Prerequisites: basic \LaTeX commands and some LyX knowledge.
There are multiple ways to generate block diagrams in \LaTeX. Here are three most popular approaches.

**xyplot using `\xymatrix`**

This tool provides the most natural support for LyX.

Big picture: looking at the block diagram

\[
\xymatrix{ & u \ar[r] & P \ar[u] & y \ar[l] \ar[r] & C \ar[u] & }$

you can see the following composition:

- two framed blocks $P$ and $C$,
- four points defining the edges of the rectangular diagram,
- arrows and lines connecting different elements,
- comments like the negative sign, $u$, and $y$.

When you draw the diagram on a piece of paper, you might use a ruled notebook for better alignment of the elements. XY-plot does the work in exactly the same way. The notebook it uses is called XY-matrix.

Basic commands:

- **Outlining**: `\xymatrix` generates a matrix where you can fill each entry with items you want (in my case, boxed transfer functions, signals, systems, etc)
- **Making connections**: `\ar[pos]` places an arrow to [pos], where pos can take the values of: l(eft), r(ight), d(own), u(up)
  - For instance, `\xymatrix{B \ar[r] & A}` generates a two by one \text{XY}-matrix with the entries of $B$ and $A$. The command `\ar[r]` generates an arrow, starting from the first entry element $B$, pointing to the right, and ending on the second entry. Overall, the commands typesets $\xymatrix{B \ar[r] & A}$
  - Arrows can span multiple cells. For instance, `\ar[rr]` means “draw an arrow that points to the right and ends two entries away on the right”. As an example, `\xymatrix{B \ar[rr] & & A}` gives $\xymatrix{B \ar[rr] & & A}$
- **Framing a block**: `*[F]{contents}` puts contents into frames. The basic usage and common examples are summarized next.
default frame: $\xymatrix{\ast[F]{A}}$ gives

Usually the default frame is too tight and must be widened by prefixing with $+$ or $++$. Most commonly, I would use a command like $\xymatrix{\ast[++[F]{A}}$, which gives

double-line frame: $\xymatrix{\ast[++[F=]{A}}$ typesets

circular frame: $\xymatrix{\ast++[o][F]{A}}$ gives

Similar to the double-line rectangular frame, we can do $\xymatrix{\ast[++[o][F=]{A}}$, to get

• Specifying the arrow formats:
  - use @{...} after \ar to let \LaTeX know the line format of the arrows
    * Dashed arrows can be specified by \ar@{-->}. For instance, $\xymatrix{a\ar@{-->}[r] & b}$ gives $a \rightarrow \rightarrow b$
    * Dotted arrows can be specified by \ar@{..>}. E.g., $\xymatrix{a\ar@{..>}[r] & b}$ gives $a \ldots \rightarrow b$
    * Invisible arrows: this is useful for labeling certain graphs. By typing \ar@{}, we can specify null format for the arrow. For instance, $\xymatrix{A\ar@{}[rrrr]\ar[r] & B}$ gives

- use @() after \ar to specify the start and the end points of the arrow.
  * For instance, $\xymatrix{A\ar@[(d,u)][r] & B}$ gives

The command $\ar@[(d,u)][r]$ reads: “draw an arrow to reach the next matrix entry on the right (say element $B$), starting downwards and finishing at the upper side of $B$.”

* We can do more fancy stuff such as:

$\xymatrix{A\ar@{(d,d)}[r] & B}$,  
$\xymatrix{A\ar@{(dr,ul)}[r] & B}$,  
and $\xymatrix{A\ar@{(d,ul)}[r] & B}$, which give, respectively,
• Tips for using \textbf{\textsf{XY}}-matrix in LYX’s Math editor (press Ctrl-m or Ctrl-Shift-m and then type \texttt{\textbackslash xymatrix})

  – the braces: \{\} is the same, but \{\} should be entered by \texttt{\{[\} since by default LYX interprets \{\} as visible braces to display in math mode (e.g. \{A + (B^2 + A)|C\} + \texttt{A}).

  – shortcuts:
    * Adding a row in \textbf{\textsf{XY}}-matrix: Ctrl-Enter
    * Adding a column in \textbf{\textsf{XY}}-matrix: Alt-m c i
    * Horizontal and vertical scaling: How do we change the spacing between different elements in \textbf{\textsf{XY}}-matrix? In the tutorial “Using \textbf{\textsf{XY}}-pic in LYX,” H. Peter Gumm described using the following macros in the preamble of the LYX document (Layout > Document > Preamble):

      \begin{verbatim}
\newcommand{\xyR}{1}{% \xydef@\xymatrixrowsep@{#1} % end of \xyR \newcommand{\xyC}{1}{% \xydef@\xymatrixcolsep@{#1} % end of \xyC
      \end{verbatim}

      Place the cursor inside the \textbf{\textsf{XY}}-matrix, just before the first entry. Then enter \texttt{\xyR\{} and/or \texttt{\xyC\{} followed by the desired values of dimensions in the braces. The default spacing is \texttt{2pt}. As an example, see the difference between

      \begin{verbatim}
\xymatrix{
  A & B \\
  C & }
\end{verbatim}

      and

      \begin{verbatim}
\xyC{9pc}\xyR{2pc}\xymatrix{
  A & B \\
  C & }
\end{verbatim}

      Results:

      \begin{verbatim}
A \rightarrow B \quad \text{default}
\end{verbatim}

      \begin{verbatim}
A \rightarrow B \quad \text{horizontally stretched}
\end{verbatim}

  \begin{center}
  \begin{tikzpicture}
    \draw (0,0) -- (1,0) node[midway, above] {default};
    \draw (0,0) -- (0,1) node[midway, left] {C};
    \end{tikzpicture}
\end{center}

  \begin{center}
  \begin{tikzpicture}
    \draw (0,0) -- (1,0) node[midway, above] {horizontally stretched};
    \draw (0,0) -- (0,1) node[midway, left] {C};
    \end{tikzpicture}
\end{center}

  \begin{itemize}
    \item Placing comments on arrows: In

      \begin{verbatim}
A \ar[r]\sp>\{+\} & B
C \end{verbatim}

      we want to put a negative sign near the end of the arrow on the left side. We can do so using \texttt{\ar[u]\sp(0.9){-}}, which reads “draw an arrow that points up, and has a superscript (sp) \texttt{^\(-\)} located at 0.9 of the full length of the arrow.”

      It is pretty useful to use \texttt{\sp{}} to let \texttt{\LaTeX} figure out the spacing and automatically locate the arrow comments near the end location. For instance, \texttt{A \ar[r]\sp{}} \texttt{B typesets A \rightarrow B}.

      \footnote{The closing brace will be automatically supplied by \texttt{\LaTeX}.}
\end{itemize}
• Arrows passing under an element: use \texttt{\ar'[ur]'[uurr]} to go pass the upper right element and then reach the uurr element. For instance

\begin{center}
\begin{tikzpicture}
\node (model) at (0,0) {reference model};
\node (system) at (0,-2) {adjustable system};
\draw[->] (model) -- node[above]{$u(k)$} (system);
\draw[->] (system) -- node[above]{$\epsilon(k+1)$} (model);
\end{tikzpicture}
\end{center}

• Defining new arrow types: we can define a new arrow type \texttt{"|>"} by
\texttt{\newdir{|>}{%/4.5pt/@{|}*:(1,-.2)@^{>}*:(1,+.2)@_{>}}} and use it in arrow formats such as \texttt{\ar@[=|>]\[r\]} and \texttt{\ar@[{-|>]\[r\]}, to get

\begin{center}
\begin{aligned}
A & \rightarrow>B \\
A & \rightarrow>B
\end{aligned}
\end{center}

Other commands:

• Drawing a switch: \texttt{XYpic} supports also numerical axis locations. For instance, \texttt{\ar@[{-}(21,5)*+{}}\texttt{\sp{T}_{s}}} typesets a switch

\begin{center}
\begin{tikzpicture}
\node (T) at (0,0) {$T_s$};
\draw[->] (T) -- node[above]{$\circ$} (0,-1);
\end{tikzpicture}
\end{center}

• Arrows passing under: just insert a \texttt{'} between the targets on the path

\begin{center}
\texttt{\xymatrix{\circ\ar@{|-->}'[dr]^{a}'[rr]+D*{\bullet}^b[drrr]^{c} & \circ \ar@{|-->}'[dr]^{a}'[rr]+D*{\bullet}^b[drrr]^{c} & \circ \ar@{|-->}'[dr]^{a}'[rr]+D*{\bullet}^b[drrr]^{c} & \circ \ar@{|-->}'[dr]^{a}'[rr]+D*{\bullet}^b[drrr]^{c} \ar@{|-->}[r]}}
\end{center}

• Framing a group of objects:
\texttt{\xymatrix{\frac{2}{s+2} & b\restore\texttt{\save}'[dr]^{a}'[rr]+D*{\bullet}^b[drrr]^{c} \ar@{|-->}[r] & \save\frac{2}{s+2} & b\restore\texttt{\save}'[dr]^{a}'[rr]+D*{\bullet}^b[drrr]^{c} \ar@{|-->}[r]}}
Using these basic elements can generate more complex block diagrams such as

\[ \begin{align*}
0 &\rightarrow 1 \\
\phi(k) &\rightarrow \phi(k) \\
\phi(k) &\rightarrow \phi(k)
\end{align*} \]

and

\[ \begin{align*}
C(z) &\rightarrow P(z) \\
P(z) &\rightarrow \bar{r}(k) \\
\bar{r}(k) &\rightarrow \bar{e}(k)
\end{align*} \]
PSTricks and pst-sigsys

pst-sigsys stands for 'additional PSTricks for signal processing'. Below are some brief notes that hopefully illustrate the capabilities of the package. I hope to update more when I get the time.

- Remarks:
  - The lines and arrows generated by the package are more customizable compared to xyplot.
  - The package does not support pdflatex. To obtain the final pdf with correct graphs, one has to compile to dvi first then utilize the command ps2pdf.

- A basic example:

```latex
\begin{pspicture}(6,2)
\rput(0,1){\rnode{x}{$x[n]$}}
\psblock[framesize =0.75 0.5](2,1){a}{$H_1$}
\psblock[framesize =1.5 1](4,1){b}{$H_2$}
\rput(6,1){\rnode{y}{$y[n]$}}
\psset{style = Arrow }
\ncline[nodesepA =.15]{x}{a}
\ncline{a}{b}
\ncline[nodesepB =.15]{b}{y}
\end{pspicture}
```

typesets

$x[n] \rightarrow H_1 \rightarrow H_2 \rightarrow y[n]$

- More examples
Compared to pst-sigsys, tikz is sometimes more convenient since it supports pdflatex directly. It even supports \LaTeX\ instant preview, so that we can see the drawings within \LaTeX\ before compiling them.

To use the package, we will need to add \texttt{\usepackage{tikz}} and \texttt{\usepackage{pgfplots}} at the beginning of the document. Also, it is very useful to define some basic formats at the beginning of the document:

\begin{verbatim}
\tikzstyle{block} = [draw, rectangle, minimum height=2em, minimum width=4em]
%fill=blue!20
\tikzstyle{sum} = [draw, fill=blue!20, circle, node distance=1cm]
\tikzstyle{input} = [coordinate] \tikzstyle{output} = [coordinate]
\tikzstyle{pinstyle} = [pin edge={to-,thin,black}]
\end{verbatim}

The first example\footnote{Modified from an example in \url{http://www.texample.net/tikz/examples/control-system-principles/}} uses relative location for each elements:

\begin{verbatim}
\begin{tikzpicture}[auto, node distance=2cm,>=latex']
\node [input, name=input] {};
\node [sum, right of=input] (sum) {};
\node [block, right of=sum, node distance=3.5cm] (controller) {$C(z^{-1})$};
\node [block, right of=controller, pin={pinstyle}above:$d(k)$}, node distance=4cm] (system) {$P(z^{-1})$};
\draw [->] (controller) -- node[name=u] {$u(k)$} (system);
\node [output, right of=system] (output) {};
\node [block, below of=u] (measurements) {$F(z^{-1})$};
\draw [->] (input) -- node {$r(k)$} (sum);
\draw [->] (sum) -- node {$e(k)$} (controller);
\draw [->] (system) -- node [name=y] {$y(k)$} (output);
\draw [->] (y) |- (measurements);
\draw [->] (measurements) -| node[pos=0.99] {$-$} node [near end] {$y_m(k)$} (sum);
\end{tikzpicture}
\end{verbatim}

\begin{tikzpicture}[auto, node distance=2cm,>=latex']
\node [input, name=input] {};
\node [sum, right of=input] (sum) {};
\node [block, right of=sum, node distance=3.5cm] (controller) {$C(z^{-1})$};
\node [block, right of=controller, pin={pinstyle}above:$d(k)$}, node distance=4cm] (system) {$P(z^{-1})$};
\draw [->] (controller) -- node[name=u] {$u(k)$} (system);
\node [output, right of=system] (output) {};
\node [block, below of=u] (measurements) {$F(z^{-1})$};
\draw [->] (input) -- node {$r(k)$} (sum);
\draw [->] (sum) -- node {$e(k)$} (controller);
\draw [->] (system) -- node [name=y] {$y(k)$} (output);
\draw [->] (y) |- (measurements);
\draw [->] (measurements) -| node[pos=0.99] {$-$} node [near end] {$y_m(k)$} (sum);
\end{tikzpicture}
It might be more preferred to construct a matrix and put different nodes there first:

\begin{tikzpicture}[auto, node distance=2cm, >=latex']
\matrix[column sep = .75cm, row sep = .375cm]
{
\node (u) {$u$}; & \node [coordinate](d1){}; & \node [block](plant) {$P$}; & \node [coordinate](d2){}; & \node (y) {$y$}; & \\
& & & & \node [block](obs){Observer}; & \node (xhat) {$\hat{x}$};
};
\draw [->] (u) – (plant);
\draw [->] (plant) – (y);
\draw [->] (obs) – (xhat);
\draw [->] (d1) |- (obs.190);
\draw [->] (d2) |- (obs.170);
\end{tikzpicture}
\begin{tikzpicture}
\[
xscale = 1, \% \text{to scale horizontally everything but the text}
yscale = 1, \% \text{to scale vertically everything but the text}
\]
\% NODES DEFINITION
\matrix [row sep = .375cm, column sep = .75cm, ]
\{
\% —————————— row 1
\node (nInputv) [yshift = 0.25cm] {$v$}; &
\node (n22) [coordinate] {}; &
\node (nPhi) [block] \{\parbox[c]{1.2in}{Static Nonlinearity \ \ \ \ \ \ \ \ \ \ \ \ \phi(v,y)\}}; &
\node (nInputu) [above]{$u$}; &
\node (nSystem) [block] \{\parbox[c]{.85in}{Nonlinear \ \ \ Plant}\}; &
\node (n26) [coordinate, xshift = -0.2cm] {}; &
\node (nOutput) {$y$}; &
\}
& & & & & &
& & & & & &
& & & & & &
\%
\% —————————— row 2
\node (n11) [coordinate] {}; &
\node (n12) [coordinate, xshift = 0.2cm] {}; &
\node (n13) [coordinate] {}; &
\node (n14) [coordinate] {}; &
\node (n15) [coordinate] {}; &
\}
\%
\% ———— % PATHS
\draw [->] (nInputv) – (nPhi.172);
\draw [->] (nPhi) – (nSystem);
\draw [->] (nSystem) – (nOutput);
\draw [-] (n26) |- (n12);
\draw [-] (n12) |- (nPhi.188); \\
\% auxiliary nodes
\node [coordinate, xshift = 0.4cm, yshift = 1cm] (nAux1) at (n26) {};
\node [coordinate, xshift = 0.4cm, yshift = -1cm] (nAux2) at (n12) {};
\draw [dashed] (nAux1) -| (nAux2) -| (nAux1) node [above, pos = 0.38] {feedback linearized system};
\end{tikzpicture}
Tikz can also generate simple graphs such as

![Diagram of overlapping circles labeled A, B₀, and C.]

Several more detailed examples are provided next:

```latex
\begin{tikzpicture}
\begin{axis}[
grid=major,samples=30,mark=none]\addplot[blue,very thick,domain=-1:1]
x^3;\end{axis}\end{tikzpicture}
```

We can actually define a new function to avoid repeated writing the same codes:

```latex
\newcommand{\plotfun}[3][]{\begin{tikzpicture}
\begin{axis}[
width=#1,grid=major,samples=100,mark=none]\addplot[blue,very thick,domain=#3]
\x^{#2};\end{axis}\end{tikzpicture}}
```

Using now `\plotfun{cos(x)}{-180:180}` and `\plotfun{x^2}{-20:20}`, we can get
We can combine different functions. For instance, \( \texttt{plotfun\{(sin(x)*ln(x+1))/2\}\{0:100\}} \)
typesets

We can even use \texttt{rnd} to generate random numbers. For instance, \( \texttt{plotfun\{rnd\}\{-20:20\}} \)
typesets