icemaker

version = 0.7.4 (2020-08-04)

David Johns,
david.johns@icewire.ca

Contents

1 Introduction

1.1 About icemaker

The icemaker app is a command line app useful for 2 main functions: (1) generate a .svg file from an LTSpice .asc file, (2) generate a .tex file from a .prb file

The choice of whether a .svg or .tex file is generated is determined by the file extension related to the -export flag.

1.2 The need for LTSpice to svg

For a modest sized schematic, a nicely hand drawn schematic takes about 1 min to create. Unfortunately, using a general purpose drawing program to create the same schematic takes from 10 to 15 minutes even if a library of symbols are made in advance.

However, a good schematic editing tool can create a schematic in about 2 minutes. Unfortunately, schematic editors are well known to create schematics that are not suitable for publications in research papers or educational material.

The purpose of `icemaker` creating a .svg file is to let the user create an LTspice schematic and then use Icemaker to "beautify" the schematic so that the resulting output can be used in publications. If needed, the svg file can be further edited (or tweaked) using Inkscape.

Example usage:

`icemaker -export=example.svg -text=subscript` example.asc

1.3 The need for problem to tex

Latex is an excellent way to create educational material such as textbooks, examples, exams, and problem sets. The purpose of `icemaker` creating a .tex file is to allow one to create a numerical example (or problem) .prb file so that parameters can change and have the solution be automatically updated for that choice of parameters. In addition, `icemaker` has an equation engine that is used to quickly write the solution for an example (or problem).

Example usage:

```
icemaker -export=example.tex -random=false -sigDigits=4 example.prb
```

2 Installation and Setup

2.1 Setting PATH for command line access

You will need add to your command line PATH the directories where the executable code is for both Inkscape and Icemaker. This will allow Latex to automatically call Inkscape and Icemaker when building a Latex document.

Information on how to set your PATH is below

- Windows: adding to PATH in Windows
- MacOS: adding to PATH in MacOS
- Linux: adding to PATH in Linux

2.2 Add –shell-escape flag when calling Latex

The icemaker app is best used when integrated with Latex. For Latex to be able to run command line apps, Latex (and/or pdflatex) needs to be called with the flag "–shell-escape"

- Add –shell-escape flag when calling Latex (and/or pdflatex)
- For example, in texmaker, this flag can be added in preferences as

```
"/Library/TeX/texbin/latex" --shell-escape -interaction=nonstopmode %.tex"
"/Library/TeX/texbin/pdflatex" -synctex=1 --shell-escape -interaction=nonstopmode %.tex
```

2.3 Install Inkscape

The best way to include an svg file into Latex is by using Inkscape and calling it through the command line automatically within Latex.

- Install Inkscape
  You should be using Inkscape version 1.0 or above.
  (versions below 1.0 have issues with exporting for latex)
- Add the folder (that holds the executable inkscape code) to your PATH so that inkscape is easily found when called from the command line.
- Test that it works by running "inkscape –help" in any directory.

To include an svg file into latex, use the 2 following commands:

```
\immediate\write18{inkscape -D --export-filename=filename.pdf --export-latex filename.svg}
\input{filename.pdf_tex}
```

(these commands are within the Latex file and you need to include the flag "–shell-escape" when Latex is called as discussed above).

A svg/Latex example is in FigureSvg.zip which includes one svg figure
https://www.icewire.ca/downloads.html

- Test that the above svg/Latex example works on your system

2.4 Install LTSpice

- Install LTSpice

You can use LTSpice with its built in symbols but it is suggested you add an svg library to improve the schematic drawing experience.

- Install svg library for LTSpice
  https://www.icewire.ca/downloads.html

Install the symbols in a directory where LTspice can see them

- Windows: typically in "user"\Documents\LTspiceXVII\lib\sym\svg (svg is the new added directory)
- MacOS: typically in "user"/Library/ApplicationSupport/LTspice/lib/sym/svg (svg is the new added directory)

2.5 Install Icemaker

- Download the appropriate zip file for your operating system

  Windows (64 bit): icemakerWin64.zip

  Windows (32 bit): icemakerWin32.zip

  MacOS (64 bit): icemakerMacOS64.zip

  Linux (64 bit): icemakerLinux64.zip

  https://www.icewire.ca/downloads.html

- Unzip zip file
- Copy icemaker file to directory where you want it installed (typical directories shown below)

Windows: C:\Program Files\

MacOS/Linux: /usr/local/bin

- Test that it is installed correctly and the PATH has been updated: run the following command in another terminal window (MacOS/Linux) or another command prompt (Windows)

  icemaker –help

2.6 Test Overall Setup

- Download the example FigureAsc.zip
  https://www.icewire.ca/downloads.html
- Unzip and build the latex example: figureAsc.tex
- Delete all the files in the tmp directory and rebuild the latex example

In this example, one LTSpice figure is included into a latex document. The LTSpice figure is automatically converted to an svg file (using icemaker) and then the svg file is included (using inkscape). You should be able to delete all the files in the tmp directory and when run again, they should be re-generated automatically.

3 Running Icemaker

3.1 Command Line Options

The command line options for `icemaker` are the following:

- `-help`

  Print out help info

- `-version`

  Print out version info

- `-export=path/filename.ext`

  where the output should be placed

  path is the directory path that can include .. or . and subdirectories

  .ext is the file extension and should be either .svg or .tex

- `-symPath=path`

  where the LTSpice symbols are located (LTSpice to svg option)

  path is the directory path to the LTSpice symbols

- `-text=option`

  where option is one of ... (LTSpice to svg option)

  noChange: Leave all text as is (default setting)

  latex: instantiation names will be put in latex equations

  subscript: instantiation names will have subscripts and _{x1} will put x1 as a subscript

  symbol: input should be an LTSpice .asy symbol file and an svg output will be created for that symbol

- `-Tdots=option`

  where option is one of ... (LTSpice to svg option)

  true: Place dots on T wire connections (default setting)

  false: Only place dots on 4 wire connections

- `-random=option`

  where option is one of ... (problem to tex option)

  true: Parameters are randomized

  false: Parameters are the first values in the sets (default setting)

- `-sigDigits=value`

  where value is an integer that sets the number of significant digits in the output (default: 4) (problem to tex option)

3.2 Error Logging

When running Icemaker, error information is placed as comments at the beginning of the ouptput file (either the .tex or .svg file). If things do not work as you expect, check the error log information first as Latex error information can sometimes be misleading.

4 LTSpice to svg

When `icemaker` is called with the export option, `-export=filename.svg`, and the input is a `.asc` file, an svg file is created from the asc file. The benefit of the svg file is that the svg file is of publication quality. In addition, if required, the svg file can be opened with `inkscape` and modifications can be made (such as adding colours, shading, etc).

4.1 Example

Given an LTSpice .asc file as shown in Fig. 1(a), the resulting svg image is shown in Fig. 1(b). In this example, the `-text=subscript` option was used so that the instantiation labels for the elements (resistors, transistors, etc) all are shown with subscripts.

In this example, the command line used is

```
icemaker -export=example.svg -text=subscript example.asc
```

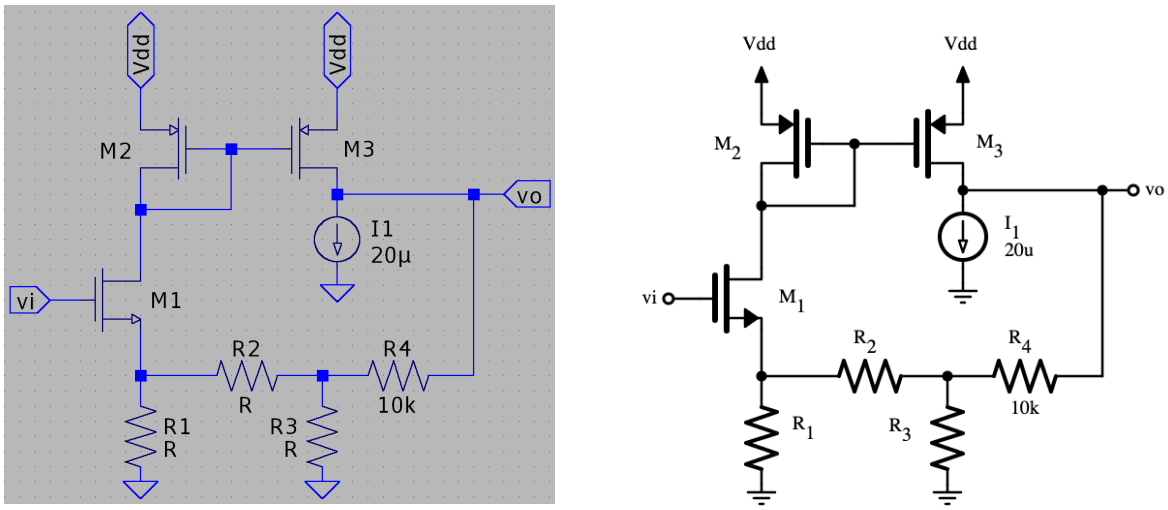
Grounds are changed as shown as are input/output and Vdd connections.

Power supplies should be created with a bi-direct port type in LTSpice while input/outputs should be created with either an input or output port type.

Tdots are shown at wire T junctions but can be eliminated with the -Tdots option. A dot is always shown when 4 wires are connected.

Wire crossings are shown with jumpovers as shown in Fig. 2

With the -text=latex option, all instantiation names are put in latex equations and the font will match equations with the rest of the latex document. Also, latex equations can be added to the LTSpice figure as comments.



(a) (b)

Figure 1: Example: (a) LTSpice .asc screen image, (b) .svg image

4.2 Creating your own LTSpice symbols

The default LTSpice symbols are already included in icemaker. However, if you decide to create new LTSpice symbols and want to use icemaker, you need to let icemaker know where the LTSpice symbols are located through the -symPath="path" flag. Also, any new symbols should NOT have the same names as existing LTSpice symbols.

Also, since LTSpice is limited in its drawing capability (no line thickness, colors, etc), you may wish to redefine the svg symbols associated with an LTSpice symbol. You can modify the svgDefn.svg file (currently in the svg directory) and place the modified file in the symPath directory (or in the svg directory placed inside the symPath directory).

Finally, for Windows users, the -symPath="path" flag needs to use \\ (double backslash) for separations between directories since Latex reserves a single \ for recognizing Latex commands.

5 Problem to tex

Problem to tex is a way to make problems have parameters that can change and the solution is re-calculated for that solution. In addition, the solution is easily written as equations which are then displayed as latex solutions. For this to work, icemaker has a built-in equation solver similar to Julia or Matlab.

5.1 Examples

This example is available at testBasic.zip

In this example, basic01.prb is a user generated problem file and contains the following text:

```
\runParam{x = [2, 3, 4, 5]}
\runParam{y = [6, 7, 8, 9] }
\runParam{k = [8]}
\question Given \val={x}, \val={y}, and \val={k} find $z = x^2+y-k$\\
\textbf{Solution}\\
\run(){z=x^2+y-k}\\
\hlite{\val={z}}
```

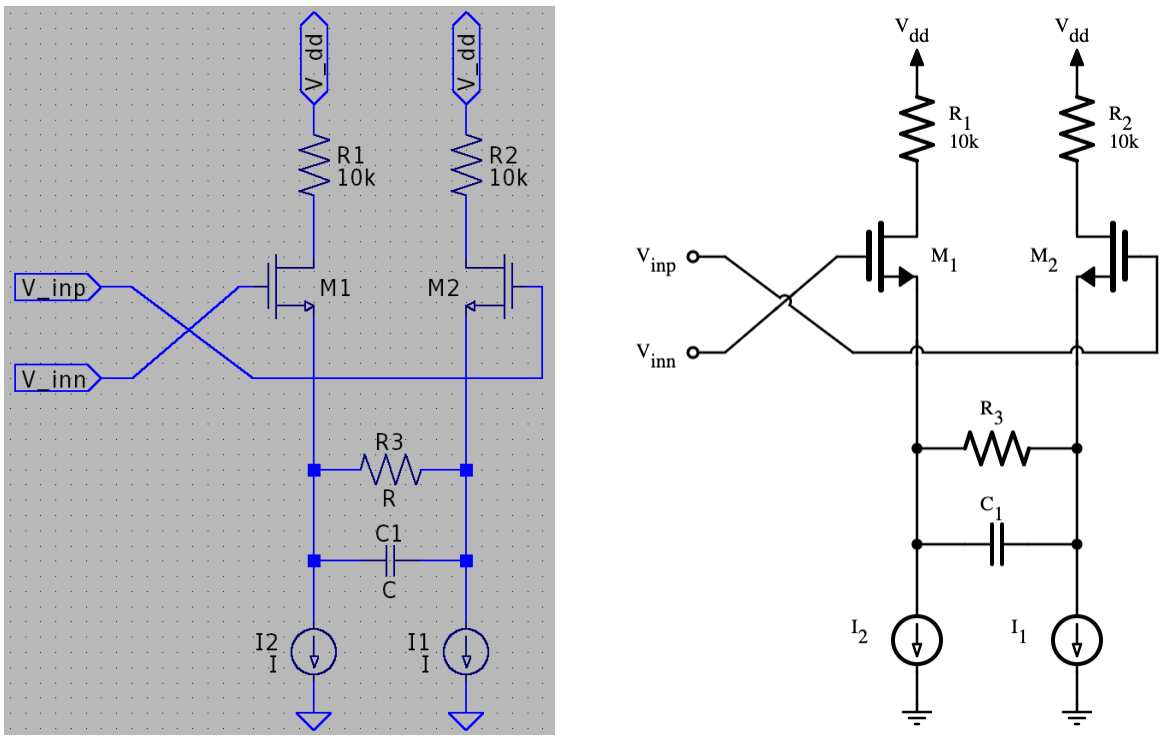


Figure 2: Wire crossing example: (a) LTSpice, (b) svg

For the above problem, x, y and k are all parameters where x is any one of 2,3,4 or 5 and y is any one of 6,7,8 or 9 and k is 8. For the solution, it is simply written as the equation and a built-in equation solver solves the equation and write it out in correct Latex form. The solution in this case is written as

```
\run(){z=x^2+y-k}
```

and when running with the following command

```
icemaker -export=example.tex example.prb
```

the following tex file is created ...

```
% Created with icemaker: version = 0.7.1 (2020-07-16)
\question Given \mbox{$x = \mbox{$2 \units{ }$}$}, \mbox{$y = \mbox{$6 \units{ }$}$},
and \mbox{$k = \mbox{$8 \units{ }$}$} find $z = x^2+y-k$\\
\textbf{Solution}\\
\mbox{$z=x^2+y-k  = (2)^2+(6)-(8)$}\\
\hlite{\mbox{$z = \mbox{$2 \units{ }$}$}}
```

The command \mbox is used so that inline equations can be used either within or outside other inline equations. The command \units is used to improve the font when displaying units (there are no units for this example).

In the same testBasic.zip file, the second example basic01.prb contains the following text:

```
\runParam{V_1 = [2, 3, 4, 5]# \paramUnits{V}}
\runParam{R_1 = [6, 7, 8, 9]# \paramUnits{k \Omega}}
```

```
\question Given \val={V_1} and \val={R_1}, find the current  $I_R = V_1/R_1$\\
\textbf{Solution}\\
\run(){I_R=V_1/R_1}\\
\hlite{\val={I_R}}
```

What is interesting here is that units can be assigned to parameters and the equation solver takes into account unit prefixes to generate the correct solution value prefix (i.e, f, p, n, μ, m, k, M, G, etc).

Also related to units, variables that start with the letters (small or capital letters) [V, R, I, C, L] have default units of [volts, ohms, amps, farads, henries]. However, the units for any variable can be changed using the \paramUnits{} command.

In this example, the above 2 problem files are included in a filed called basic.tex which contains the following text:

```
\documentclass[11pt]{exam}
\usepackage{import,xcolor}
\newcommand{\incProb}[1]{
    \immediate\write18{icemaker -export=Problems/tmp/#1.tex
    -random=false -sigDigits=4 Problems/#1.prb}
  \import{Problems/tmp/}{#1.tex}
}
\newcommand{\hlite}[1]{%
  \colorbox{yellow!50}{#1} }
\newcommand{\units}{\,\mathrm}

\begin{document}
\begin{questions}

\incProb{basic01}
\incProb{basic02}

\end{questions}
\end{document}
```

The tex file is using the exam class (for question numbering) and has 3 commands defined: \incProb; \hlite; \units.

When the above basic.tex is compiled as a text document, the following output is generated:

1. Given $x = 2$, $y = 6$, and $k = 8$ find $z = x^2 + y - k$

   Solution
   $z = x^2 + y - k = (2)^2 + (6) - (8)$
   $z = 2$

2. Given $V_1 = 2\,\mathrm{V}$ and $R_1 = 6\,\mathrm{k\Omega}$, find the current $I_R = V_1/R_1$

   Solution
   $I_R = V_1/R_1 = (2)/(6e3)$
   $I_R = 333.3\,\mu\mathrm{A}$

In this case, all default parameters are used but it is one line change to obtain random parameters and change the significant number of digits.

```
icemaker -export=Problems/tmp/#1.tex -random=true -sigDigits=6 Problems/#1.prb
```

When run using the above random flag, one example random case is the following: (each run will be random)

1. Given $x = 5$, $y = 8$, and $k = 8$ find $z = x^2 + y - k$

   Solution
   $z = x^2 + y - k = (5)^2 + (8) - (8)$
   $z = 25$

2. Given $V_1 = 3\,\text{V}$ and $R_1 = 7\,\text{k}\Omega$, find the current $I_R = V_1/R_1$

   Solution
   $I_R = V_1/R_1 = (3)/(7e3)$
   $I_R = 428.571\,\mu\text{A}$

5.2 Commands for .prb files

- \runParam{var = [x1, x2, x3, ...]# \paramUnits{units} \paramLatex{varLatex} }

  var will be a random selection from the set of x1, x2, x3, ...

  if random=false, the default value for var is the first element

  if \paramUnits{units} is present, then the units for that variable will be units

  if \paramLatex{varLatex} is present, then when printing out, var will be replaced with varLatex

- \run{expr}

  Evaluate the expr as well as print out the expr. The expr is an expression which is a variable (new or existing) = equation of existing variables

- \runSilent{expr}

  Evaluate the expr but DO NOT print out the expr

- \run(){expr}

  Evaluate the expr, print out the expr AND print out intermediate () expr for clarity

- \run={expr}

  Evaluate the expr, print out the expr AND print out the result for the expr

- \run()={expr}

  Evaluate the expr, print out the expr AND print out intermediate () expr AND print out the result

- \val{var}

  Print out the value of var with no units

- \valU{var}

  Print out the value of var with units

- \val={var}

  Print out var equals the value of var with units

5.3 Equation Solver

Problem to tex makes use of a built in equation solver. Equations are made similar to Julia or Matlab.

5.3.1 Functions

The functions currently known in icemaker are:

> abs, asin, asinh, acos, acosh, atan, atanh, ceil, cos, cosh, exp, floor, log, log10, round, sin, sinh, sqrt, tan, tanh

the above make use of the math package for golang.
In addition, extra functions are:

- atand(x)

  returns atan(x) but returns the value in degrees

- dB(x)

  returns 10*log10(x)

- dbV(x)

  returns 20*log10(x)

- parll(a,b)

  returns the numeric parallel value (returns $(1/a + 1/b)^{-1}$)

  the latex printout of this function is || to make it more readable

6 License

6.1 Files created by icemaker

All files created by icemaker are owned by the creators of the work (that is you) and/or by the original authors of the work in cases you use derivative works.

6.2 Software License