing a MoC string to this key will color the signal accordingly.
  \item \texttt{sn=[MoC]} draws an n-tuple of signals. If the global options allow it, passing a MoC string to this key will color the signal accordingly.
  \item \texttt{v=[MoC]} draws a vector. If the global options allow it, passing a MoC string to this key will color the vector accordingly, suggesting that it contains signals of their respective MoC.
  \item \texttt{vn=[MoC]} draws an n-tuple of vectors. If the global options allow it, passing a MoC string to this key will color the vector accordingly, suggesting that it contains signals of their respective MoC.
  \item \texttt{f} draws an edge used for showing how arguments of a higher order function or a hierarchical block are further passed to its components.
  \item \texttt{fn} similar to \texttt{f}, but for n-tuples of functions.
  \item \texttt{srcport} draws a port symbol at the source of the path.
  \item \texttt{dstport} draws a port symbol at the destination of the path.
  \item \texttt{token=[token-string]} draws symbols for depicting the data structures carried by Forsyde signals or vectors. The accepted tokens strings are \texttt{scalar|vector|function}. To draw tu-
\end{itemize}

\footnote{All paths are defined in the \texttt{forsyde.paths} TikZ library, which is loaded by default by the \texttt{forsyde-tikz} package, but can also be loaded independently using the command \texttt{\usetikzlibrary{forsyde.paths}} in the document preamble.}
ple structures you have to separate token keywords by - (dash). E.g.: 3-tuple of scalar, vector and function can be drawn with \texttt{token=scalar-vector-function}. To depict functions which take other data types as arguments, you can group some tokens between parentheses. E.g. a 2-tuple of a scalar and a function which takes a vector and a scalar as arguments can be drawn with \texttt{token=scalar-(vector-scalar)}.

\texttt{token pos=[0.0 .. 1.0]} the position between the start node/anchor and the end node/anchor of the token symbols’ center position. Default is 0.5.

\texttt{intersect=[path-name]} finds the first intersection with the path named \texttt{[path-name]}, and creates a coordinate from it, named by default \texttt{int} if no other name was given with the \texttt{as} key.

\texttt{as=[coordinate-name]} names the intersection coordinate found with the \texttt{intersect}.

\texttt{trans=[source-path-style]{intersection-path-name}{destination-path-style}} is a helper key which finds the intersection of the current path with \texttt{intersection-path-name}, splits the current path into two, decorated with \texttt{source-path-style} and \texttt{destination-path-style} respectively.

\texttt{-=[0.0 .. 1.0]} will create a horizontal-vertical-horizontal spline in the path. It may be accompanied by a number which determines the position of the 90 degree angle.

\texttt{=-=[0.0 .. 1.0]} will create a vertical-horizontal-vertical spline in the path. It may be accompanied by a number which determines the position of the 90 degree angle.

\texttt{-=[0.0 .. 1.0]} will create a horizontal-vertical-horizontal spline in the path. It may be accompanied by a number which determines the position of the 90 degree angle.

\texttt{=-[0.0 .. 1.0]} will create a horizontal-vertical-horizontal-vertical spline in the path. It may be accompanied by a number which determines the position of the 90 degree angle.

\texttt{=-=[0.0 .. 1.0]:[0.0 .. 1.0]} will create a horizontal-vertical-horizontal-vertical-horizontal spline in the path. It may be accompanied by two numbers separated by : which determine
the position of the 90 degree angles.
\[|-|=0.0..1.0;0.0..1.0\] will create a vertical-horizontal-vertical-horizontal-vertical spline in the path. It may be accompanied by two numbers separated by : which determine the position of the 90 degree angles.
\texttt{deviate=\ldots} is a length representing the deviation from the straight path in case of complex splines (\texttt{--|--} and \texttt{|-|--}).

### 3.2.1 Helper commands

The following commands can be alternatively used to draw paths, in case the complexity of the \texttt{path} Ti\textsc{k}Z command is not needed. The MoC can be passed to the \texttt{moc} style key.

\begin{verbatim}
\signal\[style\] (from) arrow-tip (to);
\signaln\[style\] (from) arrow-tip (to);
\vector\[style\] (from) arrow-tip (to);
\vectorn\[style\] (from) arrow-tip (to);
\function\[style\] (from) arrow-tip (to);
\functionn\[style\] (from) arrow-tip (to);
\end{verbatim}

### 3.3 Utility commands

Following is a list of miscellaneous functions and environments provided for user convenience:

\begin{verbatim}
\textleftof{node}{text}
\textrightof{node}{text}
\textaboveof{node}{text}
\textbelowof{node}{text}
\ifnodedefined{node}{true expr}{false expr}
\gettikzx\[node]\{\xmacro\}
\gettikzy\[node]\{\ymacro\}
\gettikzxy\[node]\{\xmacro\}\{\ymacro\}
\end{verbatim}

Conditional sub-drawing if a node has been defined or not. Useful for ignoring edge cases when drawing recursive structures with \texttt{\foreach}.

\begin{verbatim}
\gettikzx\{node\}\{\xmacro\}
\gettikzy\{node\}\{\ymacro\}
\gettikzxy\{node\}\{\xmacro\}\{\ymacro\}
\end{verbatim}

Gets the X coordinate of \texttt{node} and stores it in \texttt{\xmacro}.

Gets the Y coordinate of \texttt{node} and stores it in \texttt{\ymacro}.

Gets the X and Y coordinates of \texttt{node} and stores them in \texttt{\xmacro} and \texttt{\ymacro} respectively.

\begin{verbatim}
\standard[shape=leaf shape, ni=3, no=2, type=comb, nf=1] (n) <0,0> \{\$P_1\$};
\resetportinfo{n}\wpinfo[\texttt{east}](2)\wpinfo(3)\wpinfo(1)\wpinfo(2)\wpinfo(6)
\end{verbatim}

Figure 8: An example of using the \texttt{\resetportinfo}, \texttt{\wpinfo} and \texttt{\wpinfo} commands

13
\resetportinfo{node}
Resets the counters for the \epinfo and \wpinfo commands which decorates ports with information. These commands can be used for decorating ports, for example, with production and consumption rates in case of SDF processes.

\wpinfo[anchor]{label}
Places a label node next to a western port and increases its counter.

\epinfo[anchor]{label}
Places a label node next to a eastern port and increases its counter.

## 3.4 Further customizing the environment

The following commands can be used in the document preamble for changing environment variables.

\begin{Verbatim}
% Colors
\renewcommand{\defaultdrawcolor}{[color]}
\renewcommand{\defaultfillcolor}{[color]}
\definecolor{sycolor}{[coord sys]}{[color coord]}
\definecolor{ctcolor}{[coord sys]}{[color coord]}
\definecolor{decolor}{[coord sys]}{[color coord]}
\definecolor{sdfcolor}{[coord sys]}{[color coord]}
\definecolor{blackboxcolor}{[coord sys]}{[color coord]}

% line widths
\renewcommand{\compositelinewidth}{[size]}
\renewcommand{\skeletonlinewidth}{[size]}
\renewcommand{\signalpathlinewidth}{[size]}
\renewcommand{\functionpathlinewidth}{[size]}
\renewcommand{\vectorpathlinewidth}{[size]}

% sizes, etc.
\renewcommand{\tokensize}{[size]}
\renewcommand{\halftokensize}{[size]}
\end{Verbatim}
The ForSyDe system which performs the the Fast Fourier Transform can be defined in terms of atoms as:

\[ \text{fft}_S \ k \ vs = \text{bitrev}_S ((\text{stage} \otimes \text{kern}) \hat{\otimes} \ vs) \]  

where the constructors

\[ \text{stage} \ wdt = \text{concat}_S \circ (\text{segment} \otimes \text{twiddles}) \circ \text{group}_S \ wdt \]  
\[ \text{segment} \ t = \text{unduals}_S \circ (\text{butterfly} \ t \ \hat{\otimes}) \circ \text{duals}_S \]  
\[ \text{butterfly} \ w = (((\lambda \ x_0 \ x_1 \rightarrow x_0 + wx_1, x_0 - wx_1) \ \Delta) \ \oplus) \]  

are aided by the number generators

\[ \text{kern} = \text{iterate}_S (\times 2) \ 2 \]  
\[ \text{twiddles} = (\text{reverse}_S \circ \text{bitrev}_S \circ \text{take}_S (\text{lgth}_S vs/2))(\text{wgen} \ \hat{\otimes} \ \langle 1.. \rangle) \]  
\[ \text{wgen} \ x = -\frac{2\pi (x - 1)}{\text{lgth}_S vs} \]
4 The forsyde-math package

This package provides a set of symbols and commands for writing equations. At the moment it covers only the formal notation associated with the ForSyDe-ATOM framework. The package is imported using the command \usepackage{math}[forsyde] in the document preamble.

4.1 Operator symbols

The forsyde-math package exports a set of symbols written in the METAfont language (see [1.3]). These symbols are typed in using commands following the naming convention:

\text{name} % inside a math environment
\text{name} % inside a text environment

where the symbol name is from the table below.\footnote{The symbol naming scheme reflects their semantics defined in the ForSyDe-ATOM framework, as two acronyms: first one denoting the layer and the second one denoting the constructor}

<table>
<thead>
<tr>
<th>&lt;name&gt;</th>
<th>5pt</th>
<th>7pt</th>
<th>8pt</th>
<th>9pt</th>
<th>10pt</th>
<th>12pt</th>
<th>14.4pt</th>
<th>17.28pt</th>
<th>20.74pt</th>
<th>24.88pt</th>
</tr>
</thead>
<tbody>
<tr>
<td>BhFun</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
</tr>
<tr>
<td>BhApp</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
</tr>
<tr>
<td>BhDef</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
</tr>
<tr>
<td>BhPhi</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
</tr>
<tr>
<td>BhDeg</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
</tr>
<tr>
<td>MocFun</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>MocApp</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>MocCmb</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>MocPre</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>MocPhi</td>
<td>○</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>MocDel</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>SkelFrm</td>
<td>○</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>SkelPip</td>
<td>○</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>SkelFun</td>
<td>○</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
</tbody>
</table>
4.2 Math commands

There are a couple of macros defined for math environments, mainly for convenience. They are listed below:

<table>
<thead>
<tr>
<th>Command</th>
<th>Example</th>
<th>Result</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>\id{name}</td>
<td>$\text{id}$</td>
<td>$\text{name}$</td>
<td>wraps \text{name} as an identifier, rather than loose characters</td>
</tr>
<tr>
<td>\context{ctx}{f}</td>
<td>$\text{id}$</td>
<td>$\text{func}$</td>
<td>associates context \text{ctx} to function \text{f}</td>
</tr>
<tr>
<td>\Constructor{name}{layer}</td>
<td>$\text{id}$</td>
<td>$\text{func}$</td>
<td>generic infix name for constructor in a user-defined layer</td>
</tr>
<tr>
<td>\BehCons{name}</td>
<td>$\text{id}$</td>
<td>$\text{default}$</td>
<td>infix name for constructor in the behavior layer</td>
</tr>
<tr>
<td>\MocCons{name}</td>
<td>$\text{id}$</td>
<td>$\text{mesly}$</td>
<td>infix name for constructor in the MoC layer</td>
</tr>
<tr>
<td>\SkelCons{name}</td>
<td>$\text{id}$</td>
<td>$\text{mesh}$</td>
<td>infix name for constructor in the skeleton layer</td>
</tr>
<tr>
<td>\SkelVec{exp}</td>
<td>$\text{id}$</td>
<td>$\text{mesh}$</td>
<td>surrounds \text{exp} in vector type delimiters, associated with the skeleton layer</td>
</tr>
</tbody>
</table>

4.3 Font map

The ForSyDe–Atom operators from 4.1 have been created using METAfont and have been bundled as a font family called forsydeatom. These fonts can be imported in accordance to the LATEX2ε standard. The math symbol font based on this font family is called atomoperators and it declares all symbols as binary operators.

In case you need to access the fonts directly (and not through the forsyde-math package), here is the mapping of the forsydeatom font family:
\documentclass{standalone}
\usepackage[plot,tikz]{forsyde}
\usepackage{filecontents}
\begin{filecontents}{ct-sampde-i1.flx}
0.0 : 0 , 9.983480705722622e-2 : 0.10000000149,
0.5646425989611361 : 0.60000000894 , 0.6442172934010967 : 0.70000001043,
0.96358077769631 : 1.30000001937 , 0.9854495936745004 : 1.40000002086,
0.8632093171439696 : 2.10000003129 , 0.5984721912401382 : 2.50000003725,
0.5155014142351443 : 2.60000003874 , 0.2392493170657283 : 2.90000004321,
-0.15774590023230295 : 3.30000004917,
-0.2555412776230858 : 3.40000005066 , -0.5298363260375504 : 3.70000005513,
-0.6185881429834538 : 3.80000005662 , -0.8715756491042981 : 4.20000006258,
-0.9161645277800696 : 4.30000007003 , -0.9999232696597815 : 4.80000007152,
-0.9961645277800696 : 4.80000007152 , -0.9824524964336662 : 4.90000007301,
-0.82268468292623 : 5.30000008642 , -0.7727641178059984 : 5.400000088046,
-0.3154304920391934 : 6.60000009834 , 0.40484843505764895 : 6.70000010728,
0.728691091159304 : 7.10000011324 , 0.7936684574683253 : 7.20000011771
\end{filecontents}
\begin{filecontents}{ct-sampde-o1.flx}
0.0 : 0,
1.0 : 1.570796326794,
1.793238462865701e-12 : 3.141592653588,
-1.0 : 4.712388980382,
0.0 : 6.283185307176,
1.0 : 7.85398163397,
-3.040302981061924e-7 : 8
\end{filecontents}
\begin{document}
\begin{tikzpicture}
\standard[process, ni=2, no=1, moc=ct, type=sampDE](p1);
\begin{signalsCT}[name=ct-in, timestamps=1.57, grid=1.57, at={p1.west}, anchor=north east, xshift=-.5cm, yshift=.2cm, xscale=1]{7.86}
\signalCT*[outline,ordinate=0,ymin=-1,ymax=1]{ct-sampde-i1.flx}
\end{signalsCT}
\begin{signalsDE}[name=de-in, timestamps=3.14, grid=3.14, at={p1.west}, anchor=south east, xshift=-.4cm, yshift=.2cm]{8}
\signalDE*[trunc]{ct-sampde-o1.flx}
\end{signalsDE}
\begin{signalsDE}[name=de-out, timestamps=3.14, grid=3.14, outputs={p1.east}]{7.8}
\signalDE*[trunc,last label=false]{ 1.0 : 0, 1.0 : 1.570, 1.0 : 3.141, 1.0 : 4.712, 1.0 :
2.683, 1.0 : 7.853, 1.0 : 8 }
\end{signalsDE}
\end{tikzpicture}
\end{document}
5 The forsyde-plot package

This package provides environments and plot commands for visualizing the contents of FORSyDe signals. It is meant to be an alternative to GNUplot or other plotting tools, but not necessarily a replacement for these. The package is imported using the command \usepackage[plot]{forsyde} in the document preamble.

5.1 The signalsSY environment

This environment creates a TikZ matrix of events, where each row is the content of a signal, and “synchronous” events are aligned on columns. This environment is suitable for depicting signals where the time/causality between events is implicit from their position in the signal. This includes synchronous signals, but also any dataflow signals.

\begin{signalsSY}[options]
  body...
\end{signalsSY}

The body of the environment usually consists of a series of \signalSY commands, but it may also contain any command acceptable within a TikZ matrix environment. Notice that it needs an initial line break. The options are:

- name= the name of the TikZ element. Default is sigplot.
- at= the position of the plot within the tikzpicture. Default is (0,0).
- xshift=dist, yshift=dist, shift=coord a displacement in the respective direction.
- anchor=anchor where this element should be anchored.
- left of=coord, right of=coord, above of=coord, below of=coord positions the element accordingly.
- inputs=coord, outputs=coord positions the element left of/right of the given coordinate.

\begin{signalsSY}[name=n]
  \signalSY{1,2,3,4,5,6,7,8,9}
  \signalSY{2,1,2,3,-2,-6,6}
  \signalSY{9,8,7,6,5,4,3,2,1}
\end{signalsSY}

Figure 9: The \signalSY draw command. The red boxes are nodes accessed with [name]=[row]=[column].

\signalSY{events}

The SY signal is simply a row in a matrix, and the elements can be accessed according to the TikZ matrix of nodes.
5.2 The \texttt{signalsDE} environment

This environment creates a (simple) plot of DE signals similar to Modelsim or GTKwave, within a \texttt{tikzpicture}.

\begin{signalsDE}\{\texttt{options}\}\{\texttt{xmax}\}
\texttt{body}...
\end{signalsDE}

The body within the environment consists of a series of \texttt{signalDE} commands. It \textit{requires} an \texttt{xmax} number which stands for the last timestamp plotted. The \texttt{options} are:

- \texttt{name} = the name of the TikZ element. Default is \texttt{sigplot}.
- \texttt{grid}=\texttt{step} draws a dashed line every \texttt{step} time (stamps).
- \texttt{timestamp}=\texttt{step} shows the time (stamp) above the plot at every \texttt{step}.
- \texttt{grid and time}=\texttt{step} is equivalent to \texttt{grid}=\texttt{step} and \texttt{timestamp}=\texttt{step}.
- \texttt{label pos}=position of the label within the plotted bar. Default is \texttt{center}.
- \texttt{signal sep}=\texttt{dist} is the vertical distance between two consecutive signals.
- \texttt{xscale}=\texttt{ratio} and \texttt{yscale}=\texttt{ratio} scales the plot.
- \texttt{at}=the position of the plot within the \texttt{tikzpicture}. Default is (0,0).
- \texttt{xshift}=\texttt{dist}, \texttt{yshift}=\texttt{dist}, \texttt{shift}=\texttt{coord} a displacement in the respective direction.
- \texttt{anchor}=\texttt{anchor} where this element should be anchored.
- \texttt{left of}=\texttt{coord}, \texttt{right of}=\texttt{coord}, \texttt{above of}=\texttt{coord}, \texttt{below of}=\texttt{coord} positions the element accordingly.
- \texttt{inputs}=\texttt{coord}, \texttt{outputs}=\texttt{coord} positions the element left of/right of the given coordinate.

\begin{filecontents}{de-input1.flx}
1:0, 2:2, 3:3, 4:6, 5:8, 5:15
\end{filecontents}
% ...
\begin{signalsDE}\{\texttt{name}=\texttt{n}, \texttt{grid and time}=\texttt{3}\}\{10\}
\texttt{signalDE}\{\texttt{trunc}\}\{2.0:0, -2.0:3, 3.0002:5, -3:8, 5:15\}
\texttt{signalDE}\{\texttt{last label}=\texttt{false}\}\{\texttt{de-input1.flx}\}
\texttt{signalDE}\{\texttt{$\epsilon$}=0, 1.0002:4, $\lambda$:6, 2.0:7, 5:15\}
\end{signalsDE}

\begin{filecontents}{de-input1.flx}
1:0, 2:2, 3:3, 4:6, 5:8, 5:15
\end{filecontents}
% ...
\begin{signalsDE}\{\texttt{name}=\texttt{n}, \texttt{grid and time}=\texttt{3}\}\{10\}
\texttt{signalDE}\{\texttt{trunc}\}\{2.0:0, -2.0:3, 3.0002:5, -3:8, 5:15\}
\texttt{signalDE}\{\texttt{last label}=\texttt{false}\}\{\texttt{de-input1.flx}\}
\texttt{signalDE}\{\texttt{$\epsilon$}=0, 1.0002:4, $\lambda$:6, 2.0:7, 5:15\}
\end{signalsDE}

\begin{figure}[h]
\centering
\begin{tikzpicture}
\begin{scope}
\draw[->] (0,0) -- (6,0) node[below] {\texttt{x}};
\draw[->] (0,0) -- (0,4) node[left] {\texttt{y}};
\foreach \y in {1,...,4}
\draw[step=1, thin, dashed] (0,\y) -- (6,\y);
\foreach \x in {0,1,...,6}
\draw[step=1, thin, dashed] (\x,0) -- (\x,4);
\fill[red] (1,3) circle (2pt) node[below] {n.e1};
\fill[red] (2,2) circle (2pt) node[below] {n.e2};
\fill[red] (3,1) circle (2pt) node[below] {n.e3};
\fill[red] (4,0) circle (2pt) node[below] {n.e4};
\fill[red] (5,-1) circle (2pt) node[below] {n.e5};
\end{scope}
\end{tikzpicture}
\caption{Figure 10: The \texttt{signalDE} draw command. The red dots are custom anchors accessed with \texttt{[name]=\texttt{[anchor]}}.}
\end{figure}
name= the name of the signal.
trunc a flag activating value truncation to integers. Works only if all values of a signal are numbers. Default is false.
last label a flag enabling the last value in a signal to be printed or not. Default is true.

5.3 The signalsCT environment

This environment creates a (simple) stacked plot of CT signals within a tikzpicture.

\begin{signalsCT}[options]{xmax}
body...
\end{signalsCT}

The body within the environment consists of a series of \signalCT commands. It requires an xmax number which stands for the last timestamp plotted. The options are:

name= the name of the TikZ element. Default is sigplot.
grid=step draws a dashed line every step time(stamps).
timestamp=step shows the time(stamp) above the plot at every step.
grid and time=step is equivalent to grid=step and timestamp=step.
label pos= position of the label within the plotted bar. Default is center.
signal sep=dist the vertical distance between two consecutive signals.
yscale= ratio, xscale= ratio scales the plot.
at= the position of the plot within the tikzpicture. Default is (0,0).
xshift=dist, yshift=dist, shift=coord a displacement in the respective direction.
anchor=anchor where this element should be anchored.
left of=coord, right of=coord, above of=coord, below of=coord positions the element accordingly.
inputs=coord, outputs=coord positions the element left of/right of the given coordinate.

\signalCT*[options]{events_or_datafile}

The CT signal command has two versions: a starred (*) version taking as argument the path to a file containing the input data, or the non-starred version which takes as argument the actual data (events). In both cases the data needs to be formatted as

value1 : timestamp1, value2 : timestamp2, ..., valueN : timestampN

where timestampN > xmax. The options are:

name= the name of the signal.
ordinate= draws the X axis (ordinate) on the value passed as argument.
outline a flag for drawing the outline for a signal. Default is false.
ymax=num, ymin=num the maximum and minimum values covered by the plot. Default is ymin=0 and ymax=1.
line style= a style passed to the drawing tool.
\begin{tikzpicture}
\begin{signalsCT}
\signalCT [name=sig-ct1, outline, ymin=-1.1, ymax=2.1, line style={ultra thick}]
{-1.0:0, -1.0:0.9999999776, 0.0:1.009999977376, 0.0:1.9999999552, 1.0:2.009999954976, 1.0:2.9999999328, 2.0:3.009999932576, 2.0:3.29999992608}
\signalCT [name=sig-ct2, outline, ymin=-1.1, ymax=2.1, draw ordinate] {ct-input1.flx}
\signalCT [name=sig-ct3, ymin=-1.1, ymax=2.1, draw ordinate, line style={ultra thick}]
{-1.0:0,-1.0:2.39999994624,0.0:2.409999946016,0.0:3.29999992608}
\signalCT [name=sig-ct4, outline, ymin=-1.1, ymax=2.1, draw ordinate]
{1.0:0,1.0:0.009999999776,1.0:1.39999996864,0.0:1.409999968416,0.0:2.79999993728,-1.0:2.809999937056,-1.0:3.29999992608}
\end{signalsCT}
\end{tikzpicture}

Figure 11: The \texttt{\textbackslash signalCT} draw command. The red dots are custom anchors accessed with \texttt{\textbackslash name}=[anchor].
6 Libraries of additional pictures

The ForSyDe–\LaTeX{} utilities provide a set of additional libraries for drawing pictures used in related publications or documentation. These can be imported as regular TikZ libraries, in association with the forsyde-tikz package.

\usepackage{tikz}{forsyde}
\usetikzlibrary{library}

where library is named along the lines of forsyde.pictures.<name>.

The following libraries are not extensively documented but rather listed along with the main commands and an example of usage. For more information check their source code.

6.1 The forsyde.pictures.layered library

This library provides commands for depicting the concept of layered structure introduced in the ForSyDe–Atom project.

\begin{tikzpicture}
\forsydeAtomMakeLayers[options]{list_of_layers}
\node[\textbf{FunctionLayer}] at (layer0-center) {\textbf{Function}};
\node[\textbf{BehaviorLayer}] at (layer2-center) {\textbf{Behavior}};
\node[\textbf{MoCLayer}] at (layer3-center) {\textbf{ProcessNetwork}};
\node[\textbf{SkeletonLayer}] at (layer4-center) {\textbf{ProcessNetwork}};
\forsydeAtomSignalArrow{layer4-center};
\forsydeAtomHighlightLayer[green!30]{2};
\end{tikzpicture}

Creates a picture of incremental layers wrapping each other, based on the list of labels, formatted in the following way:

inner label 0: outer label 0, inner label 1: outer label 1, ...

Each layer $N$, where $N \in [0..n]$ has:

- a named referred to by other commands: layer$N$;
- defined coordinates/anchors: layer$N$-center, layer$N$-left, and layer$N$-right;
- defined paths: layer$N$-leftpath and layer$N$-rightpath.
The options are:

- **width**: default is 3.8cm;
- **height**: default is 1.8cm;
- **length**: default is 1cm;

\texttt{\textbackslash forsydeAtomSignalArrow[options]{position}}

Creates a picture of a wide arrow representing signal of events. The **position** is where the arrow tip should point to. It has defined the following coordinates/anchors: arrow-south, arrow-north, arrow-east, arrow-west, arrow-center. The **options** are:

- **left indent**: default is 3pt;
- **right indent**: default is 0pt;
- **label shift**: default is -4ex;
- **layer radius**: default is 2cm;

\texttt{\textbackslash forsydeAtomSignalVector{intersection\_path}}

Is used after a \texttt{\textbackslash forsydeAtomSignalArrow} command to draw at which layer **intersection\_path** does a signal is transformed into a structure/vector of signals.

\texttt{\textbackslash forsydeAtomHighlightLayer[color]{layer\_number}}

Highlights a layer by filling it with a background color. Default color is **red**.

The \texttt{forsyde.pictures.layered} library also holds a number of shapes defined as boxes. These boxes are:

- \texttt{\textbackslash forsydeAtomValue}
- \texttt{\textbackslash forsydeAtomExtValue}
- \texttt{\textbackslash forsydeAtomTagValue}
- \texttt{\textbackslash forsydeAtomTagExtValue}
7 The forsyde-legacy package

This package offers an API for the legacy commands defined in older versions of the ForSyDe-LaTeX utilities. This way, documents compiled with old commands can be compiled with the newer versions of their respective library.

7.1 forsyde-tikz v0.3 or prior

Although from v0.4 onward the draw commands have been heavily modified, the old commands could be mapped to the new API.

\begin{verbatim}
\primitive[\keys] {id}{pos}{label}
\primitiven[\keys] {id}{pos}{label}
\leafcustom[\keys] {id}{pos}
\compositestd[\keys] {id}{clustered nodes}{label}
\compositebbox[\keys] {id}{pos}{label}
\patterncluster[\keys] {id}{clustered nodes}{label}
\patternnodestd[\keys] {id}{pos}
\patternnodecustom[\keys] {id}{pos}
\end{verbatim}

7.2 forsyde-pc v0.3 or prior

This package is obsolete and used to hold helpers associated to some ForSyDe process constructors.

\begin{verbatim}
\delay \[\moc=,f1=,inner sep=,reverse\] {id}{pos}{label}
\delayn \[\moc=,f1=,f2=,inner sep=,reverse\] {id}{pos}{label}
\map \[\moc=,f1=,inner sep=,reverse\] {id}{pos}{label}
\comb \[\moc=,f1=,inner sep=,reverse\] {id}{pos}{label}
\combII \[\moc=,f1=,inner sep=,reverse\] {id}{pos}{label}
\combIII \[\moc=,f1=,inner sep=,reverse\] {id}{pos}{label}
\combIV \[\moc=,f1=,inner sep=,reverse\] {id}{pos}{label}
\scanl \[\moc=,f1=,f2=,inner sep=,reverse\] {id}{pos}{label}
\scanlII \[\moc=,f1=,f2=,inner sep=,reverse\] {id}{pos}{label}
\scanlIII \[\moc=,f1=,f2=,inner sep=,reverse\] {id}{pos}{label}
\scanld \[\moc=,f1=,f2=,f3=,inner sep=,reverse\] {id}{pos}{label}
\scanldII \[\moc=,f1=,f2=,f3=,inner sep=,reverse\] {id}{pos}{label}
\scanldIII \[\moc=,f1=,f2=,f3=,inner sep=,reverse\] {id}{pos}{label}
\moore \[\moc=,f1=,f2=,f3=,inner sep=,reverse\] {id}{pos}{label}
\mooreII \[\moc=,f1=,f2=,f3=,inner sep=,reverse\] {id}{pos}{label}
\mooreIII \[\moc=,f1=,f2=,f3=,inner sep=,reverse\] {id}{pos}{label}
\mealy \[\moc=,f1=,f2=,f3=,inner sep=,reverse\] {id}{pos}{label}
\mealyII \[\moc=,f1=,f2=,f3=,inner sep=,reverse\] {id}{pos}{label}
\mealyIII \[\moc=,f1=,f2=,f3=,inner sep=,reverse\] {id}{pos}{label}
\source \[\moc=,f1=,f2=,inner sep=,reverse\] {id}{pos}{label}
\filter \[\moc=,f1=,f2=,inner sep=,reverse\] {id}{pos}{label}
\hold \[\moc=,f1=,inner sep=,reverse\] {id}{pos}{label}
\fillS \[\moc=,inner sep=,reverse\] {id}{pos}{label}
\zip \[\moc=,reverse\] {id}{pos}
\zipII \[\moc=,reverse\] {id}{pos}
\zipIV \[\moc=,reverse\] {id}{pos}
\zipV \[\moc=,reverse\] {id}{pos}
\unzip \[\moc=,reverse\] {id}{pos}
\unzipIII \[\moc=,reverse\] {id}{pos}
\unzipIV \[\moc=,reverse\] {id}{pos}
\end{verbatim}