zDraw schematic editor

simple but full schematic editor. written entirely in Python (using pygame for display). it is “VIM” of schematic editors - has neither baggage nor hidden agenda. Schematic file is humanly readable and easily parseable. Has built in translator to verilog, spice and to neutral format. ready-made icons exist for logic gates , transistors and simple spice building blocks. Has helper scripts to create new icons out of verilog modules and also out of handmade icons.

basics

invoking xDraw.py opens graphics window. Commands are entered from orginal terminal window. Grpahics related commands are entered by punching a keyboard key with mouse over
required location. For example, striking letter "d" over nand2 instance will paint the instance red (selected) and then You can strike either 'e' (like end) to really delete or "q" (like quit) to cancel.
Many single letters are useful in graphics window.

letters on screen (coordinate under mouse):

letter	what it does
t	center display under mouse
m	start move instance under mouse, it will be moved to where "e" will be pressed.
a	add instance (from list cmd "add nand2 nor2 input output ...)
p	add param (from list cmd "params name=xxx size=n2,p3)
w	start wire from nearest pin
.	make bend in wire being added (it adds node and continues).
d	delete whatever under mouse.
e	end (end wire or end move or end deleting)
q	abort current action (moving,wiring,deleting,grouping)
c	copy instance under mouse
s	add new instance from copying (paste).
arrows	move instance under mouse a little bit (right,left,up,down)
r,R	rotate instance
f,F	flip instance
n	add instance of node (connection of wires), if it falls on wire - wire gets splitted at this point.

v g	add instance of VCC (v) or GND (g)
G	start group (end (char "e") will close the group and select all participants), to exit group use "q"
	Using mousedown on instances - adds them to group
	you can move the group around with arrows, or "s" to duplicate it at mouse position.
V	flip: fonts drawn vs fonts system
B	flip: show/hide bounding boxes. usefull for my debug, not so much for You.
T	add text geometry. the text comes from "add aaabbbbccc" typed in terminal window before.
D	go down in hierarchy. point the mouse to subcircuit instance.
U	get back up. return from "D"
C	take screen snapshot and save it in PNG file.
Q	(temporarily out of order) quit now
X	undo. new feature. use with caution.
Y	redo. new feature. use with caution.

commands in text window (original terminal):

note that typing unique leading substring of command name should work too.

command	**what it does**
V	flip fonts drawn vs fonts system
param size n2,p3 p15/4	add any kind param to params queue, will be used by "p" on screen.
name name1 name2 name3 ...	(sugar coating for param name) add name params to queue. use "p" on screen to connect them to instances.
add	empty "add"will print all options of known instances.
add nor2 nor3	add types to queue, will be used by "a" on screen. also used by "T" - free text placer.
load	load zdraw file
load	without filename will just list all loaded schematics and pictures.
load	will display schematic if it loaded, or try to load it from "ls'ed" directories.
new	create new schematic

delete	delete current schematic from editor
change	(no params) list loaded schematics,
change	change to another loaded schematic (similar to load)
save	save current schematic (or S from graphics window)
save	save current schematic to specified zdraw file, or directory. file name must be of current root or not mentioned.
Save	just like save, but also prints and dumps verilog
rename	will duplicate current detail and You start editing new version. load can be bring You to original one.
help	prints this
print	creates svg and ps images
dump	dump verilog
verilog	dump verilog
rtl	dump rtl verilog
spice	dump spice and picture (.zpic)
quit	just quit, same exit.
sys	execute shell command , like "open screenshot.ps" or something.
ls	list all schematics in this dir. default is ".". also the system remembers all listed drawings, so "load name" will search all "ls'ed" dirs.
zlib	will create a picture out of current detail.
zlib	will put the picture there.
=	way to change context variables. it can be "drawGrid = 0" to turn off the grid display, or "drawGrid = 1" to turn it back on.
	it can be "grid = 1" or "grid = 0.5" (default) to set placement grid. Notice that grid in dashed lines on screen is each four "grid" values.
	it can be "geom_text_size = 1" to change text size of free texts. (1 i just an example, 0.5 is also fine and so all "reasonable" numbers)
variables	will print all current values of context variables. Even the ones You nto supposed to mess with.
import .py	allows dynamic load of your code. To use for reports or special format dumps. Ask me about it. It needs more explaining.

import

example of import:
Suppose You wrote python file dox.py with contents, like this:

```
def run(wrds):
   print(wrds)
   print(Glbs.contexts)
```

to use it, in terminal type:

import dox.py // notice - You must specify .py in filename (unlike pure python)

dox aaa bbb // dox import must have run(wrds) definition. This function is run, wrds are splitted string activated it.

this, unless errors, wil lprint the activation words and also all context variables of Glbs (class of all things in zDraw).

** import dox.py ** can also be added to .zdrawrc

all commands can be abreviated to the shortest unique string

invocation:

zDraw.py [<schem_fname> <schem_fname> ...] [-do [dump|quit]]

-do quit will quit

-do print quit

-do verilog

-do " will do it

zDraw.py schem.py -do spice -do quit : will load schematic, dump it to spice and quit, without opening graphics.

-include : will execute lines in file, as if typed from terminal. dont put quit in it.

logging:

pymon.log0 : is log file, recording all comands and errors in Your session.

special notes:

if parameter name given with value "$inst" - it will display the instanse name.

.zdrawrc file

It can be put in your home dir (~) or current dir or both.

the format is simple. each line has two tokens.

first token is parameter name, second is parameter value.

Any system config param can be overwritten, but here are

Some usefull ones:

```
pics_lib  /home/global_user/zpic_lib
pics_lib  ~/myprivate_zpics
```

```
    import   /home/ilia/blablabla/my_private_dump.py

    backgroundColor white
    screen_proximity 10.0
    linewidth 1.0
    wire_color 0,255,0
    pic_color 255,255,0
    param_color 0,150,150
    geom_color 0,0,255
    instance_color 150,150,0
    pic_text_size 0.15
    param_text_size 0
    geom_text_size 1.0
    grid 0.5
    shyParams name,size    : parameters that show only the value (and not param=value)
    lineWidth 1.5
    vectorTextWidth 1.0

    wire_direction [0,1]   : do we put small circle on wires end (as indication of the
direction).
```