S*a*-Ti*k*Z*

Claudio Fiandrino
claudio.fiandrino@gmail.com

December 16, 2012

Contents

Introduction

The S*a*-Ti*k*Z library helps in drawing *switching-architectures*. In particular, one of its aims, is to help students to verify if their exercises are correct, but it could also help teachers in preparing lecture notes.

The S*a*-Ti*k*Z library can be loaded by means of:

```
\usetikzlibrary{switching_architectures}
```

and in this case you should also load manually:

*This package has version number *v*0.2b of December 16, 2012; it is released under and subject to the LaTeX Project Public License (LPPL).

```
\usepackage{tikz}
\usetikzlibrary{calc, positioning, decorations.pathreplacing}
```

or by means of:

```
\usepackage{sa-tikz}
```

In the latter case automatically the Ti*k*Z package and the libraries `calc`, `positioning` and `decorations.pathreplacing` are loaded.

The version *v*0.2b provides a way to define Clos Networks Strictly-non-Blocking (snb) and Rearrangeable (rear). Future implementations will provide a way to draw also Benes and Cantor Networks.

1 Basic usage

The simplest use of the package is to define a

`\node`

Basic command definition.

with one of the following options

`/tikz/clos snb` (no value)

Option for drawing a Clos Network Strictly-non-Blocking.

`/tikz/clos rear` (no value)

Option for drawing a Clos Network Rearrangeable.

`/tikz/benes` (no value)

Option for drawing a Benes Network.

`/tikz/benes complete` (no value)

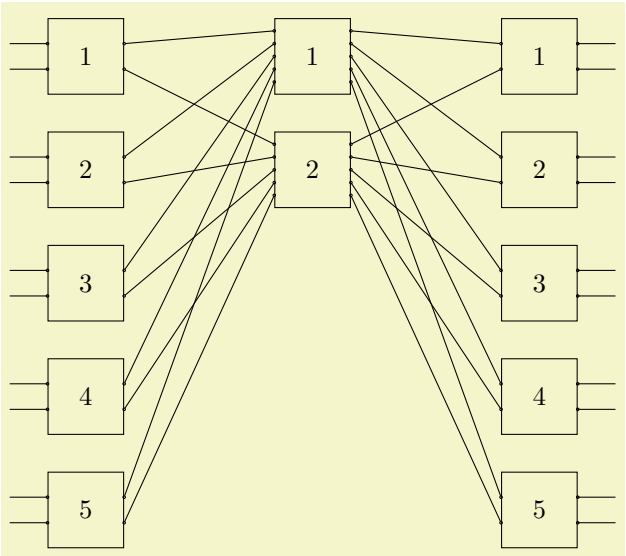
Option for drawing a Benes Network with the lowest level of recursion.

inside a `tikzpicture` environment:

```
\begin{tikzpicture}[⟨options⟩]
  ⟨environment contents⟩
\end{tikzpicture}
```

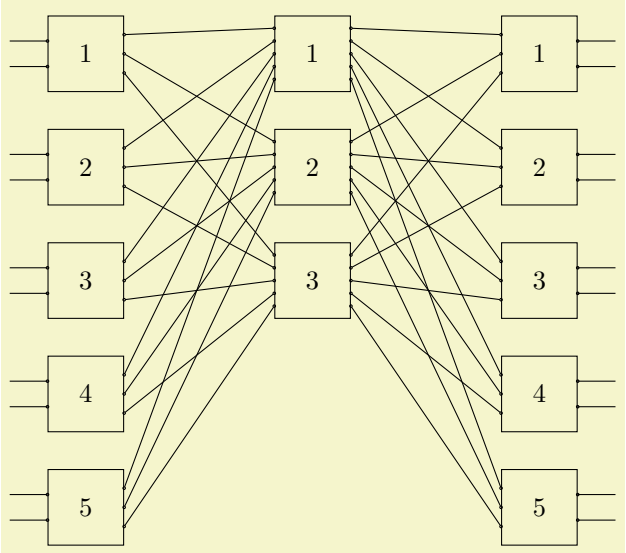
1.1 Examples of Clos Networks

The following example shows a Rearrangeable Clos Network.




```
\begin{tikzpicture}
    \node[clos rear] {};
\end{tikzpicture}
```

The following example shows a Strictly-non-Blocking Clos Network.




```
\begin{tikzpicture}
    \node[clos snb] {};
\end{tikzpicture}
```

Notice from the examples that automatically the library is able to compute the constraints that define a Clos Network to be Strictly-non-Blocking or Rear-

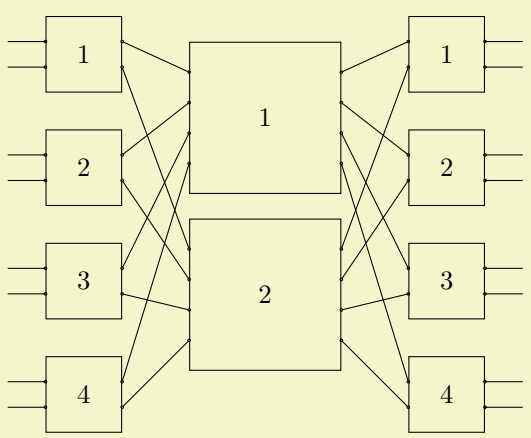
rangeable. Moreover, the network drawn is characterized by:

- the first stage with:
  - a number of modules equal to 5;
  - each one with two input ports;
- the last stage with:
  - a number of modules equal to 5;
  - each one with two output ports.

Each module of the network is numbered according to the stage it belongs to.

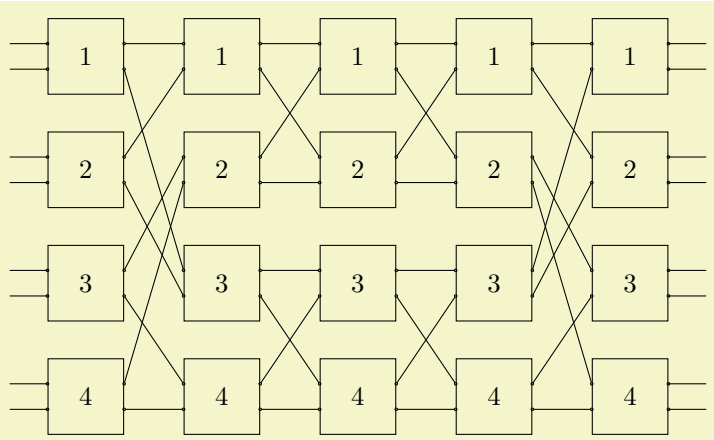
1.2 Examples of Benes Networks

The simplest example of a Benes Network:




```
\begin{tikzpicture}
    \node[benes] {};
\end{tikzpicture}
```

is a Benes Network in which there are 8 input and output ports. To draw a Benes Network in which all modules are visible, the key `benes complete` should be used rather than the `benes` key. An example:




```
\begin{tikzpicture}
    \node[benes complete] {};
\end{tikzpicture}
```

2 The options

2.1 Designing choices - Clos Network

The two first important design parameters are the total number of input ports of the first stage and the total number of output ports of the last stage. These two parameters could be modified by means of:

`/tikz/N={`⟨*value*⟩`}` (no default, initially `10`)

This is the number of total input ports in the first stage.

`/tikz/M={`⟨*value*⟩`}` (no default, initially `10`)

This is the number of total output ports in the last stage.

Usually, a second design parameter is the number of modules present in the first and last stage. S*a*-Ti*k*Z defines:

`/tikz/r1={`⟨*value*⟩`}` (no default, initially `5`)

This is the number of total input ports in the first stage.

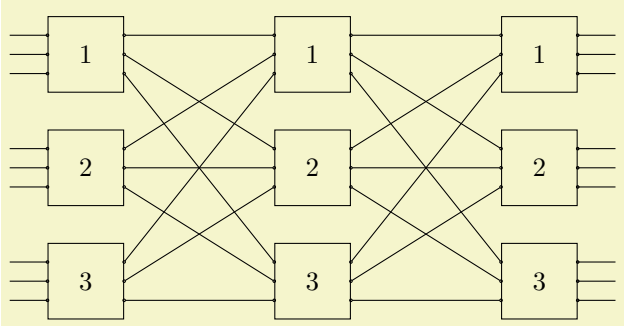
`/tikz/r3={`⟨*value*⟩`}` (no default, initially `5`)

This is the number of total output ports in the last stage.

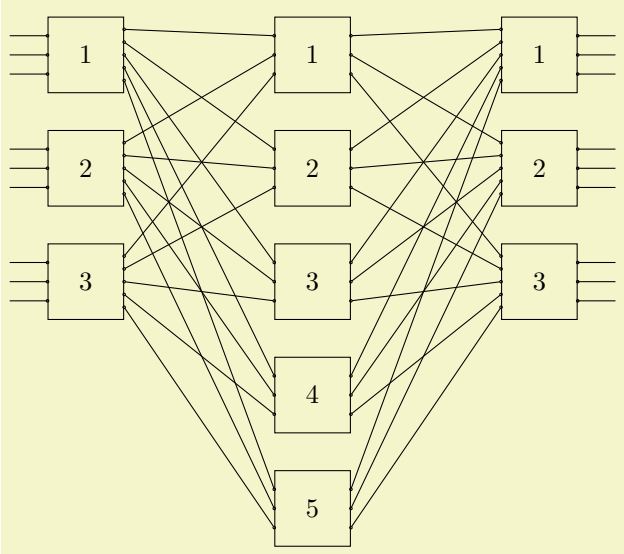
The two design parameters provide the number of ports of each module:

$$\mathrm{m}_1 = \frac{N}{r1} \qquad\qquad \mathrm{m}_3 = \frac{M}{r3}$$

Some examples considering `N`=9, `r1`=3, `M`=9 and `r3`=3.

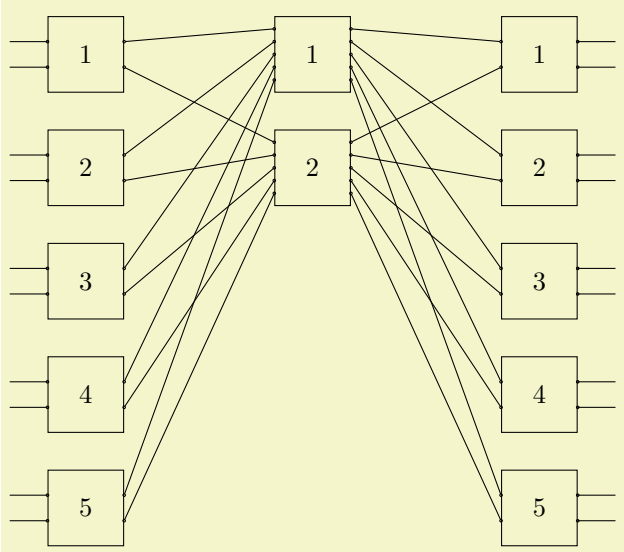



```
\begin{tikzpicture}
    \node[N=9,r1=3,M=9,r3=3,clos rear] {};
\end{tikzpicture}
```




```
\begin{tikzpicture}
    \node[N=9,r1=3,M=9,r3=3,clos snb] {};
\end{tikzpicture}
```

Notice a very important thing: the type of the architecture should be loaded *after* all the design choices when they have been set in the `\node`; indeed, if you do not respect this constraint you will end up with an architecture with default values. For example:




```
\begin{tikzpicture}
    \node[clos rear,N=9,r1=3,M=9,r3=3] {};
\end{tikzpicture}
```

2.2 Designing choices - Benes Network

Benes Networks are composed of 2×2 modules, so as design choice it just possible to select which is the number of input/output ports:

`/tikz/P={`⟨*value*⟩`}` (no default, initially `8`)

This is the number of total input/output ports in the first/third stage.

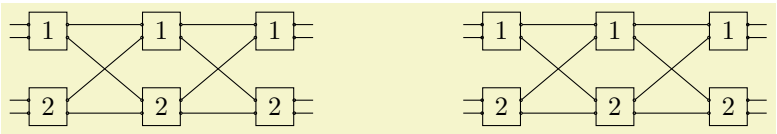
Notice that `P` could assume values

$$P = 2^p \qquad p = 2, 3, 4, \ldots$$

and the user is responsible to correctly set this parameter.

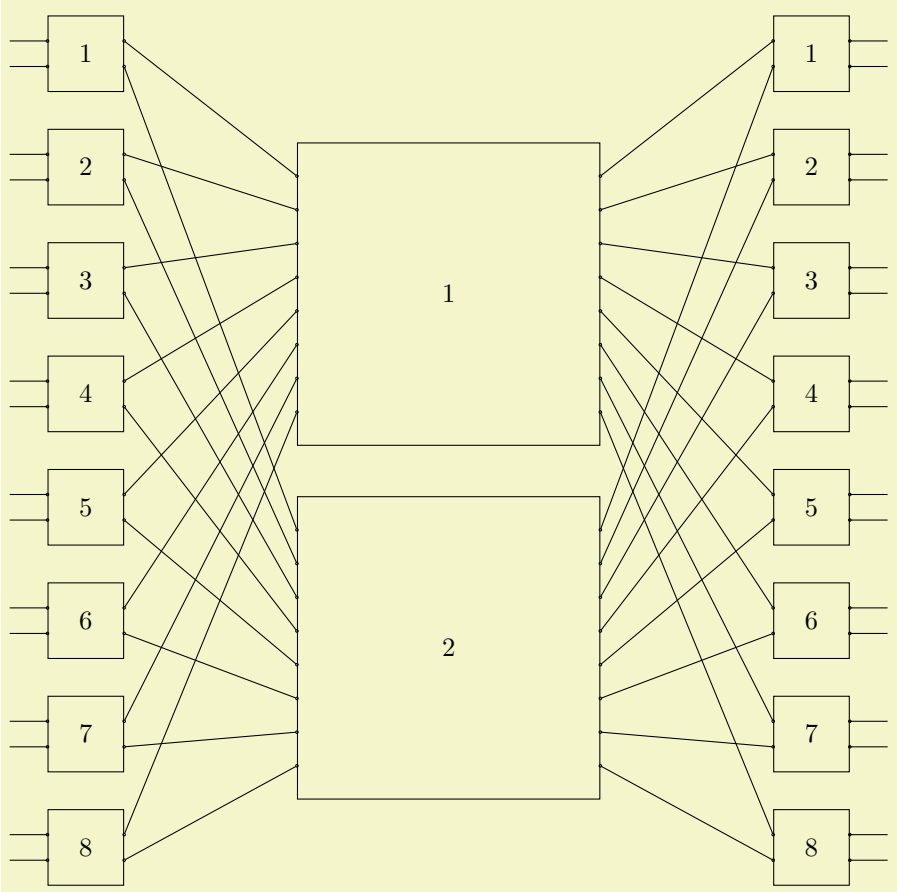
For low values of p there are no problems in visualizing the network, but as p increases the user should take care of the dimension of the modules and the separation (vertical and horizontal) of the modules: it could be customized as explained in subsection 2.3.

Notice that actually, for `P`=4 the `benes` network and the `benes complete` network are indistinguishable:



```
\begin{tikzpicture}
\tikzset{module size=0.5cm,
 pin length factor=0.5,
 module ysep=1}
 \node[P=4,benes] {};
 \begin{scope}[xshift=6cm]
     \node[module xsep=2.5,P=4,benes complete]{};
 \end{scope}
\end{tikzpicture}
```

Here is an example of Benes Network with P=16:




```
\begin{tikzpicture}
    \node[P=16,benes] {};
\end{tikzpicture}
```

It holds the same thing already said for Clos Networks: set the parameter P before declaring the `\node` be a `Benes` Network.

2.3 Output customization

This subsection focuses on how to customize the aspect of the drawing.

`/tikz/module size={`⟨*value*⟩`}` (no default, initially `1cm`)

This option allows to set the module dimension.

`/tikz/module ysep={`⟨*value*⟩`}` (no default, initially `1.5`)

This option allows to set the vertical module distance factor.

`/tikz/module xsep={`⟨*value*⟩`}` (no default, initially `3`)

This option allows to set the horizontal module distance factor.

`/tikz/module label opacity={`⟨*value*⟩`}` (no default, initially `1`)

This option allows to mask the module label when the ⟨*value*⟩ is set to 0.

`/tikz/pin length factor={`⟨*value*⟩`}` (no default, initially `1`)

This option allows to reduce/increase the length of the pins drawn in input/output. Use a ⟨*value*⟩ [0,1] to reduce the length or, viceversa, a ⟨*value*⟩ greater than 1 to increase the length.

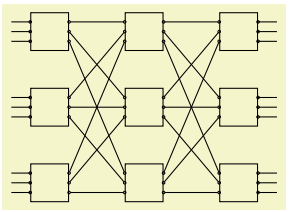
`/tikz/module font={`⟨*font commands*⟩`}` (default `\normalfont`)

This option sets the font used for module labels. The ⟨*font commands*⟩ that could be used are those ones related to the font size (i.e. `\Large`) and font shape (i.e `\itshape`).

`/tikz/connections disabled=true|false` (default `false`)

This option, not active by default `connections disabled/.default=false`, allows to remove the connections between the stages when set to true. Beware: this option is valid only for `clos snb`, `clos rear`, `benes` and `benes complete` networks.

The following example shows a Rearrangeable Clos Network with some options modified. Notice that the `module label opacity` should be given as parameter of the desired network.



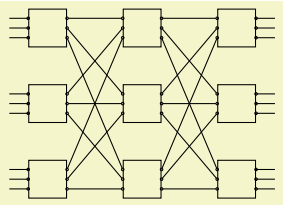
```
\begin{tikzpicture}[N=9,r1=3,M=9,r3=3]
    \node[module size=0.5cm,pin length factor=0.5,
        module ysep=1, module xsep=1.25,
        clos rear={module label opacity=0}] {};
\end{tikzpicture}
```

The options could also be introduced with the standard Ti*k*Z syntax:

`\tikzset{`⟨*options*⟩`}`

Command that process the various ⟨*options*⟩: they should be provided separated by a comma.

Therefore, the previous example could be modified into:

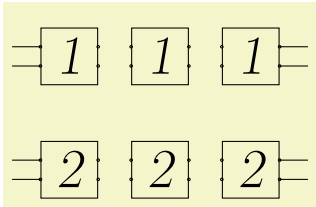


```
\tikzset{module size=0.5cm,pin length factor=0.5,
        module ysep=1, module xsep=1.25}
\begin{tikzpicture}[N=9,r1=3,M=9,r3=3]
    \node[clos rear={module label opacity=0}] {};
\end{tikzpicture}
```

It is also possible to declare styles to set some options for later use: this helps to keep the code clean especially when the same options are re-used several times; an example:

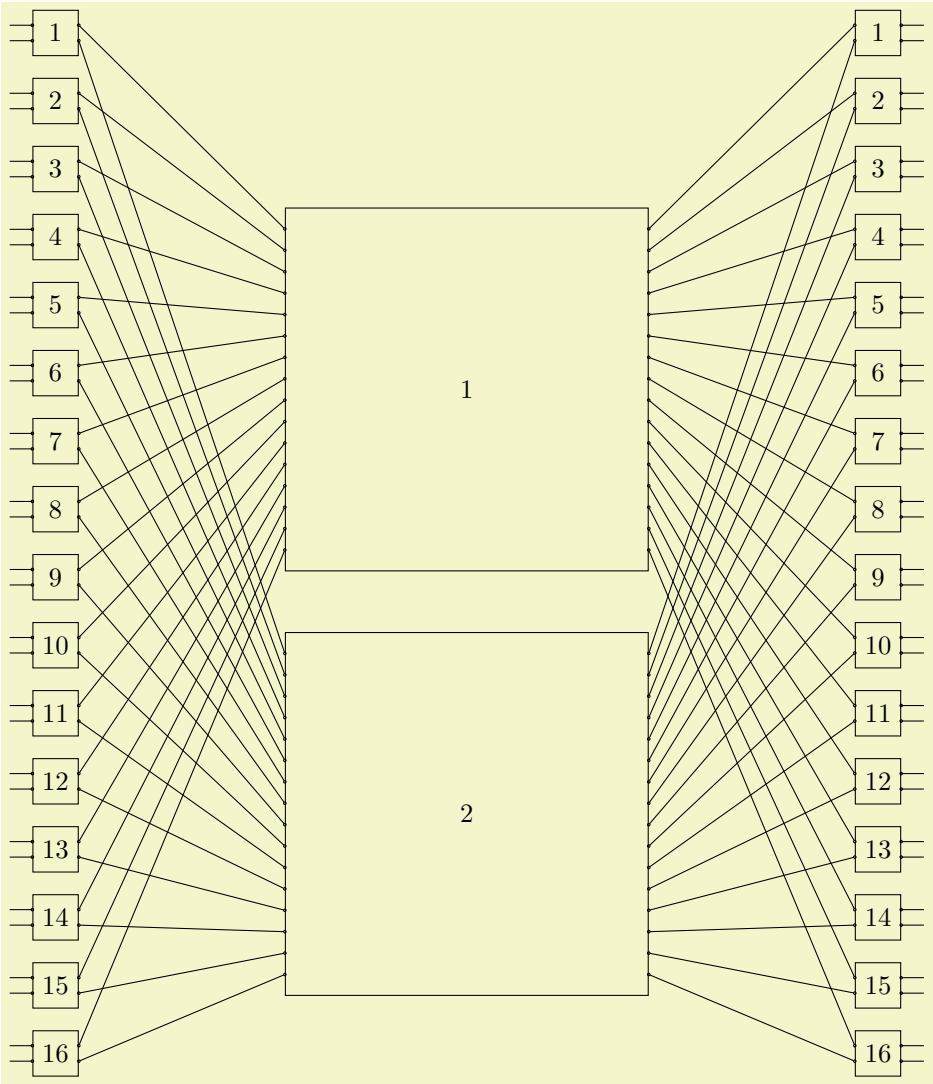
```
\tikzset{module size definition/.style={
  module size=0.75cm,
  pin length factor=0.75,
  module xsep=2,
  module ysep=2,
  }
}
\tikzset{module size definition,
 P=16,
}
\begin{tikzpicture}
    \node[benes] {};
\end{tikzpicture}
```

Here is a Benes Network 4×4 with an extremely large font size for the module labels with the connections disabled:



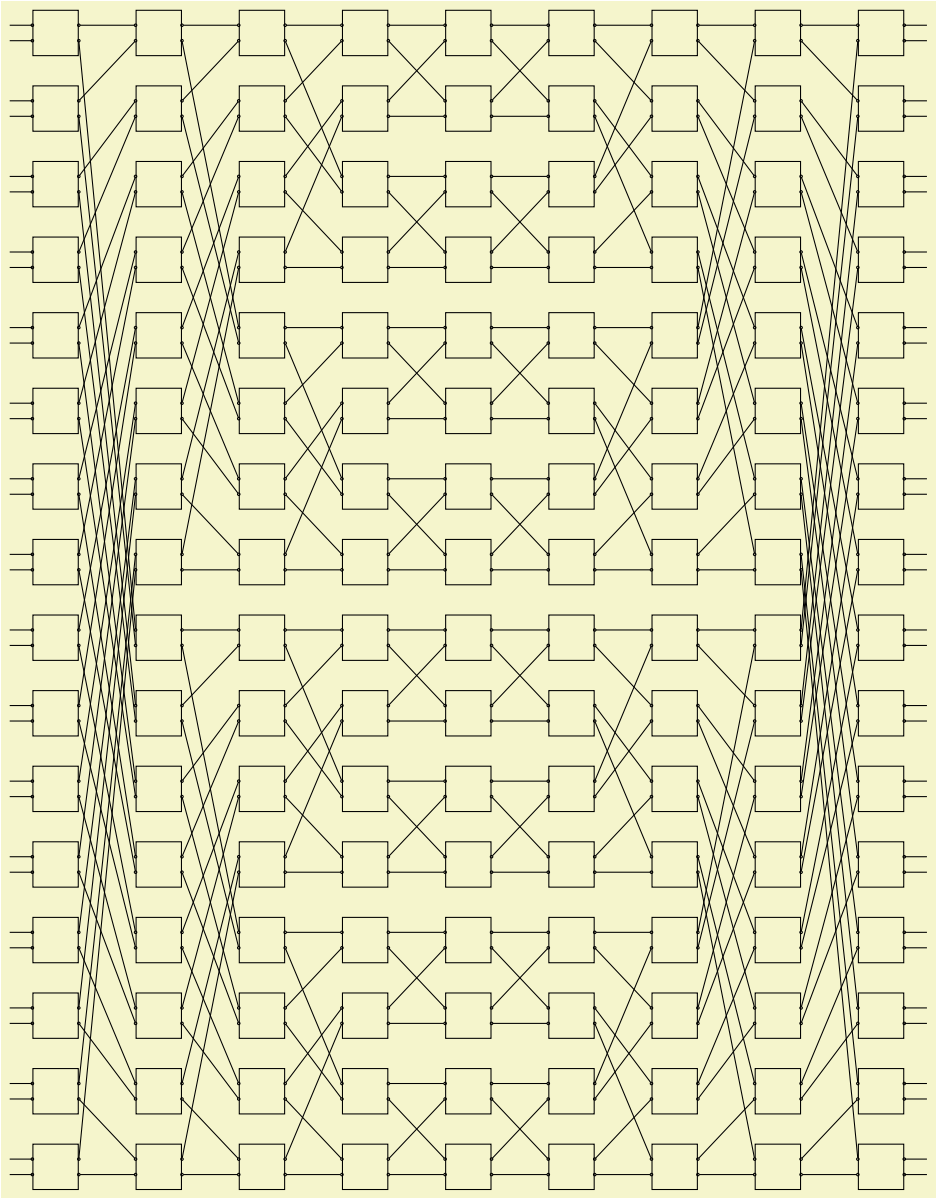

```
\tikzset{my style/.style={
  module size=0.75cm,
  pin length factor=0.75,
  module xsep=2,
  }
}
\tikzset{my style, P=4,
 module font=\huge\slshape,
 connections disabled=true
}
\begin{tikzpicture}
    \node[benes complete] {};
\end{tikzpicture}
```

An example of Benes Network 32×32:




```
\tikzset{module size=0.6cm,pin length factor=0.6,
         module ysep=0.9, module xsep=1.7}
\begin{tikzpicture}[P=32]
    \node[benes] {};
\end{tikzpicture}
```

and its complete form:



```
\tikzset{module size=0.6cm,pin length factor=0.6,
         module ysep=1, module xsep=2.275}
\begin{tikzpicture}[P=32]
    \node[benes complete={module label opacity=0}] {};
\end{tikzpicture}
```

3 Advanced usage

In this section some more advanced examples are shown.

3.1 Identifying front input/output ports

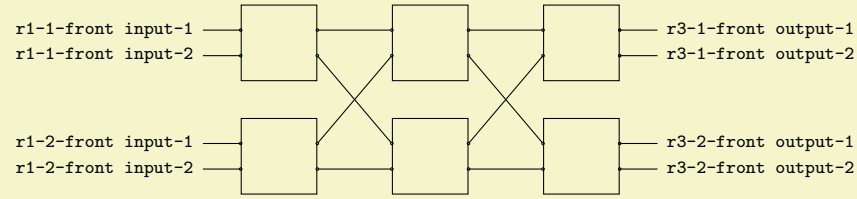
In this subsection it is shown how to reference the front input and output ports for the first and last stage. Each front input port could be accessed by means of:

`r1-module number-front input-port number`; example:
`r1-1-front input-1`;

Each front output port could be accessed by means of:

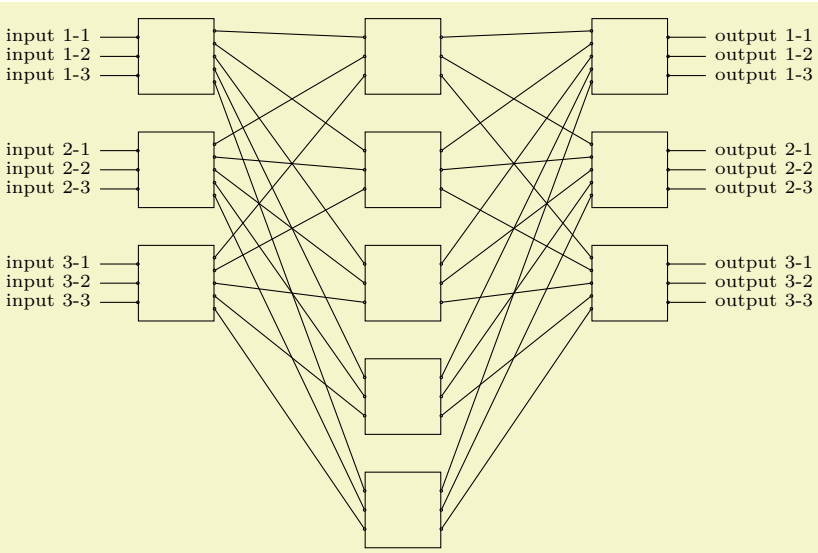
`r3-module number-front output-port number`; example:
`r3-1-front output-1`;

A simple example with a Rearrangeable Clos network of 4 input and output ports; the first stage and the last one have both 2 modules.




```
\begin{tikzpicture}[module xsep=2]
  \node[N=4,r1=2,M=4,r3=2,clos rear={module label opacity=0}] {};
  \foreach \name
    in {r1-1-front input-1,r1-1-front input-2,
        r1-2-front input-1,r1-2-front input-2}
    \node[left] at (\name) {\scriptsize{\texttt{\name}}};
  \foreach \name
    in {r3-1-front output-1,r3-1-front output-2,
        r3-2-front output-1,r3-2-front output-2}
    \node[right] at (\name) {\scriptsize{\texttt{\name}}};
\end{tikzpicture}
```

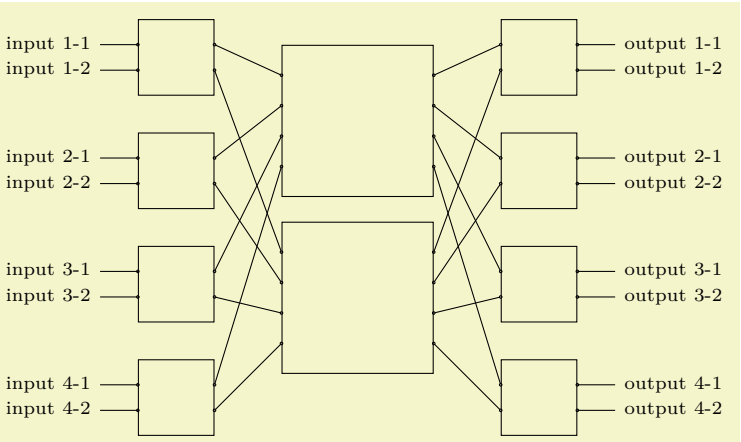
The following is a Strictly-non-Blocking Clos network of 9 input and output ports in which the first and last stage have 3 modules each one.




```
\begin{tikzpicture}
    \node[N=9,r1=3,M=9,r3=3,clos snb={module label opacity=0}] {};

\foreach \startmodule in {1,...,3}{
\foreach \port in {1,...,3}
 \node[left] at (r1-\startmodule-front input-\port)
 {\scriptsize{input \startmodule-\port}};
}
\foreach \startmodule in {1,...,3}{
\foreach \port in {1,...,3}
 \node[right] at (r3-\startmodule-front output-\port)
 {\scriptsize{output \startmodule-\port}};
}
\end{tikzpicture}
```

The same applies also for Benes Networks:



```
\begin{tikzpicture}
    \node[benes={module label opacity=0}] {};

\foreach \startmodule in {1,...,4}{
\foreach \port in {1,...,2}
 \node[left] at (r1-\startmodule-front input-\port)
 {\scriptsize{input \startmodule-\port}};
}
\foreach \startmodule in {1,...,4}{
\foreach \port in {1,...,2}
 \node[right] at (r3-\startmodule-front output-\port)
 {\scriptsize{output \startmodule-\port}};
}
\end{tikzpicture}
```

3.2 Identifying input/output ports per module

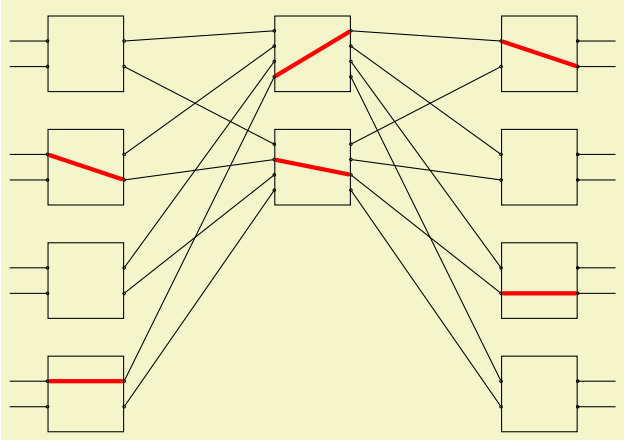
It is also possible to access, for each module of each stage, its input and output ports. The syntax is similar to the one used for the front input and output ports; each input port could be accessed by means of:

`rstage number-module number-input-port number`; example: `r1-1-input-1`;

Each output port could be accessed by means of:

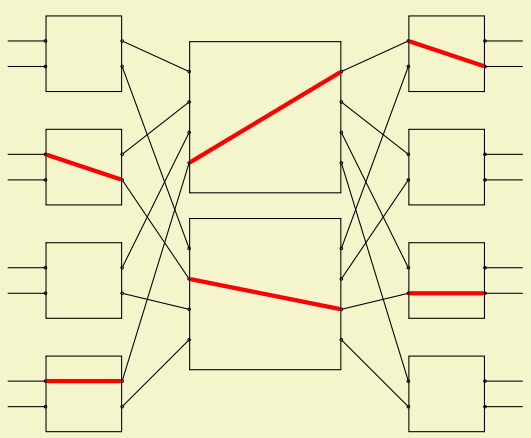
`rstage number-module number-front output-port number`; example: `r2-1-output-1`;

This allows to derive connections from the first stage to the last stage. Here is an example.



```
\begin{tikzpicture}
    \node[N=8,r1=4,M=8,r3=4,clos rear={module label opacity=0}] {};
    \draw[red,ultra thick](r1-2-input-1)-(r1-2-output-2)
    (r2-2-input-2)-(r2-2-output-3)
    (r3-3-input-2)-(r3-3-output-2);
   \draw[red,ultra thick](r1-4-input-1)-(r1-4-output-1)
    (r2-1-input-4)-(r2-1-output-1)
    (r3-1-input-1)-(r3-1-output-2);
\end{tikzpicture}
```

Similarly, in a Benes Network:



```
\begin{tikzpicture}
    \node[benes={module label opacity=0}] {};
    \draw[red,ultra thick](r1-2-input-1)-(r1-2-output-2)
    (r2-2-input-2)-(r2-2-output-3)
    (r3-3-input-2)-(r3-3-output-2);
   \draw[red,ultra thick](r1-4-input-1)-(r1-4-output-1)
    (r2-1-input-4)-(r2-1-output-1)
    (r3-1-input-1)-(r3-1-output-2);
\end{tikzpicture}
```

4 Didactic purposes

To quickly draw a Clos Network it is possible to exploit:

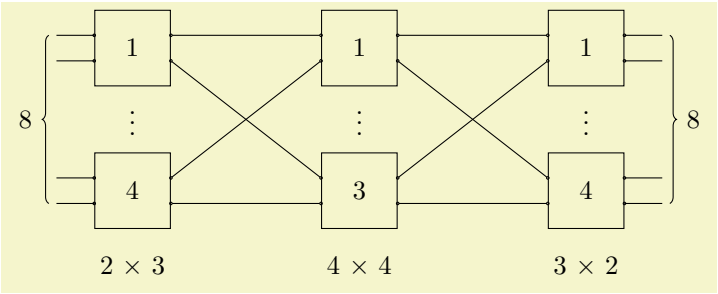
`/tikz/clos snb example` (no value)

Option for quickly drawing a Clos Network Strictly-non-Blocking.

`/tikz/clos rear example` (no value)

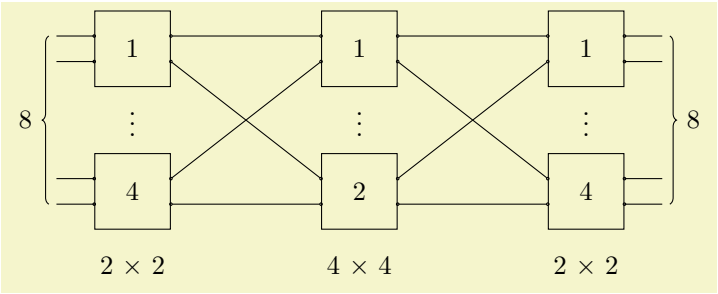
Option for quickly drawing a Clos Network Rearrangeable.

In this way the network is not seen in its whole complexity, but it is synthetically depicted. An example of a Strictly-non-Blocking Clos Network drawn with this approach:




```
\begin{tikzpicture}[N=8,r1=4,M=8,r3=4]
    \node[clos snb example] {};
\end{tikzpicture}
```

Similarly, an example of a Rearrangeable Clos Network:




```
\begin{tikzpicture}[N=8,r1=4,M=8,r3=4]
    \node[clos rear example] {};
\end{tikzpicture}
```

The networks drawn, automatically display the values at which the input parameters N, M, r1 and r3 have been set. However, to let the user to have the possibility of deploying labels rather than the input parameter values, the following option is available:

`/tikz/clos example with labels` (no value)

Option for quickly drawing a Clos Network with custom labels.

The labels could be customized by means of:

`/tikz/N label={`⟨*value*⟩`}` (default `N`)

This options sets the label representing the total number of ports in the first stage.

`/tikz/r1 label={`⟨*value*⟩`}` (default $\mathtt{r}_1$)

This options sets the label representing the number of modules in the first stage.

`/tikz/m1 label={`⟨*value*⟩`}` (default $\mathtt{m}_1$)

This options sets the label representing the number of ports per module in the first stage.

`/tikz/r2 label={`⟨*value*⟩`}` (default r_2)

This options sets the label representing the number of modules in the second stage.

`/tikz/M label={`⟨*value*⟩`}` (default `M`)

This options sets the label representing the total number of ports in the last stage.

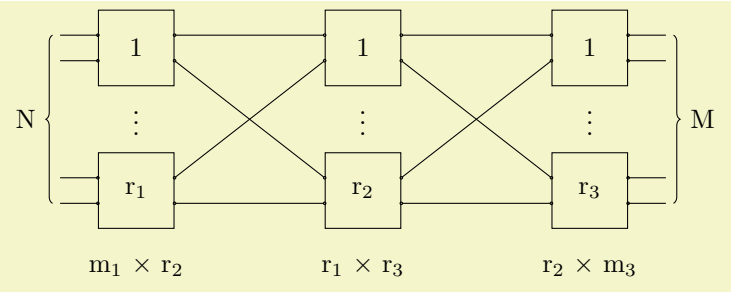
`/tikz/r3 label={`⟨*value*⟩`}` (default r_3)

This options sets the label representing the number of modules in the last stage.

`/tikz/m3 label={`⟨*value*⟩`}` (default m_3)

This options sets the label representing the number of ports per module in the last stage.

An example with the default values:



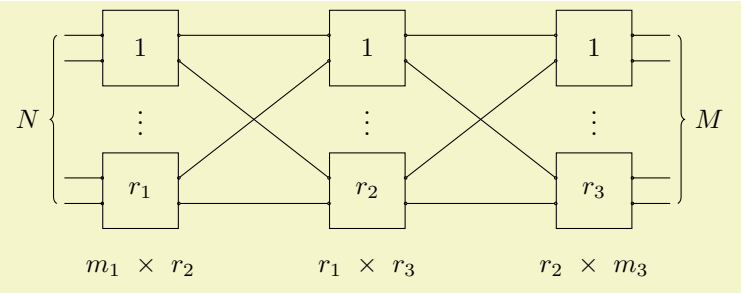

```
\begin{tikzpicture}[N=8,r1=4,M=8,r3=4]
    \node[clos example with labels] {};
\end{tikzpicture}
```

For automatically have labels in math mode, use:

`/tikz/set math mode labels=true|false` (default `false`)

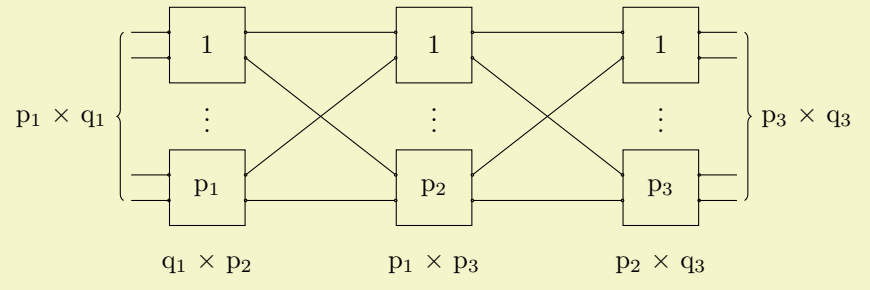
This option is normally disabled `set math mode labels/.default=false`; to ensure labels be set completely in math mode is sufficient set `set math mode labels=true` before the type of the network.

An example with labels in math mode:




```
\begin{tikzpicture}[N=8,r1=4,M=8,r3=4, set math mode labels=true]
    \node[clos example with labels] {};
\end{tikzpicture}
```

Here is an example with custom labels introduced by means of the `\tikzset` syntax.




```
\tikzset{N label={p$_1$ $\times$ q$_1$},M label={p$_3$ $\times$ q$_3$},
r1 label=p$_1$, m1 label=q$_1$, r2 label=p$_2$,r3 label=p$_3$, m3 label=q$_3$}
\begin{tikzpicture}[N=8,r1=4,M=8,r3=4]
    \node[clos example with labels] {};
\end{tikzpicture}
```

Index