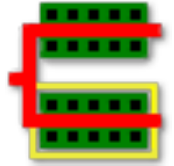


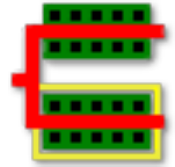
Glade

Peardrop Design Systems
7th July 2021

Agenda



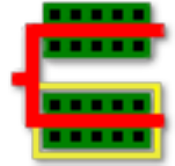
1. Symbol Creation
2. Schematics
3. Simulation
4. Schematic Driven Layout



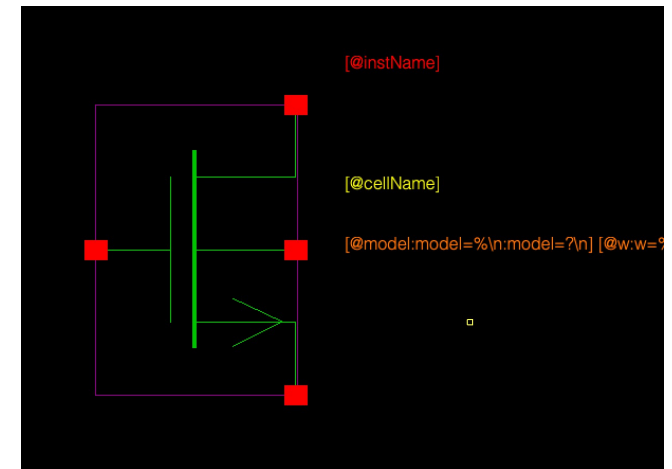
1. Symbol Creation

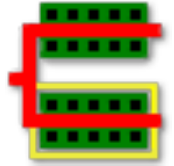
- Schematics consist of instances of symbols, wires and pins.
- Symbols are created and edited in the symbol editor
 - cellViews with a viewType of 'symbol' invoke the symbol editor menus.
- Symbols and schematics have a different grid to layout views
 - Traditionally, schematics use inches, with subdivisions like eights, sixteenths
 - Symbol and schematic views have 160 dbu/user unit and a user unit of inches
 - Allows resolution of $1/10^{\text{th}}$ of $1/16^{\text{th}}$ inch
 - Usually set snap grid to 0.0625, major grid to 8, minor grid 0.125
 - Important to maintain consistent grid between schematics & symbols!

Anatomy of a Symbol



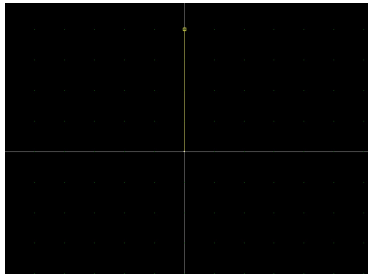
- Symbols have:
 - Shapes
 - Pins
 - Pins have net info e.g G, S, D, B
 - A bounding box
 - Used for selection and when adding wires
 - Labels
 - For showing info in schematics, e.g. instName, cellName
 - Properties
 - To control the netlister



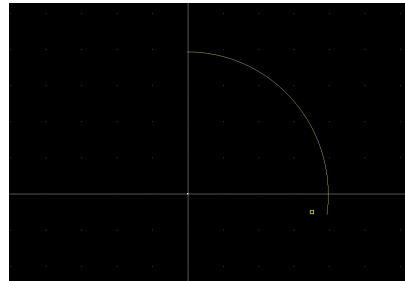


Symbol Shapes

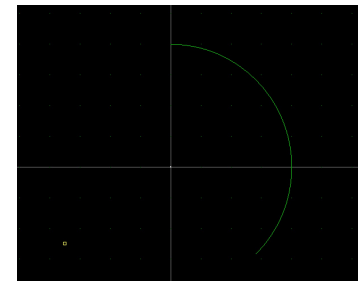
- Symbol outlines created on the 'device' layer
 - arc
 - ellipse/circle
 - rectangle/polygon
 - line



Enter 1st point of arc, then
2nd

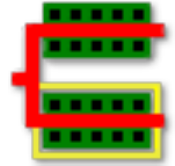


Move cursor to define arc
angle

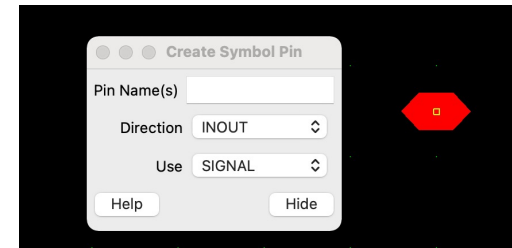
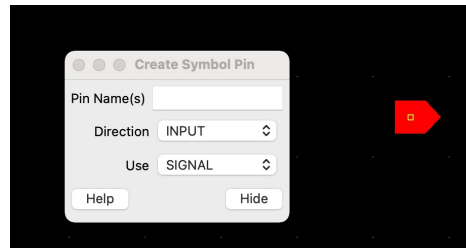
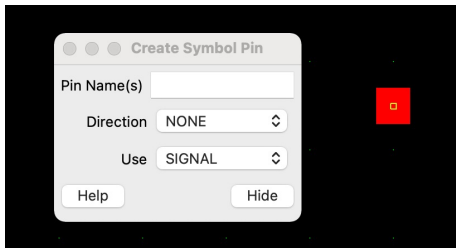


Cursor 3rd point defines
arc.

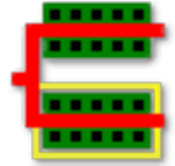
Symbol Pins



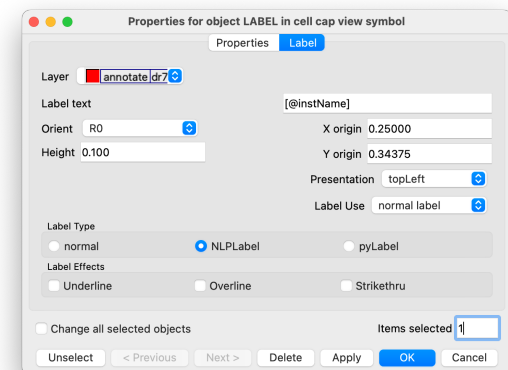
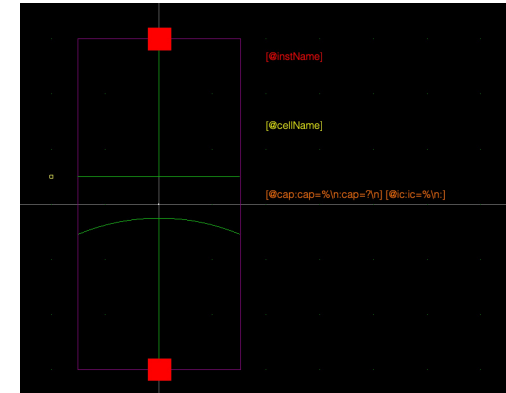
- Pins on pin layer
 - Rectangles or polygons
 - Plus net /pin info
 - Create Pin command creates the shape and net/pin info automatically
- Pins must be centered on grid
 - Failure to do so will make it hard (or impossible) to wire to pins



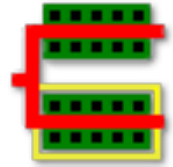
Symbol Labels



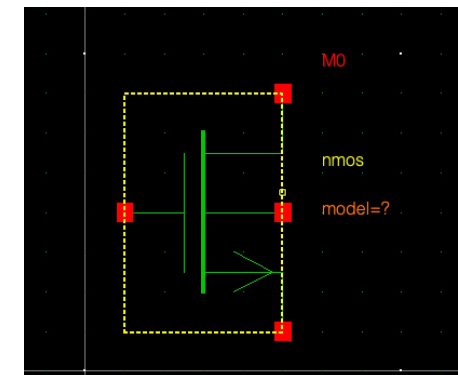
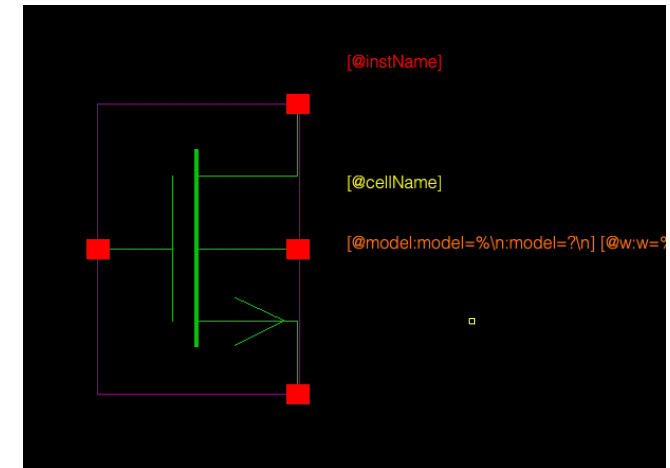
- Labels
 - Simple format for labels to display e.g. cellName, instName
 - Label type 'normal'
 - Like a layout label
 - Label type 'NLPLabel'
 - Label will be evaluated at display time according to NLP syntax rules (covered later).
 - Label type 'pyLabel'
 - Label will be evaluated as Python code at display time.

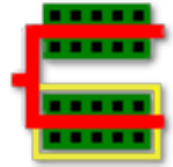


Symbol Boundary



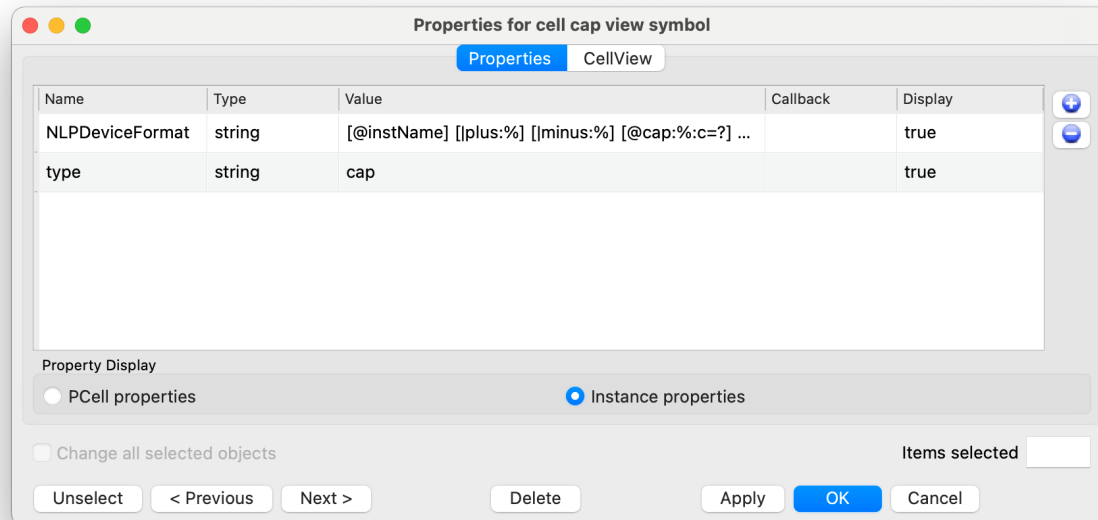
- Bounding box on 'boundary' layer
 - Used by selection and dynamic highlight
 - Used by schematic router to define blockage area
- Pins should be accessible from outside boundary

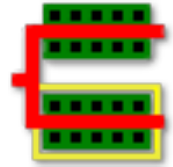




Symbol Properties

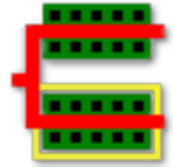
- NLPDeviceFormat
 - Controls how symbol instance is netlisted
- type
 - Used by e.g. schematics for Spice instance names





NLP Syntax

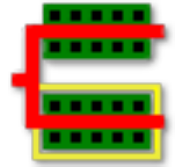
- NLP (netlister property) labels or properties
 - Labels are used to display attributes of a symbol in a schematic
 - [`@instName`] – evaluates to `instName` attribute
 - [`@cellName`] – evaluates to `cellName` attribute
 - [`@viewName`] – evaluates to `viewName` attribute
 - [`@libName`] – evaluates to `libName` attribute
 - [`@elementNum`] – evaluates to the element number (e.g. 0 for M0)
 - [`@<propertyName>`] – evaluates to the named property value
 - Searches instance first, then inst master.
 - [`|pinName:%s`] – evaluates to the named pin's net name
- NLP Expression:
 - [`@<propName>:<prefix>%<suffix>:<defaultValue>`]
 - `@<propName>` means search for this property name
 - If found, print the `<prefix>`, the property value, then the `<suffix>`
 - If not found, print the `<defaultValue>`
- NLP Expressions can be concatenated together with strings
 - Special character can be added
 - `\[` for left bracket
 - `\]` for right bracket
 - `\s` for space
 - `\n` for newline



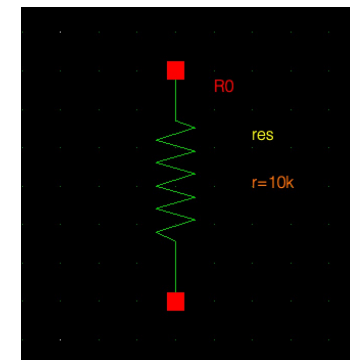
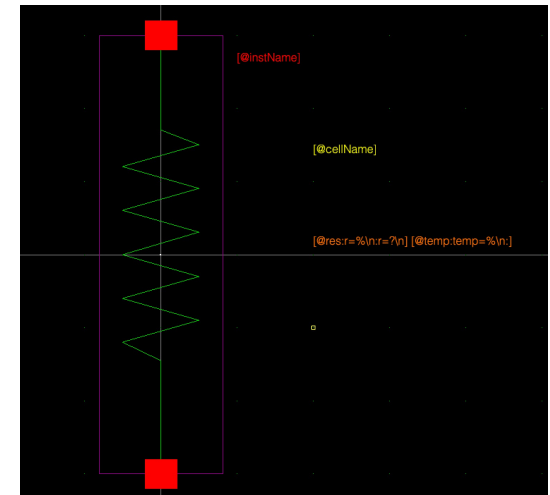
NLP Examples

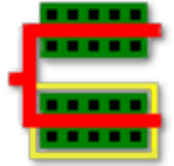
- NLP Label
 - [`@w:width=%u:width=2u`]
 - If the property 'w' exists, display 'w=<value>u'
 - If it doesn't exist, display 'width=2u'
- NLPDeviceFormat property on a symbol:
 - [`@instName`] [`|D:%`] [`|G:%`] [`|S:%`] [`|B:%`] [`@modelName`] [`@w:w=%`][`@l:l=%`]
 - Netlists as e.g.
 - M1 out in vdd vdd pmos w=1.0e-7 l=4.5e-8

Symbol Example



- Resistor with 2 pins plus and minus
- Symbol labels and properties
 - Labels for instName, cellName, properties
 - `[@instName]`
 - `[@cellName]`
 - `[@res:r=%\n:r=?\n] [@temp:temp=%\n:]`
 - Note how these are displayed when placed in schematic with property `r` set to 10k
 - NLPDeviceFormat property
 - `[@instName] [|plus:%] [|minus:%] [@res:%:r=?] [@temp:temp=%:]`
 - Using this property, resistor would be netlisted as:
 - `R0 netA netB 10k`

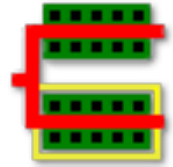




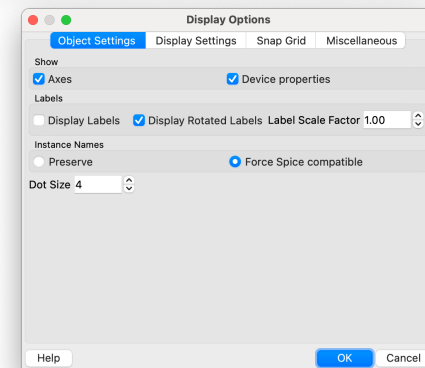
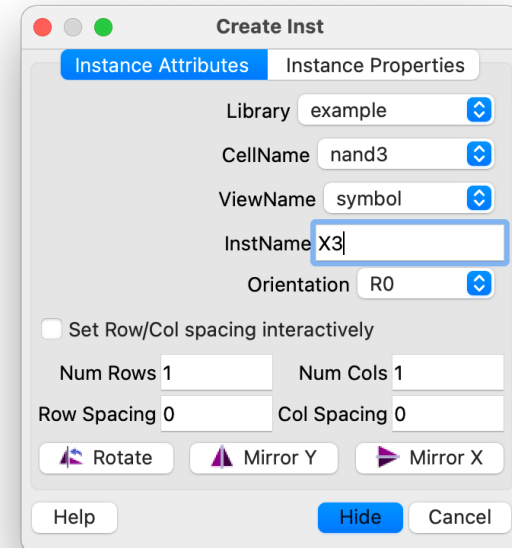
2. Schematics

- Schematics consist of instances of symbols, wires, pins and labels.
- Schematics are created and edited in the schematic editor
 - cellViews with a viewType of 'schematic' invoke the schematic editor menus.
- Schematics have connectivity
 - Check command extracts connectivity and checks for errors
 - Net names traced from IO pins or labels
- Schematics can be hierarchical
 - Symbol masters can have schematics
 - Netlister handles hierarchy
 - Global vs local nets
- Netlisting

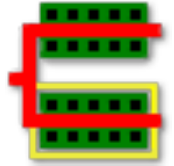
Instance creation



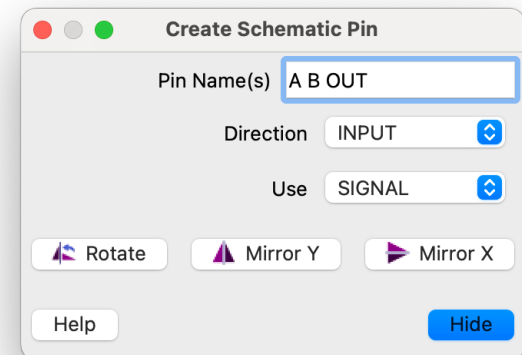
- Instance placement similar to layout
 - Change properties e.g. resistor value
 - Instances can be vectored by instance name, e.g. M0[0:7]
- Instance name can either be simple I0...In as layout
- Or using a spice-compatible prefix
 - M0...Mn
 - R0...Rn
 - C0...Cn
- Naming set in Display Options dialog.



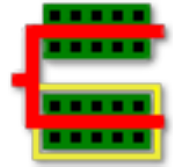
Pin Creation



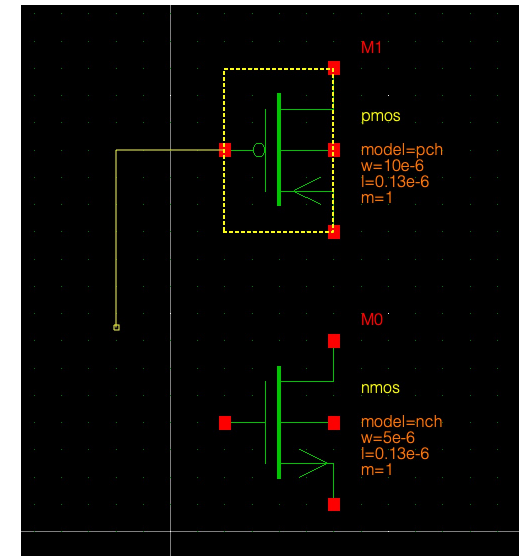
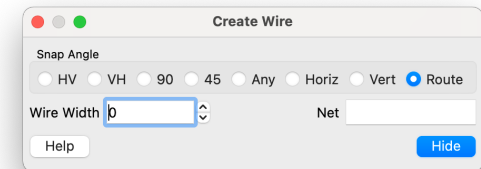
- Pin placement
 - Pins are instances
 - The pin name is a property on the instance (pinName)
 - Pins can also be vectored (bus pins)

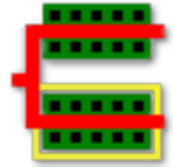


Wiring a schematic



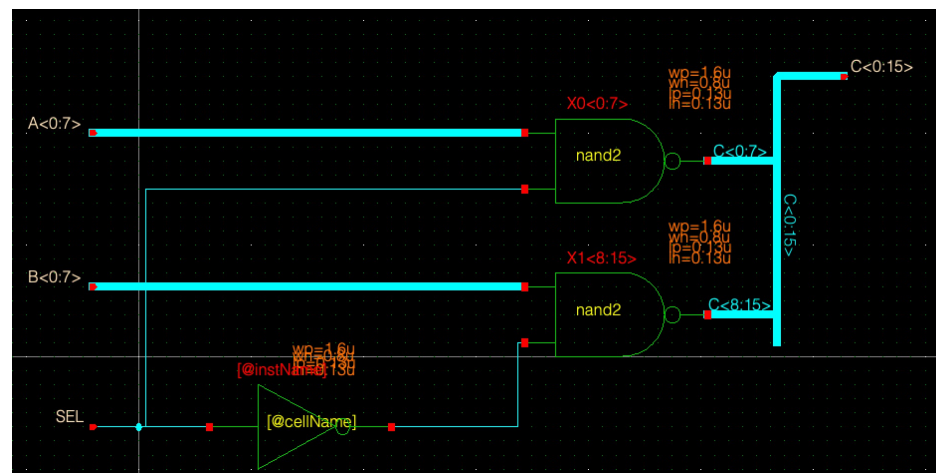
- Wiring
 - Wires connect pins and instance pins.
 - Wires can be routed manually, autorouted or a mix of both
 - Set snap mode
 - HV – horizontal then vertical
 - VH – vertical then horizontal
 - 90 – Manhattan
 - 45 – diagonal
 - Any
 - Horiz – only horizontal
 - Vert – only vertical
 - Route – point to point router
 - Start on pin/inst pin/wire
 - End on pin/inst pin/wire
 - Wires can join using a solder dot



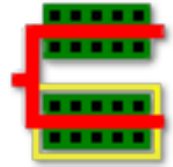


Bus wiring

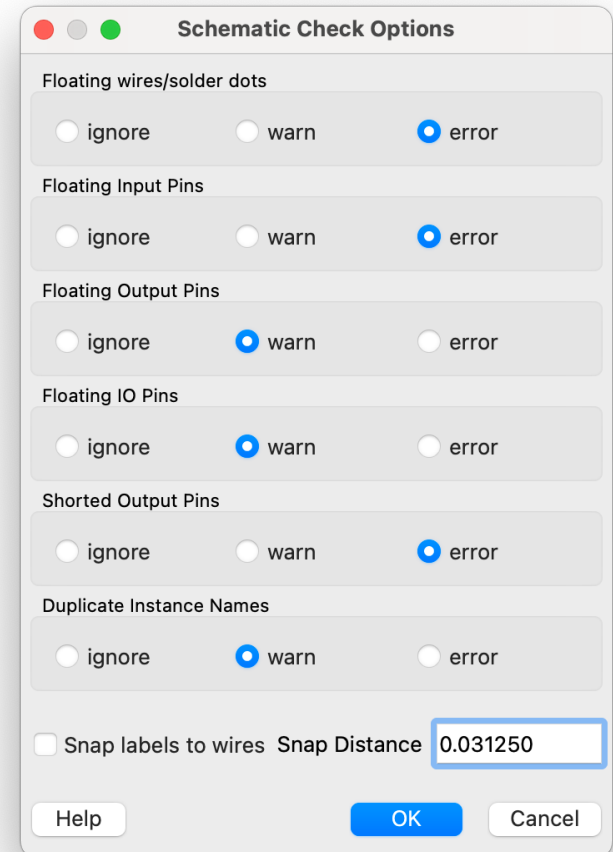
- Buses
 - Drawn as a thick line
 - Instances, pins and wires have bus notation
 - `<start_bit, stop_bit, step_num>`
 - Step_num is optional
 - E.g. `data<0:7>`, `addr<31:0>`
 - Buses can be tapped
 - Bus widths are checked by Check cmd



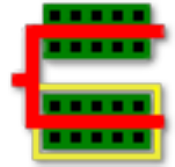
Checking



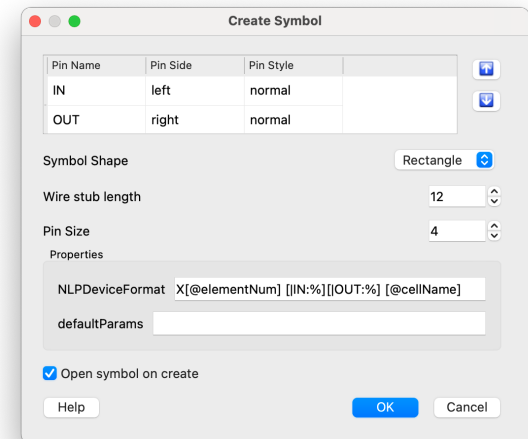
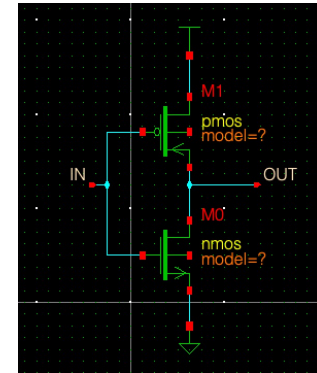
- Before a schematic can be netlisted, it must be checked (and usually saved).
 - Check->Check CellView or File->Check&Save
 - This extracts and checks the schematic connectivity
 - Checks are controlled by Check->Check Options... dialog
 - Ignore means don't care about this check
 - Warn issues warning but continues
 - Error will not allow the schematic to be marked as checked..



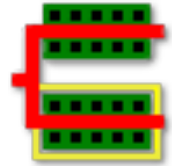
Create CellView



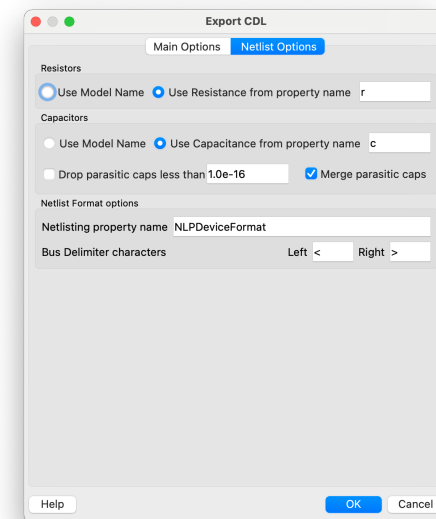
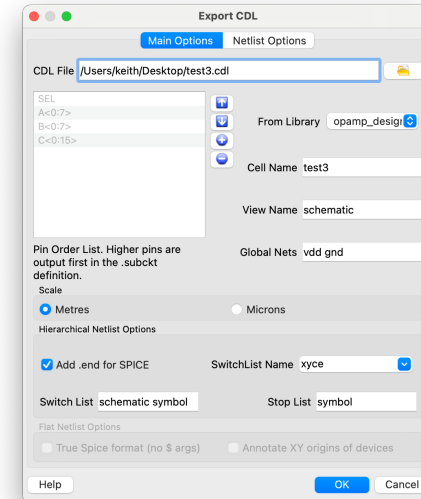
- Create a symbol from a schematic
- Simplifies task of creating symbols
 - Set pin side and style
 - Set symbol shape e.g triangle
 - Length of wires from body to pin
 - Pin size
 - NLPDeviceFormat property
 - defaultParams for adding default parameters



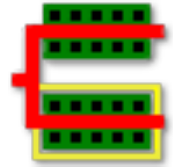
Netlisting



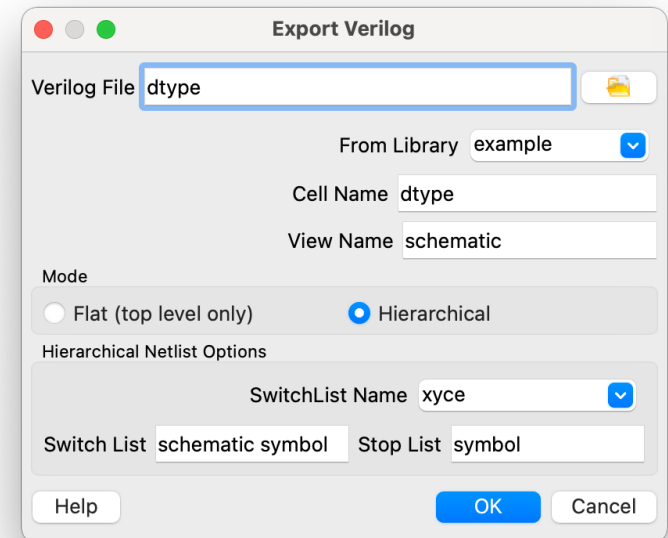
- Netlist to CDL (Spice style syntax)
 - Hierarchical netlisting for schematics
 - Supports global nets
 - Hierarchy switch and stop lists
 - Flat netlisting for extracted views
 - Netlist mode determined by viewType
- Netlister is instance-based
 - i.e. writes out every instance found in cellView
 - Ignores instances with property
 - nlAction = ignore
- Netlister looks for NLPDeviceFormat property on symbol to determine how to write instances in netlist
 - Flat netlister will look at 'type' property if no NLPDeviceFormat

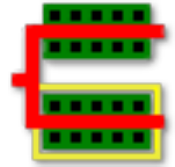


Netlisting



- Netlist to Verilog
- Hierarchical or flat (top level)
- Supports switch/stop lists
- Supports gate level (structural) Verilog only

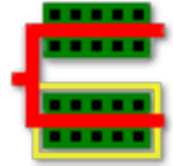




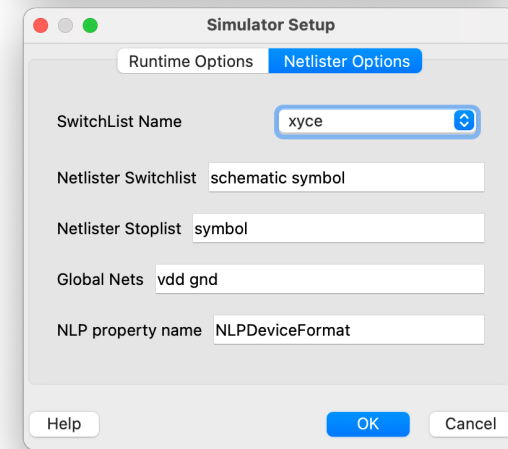
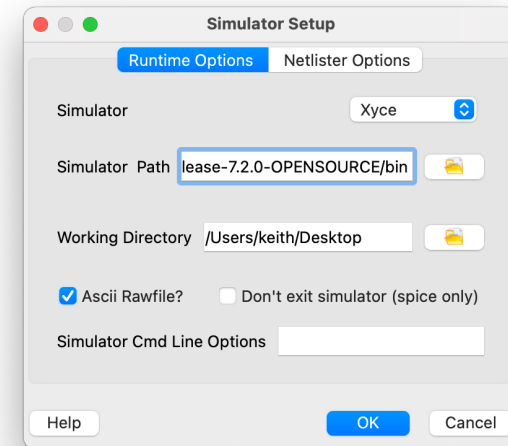
3. Simulation

- Simulation setup
 - Spice3, Xyce...
- Probing
- Simulation analysis types
 - DC OP
 - DC
 - AC
 - Transient
- Waveform display

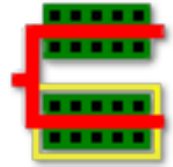
Simulator setup



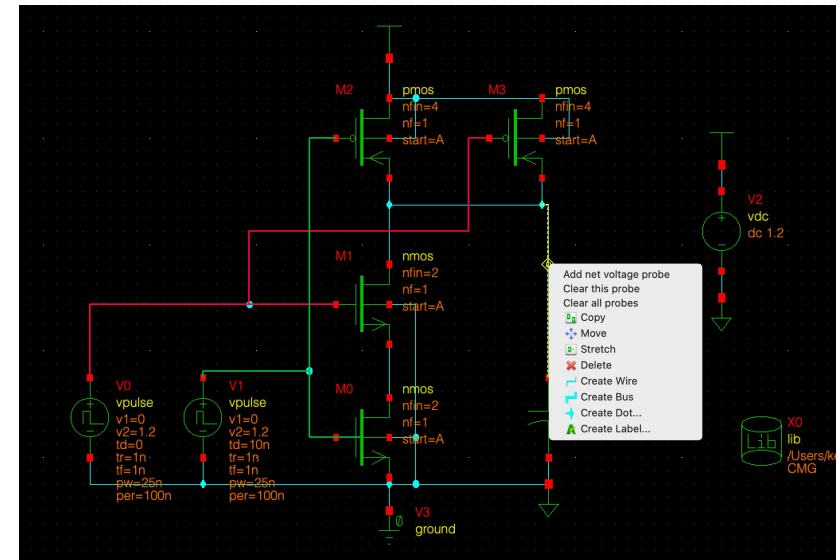
- Glade currently supports Spice3-like simulators
 - Xyce
 - Spice3f5
- Main requirement is for the simulator to write Rawfiles (either ascii or binary)
- Netlist syntax can be customized with NLP properties on symbols.
- Simulate->Setup
 - Simulator options
 - Netlisting options



Schematic probing

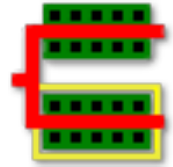


- Set nets to plot in schematic
 - RMB click displays popup menu
 - Adds a probe to the probe window
- Probe window
 - Add/remove probes
 - Edit probe expressions
 - Change probe colour/linestyle/linewidth
- Don't have to use probes
 - All voltages/currents saved to rawfile
 - Plot window can choose them

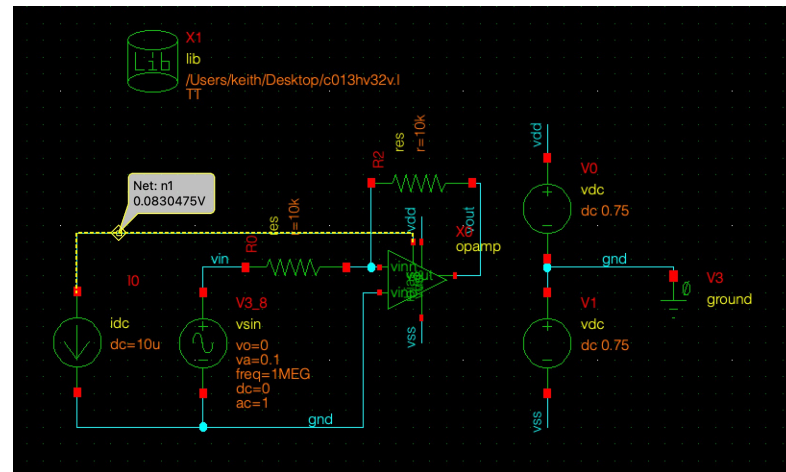
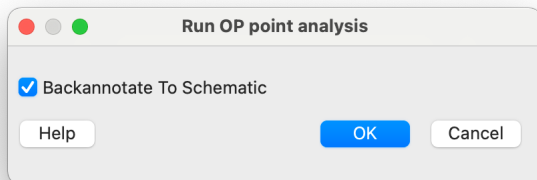


Probe Window			
Expression	Colour	Linestyle	Linewidth
V(n1)	#e6194b		2
V(n2)	#3cb44b		2
V(n0)	#ffe119		2

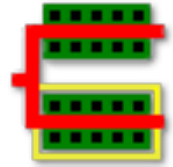
DC OP



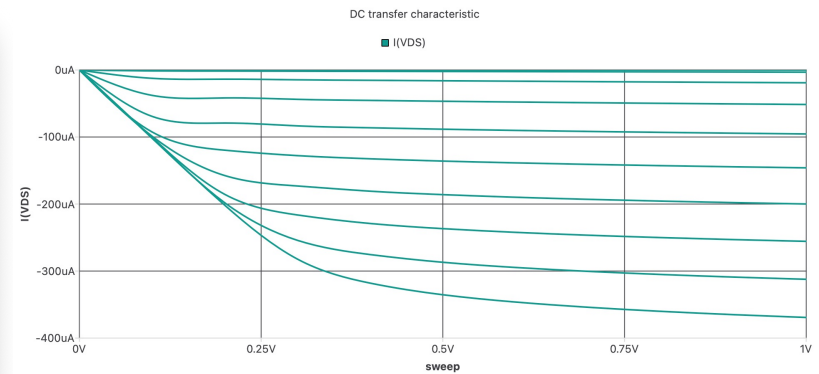
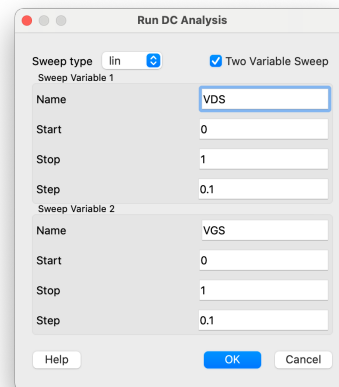
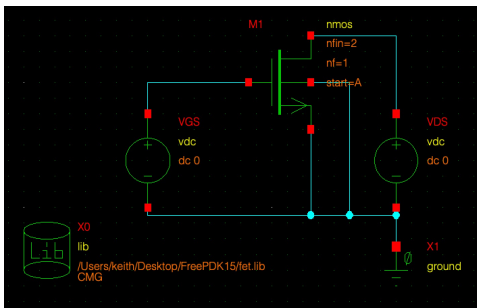
- Runs a DC operating point analysis
 - Optionally backannotates voltages to schematic
 - Shown using tooltips



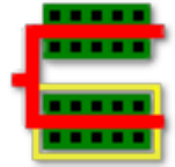
DC analysis



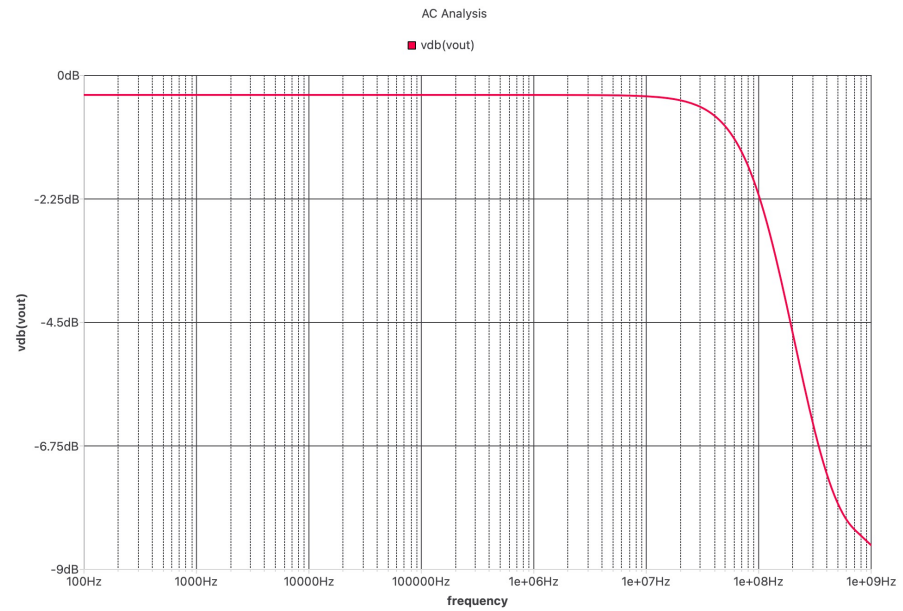
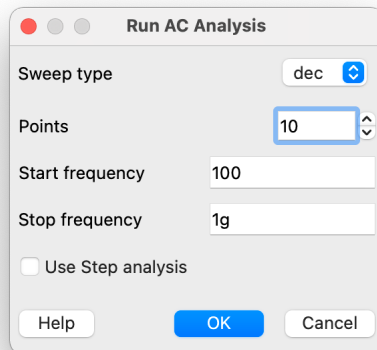
- Sweep 1 or more sources
 - E.g. V_{ds} and V_{gs} , plot I_{ds}



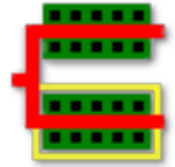
AC analysis



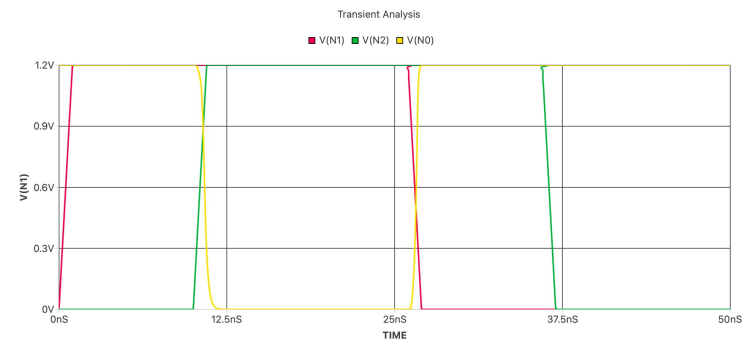
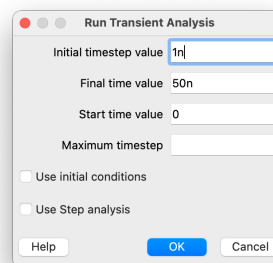
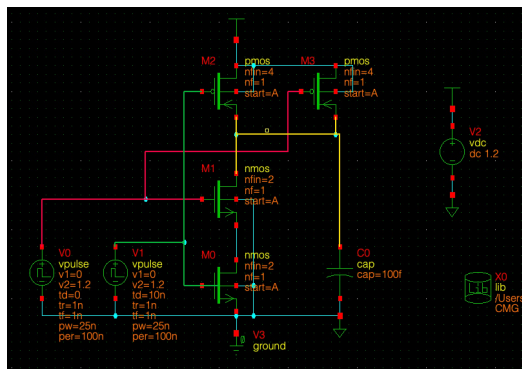
- Runs an AC sweep from start to stop frequency.



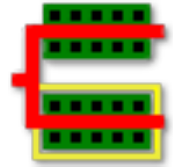
Transient analysis



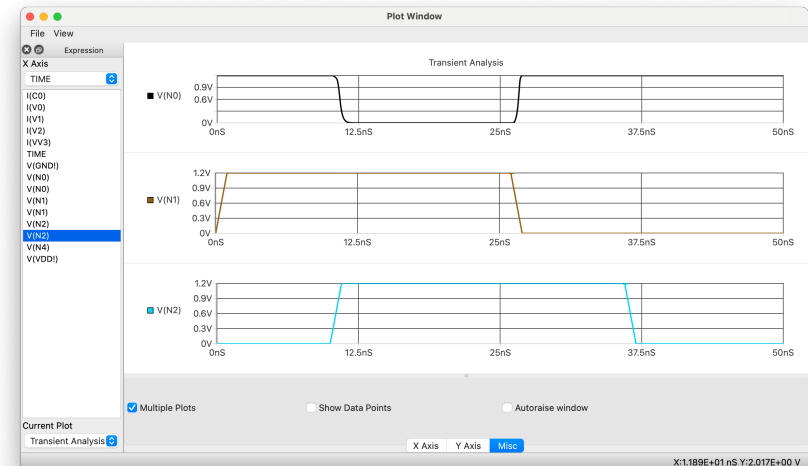
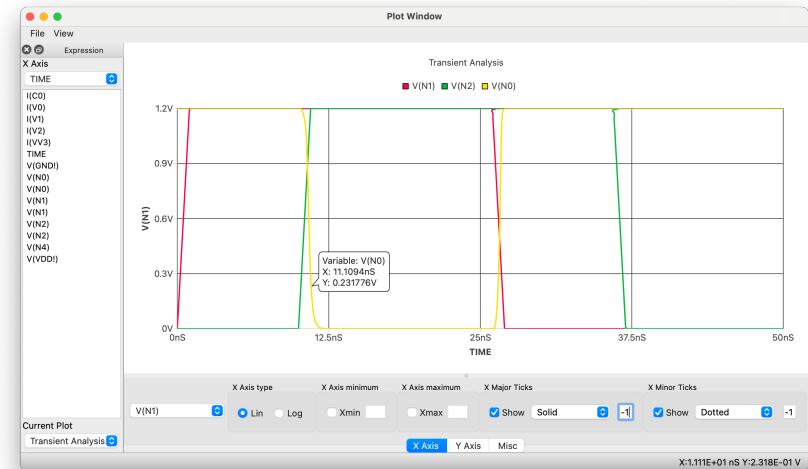
- Runs a transient simulation showing voltage/current vs time.



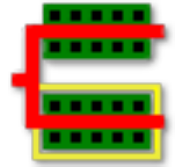
Plot window



- Plot voltages or currents vs. time/freq/each other
- Single plot or multiple plots
- Tooltips showing X/Y values
 - LMB click can pin tooltip
- Panning, zooming
- Set axes, grids

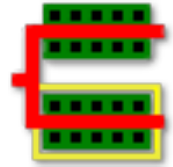


4. Schematic Driven Layout

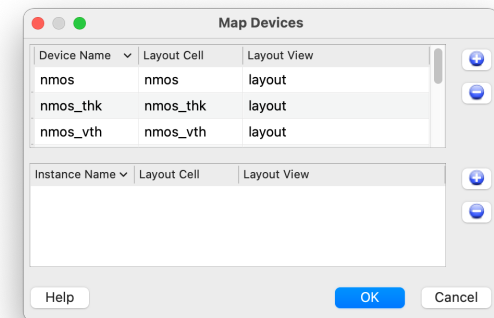
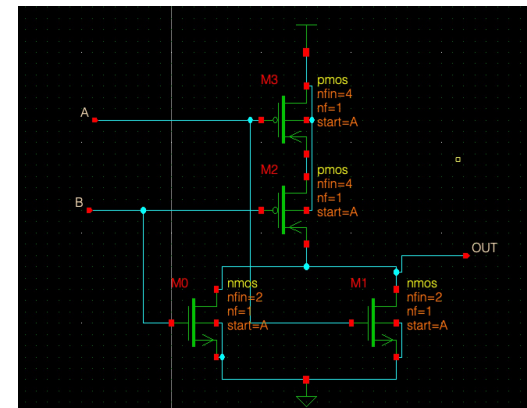


- Traditional approach didn't link schematics to layout
 - Error prone
- Glade can use schematics to drive layout
 - Connectivity driven
 - Pcells automatically sized according to schematic parameters
- Benefits
 - Faster layout
 - Netlist driven so connectivity and parameter correct

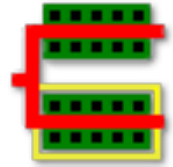
Requirements



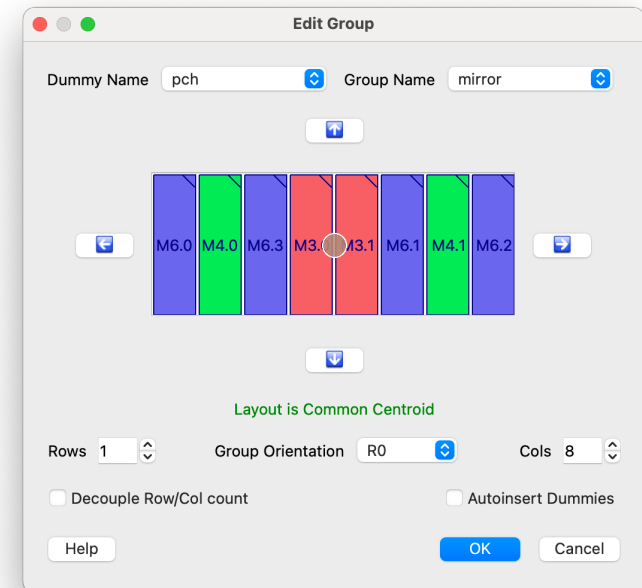
- SDL creates layout Pcells based on schematic symbols
 - Each symbol needs Pcell
- Schematic symbol names do not have to match layout Pcell names
 - Map Devices dialog can map schematic device name to layout Pcell name.



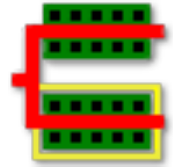
Groups



- Devices can be made into groups
 - These are placement patterns for matching
 - Various commands for manipulating groups
 - Create Group
 - Add to Group
 - Rename Group
 - Remove from Group
 - Delete Group
 - Edit Group
 - Typically use m factor devices for common centroid patterns



Create Layout



- Create Layout cmd
 - Sets target layout cellView
 - Sets mfactor insts vs. flattened insts
 - Sets utilization and scale factor
 - Optional Width and Height
 - Placement
 - Schematic – uses schematic cords
 - Area – stacks them to right of cell area
 - Group – uses group info
 - Pin placement side/width/layer

Create Layout

Library Name: FreePDK15

Cell Name: nor2

View Name: layout

☒ Create m factor instances

Scale Factor: 5.00

Utilisation %: 60.0

Width: Height:

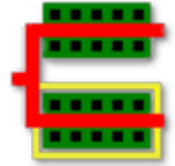
Placement method

☐ Schematic ☐ Area ☒ Group

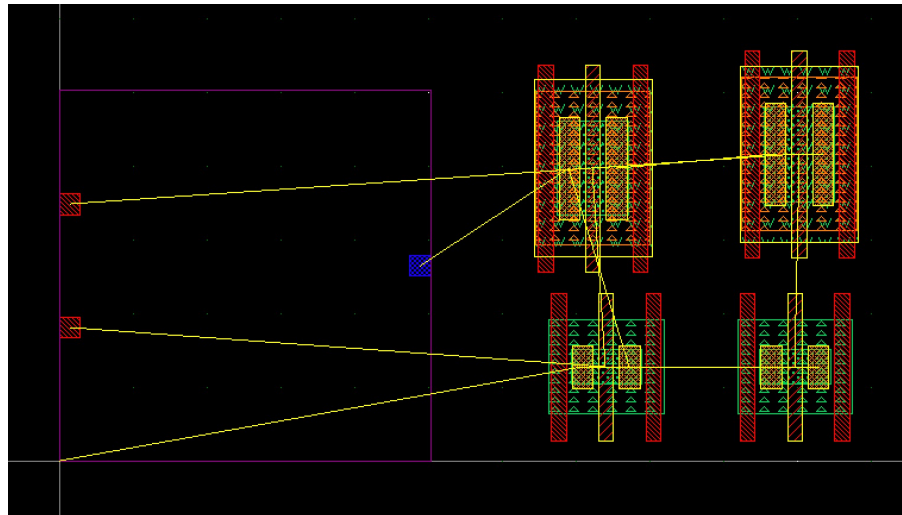
Pin Name	Pin Width	Pin Side	Pin Layer
B	0.028	left	GATEA dwg
OUT	0.028	right	M1A dwg
A	0.028	left	GATEA dwg

Help OK Cancel

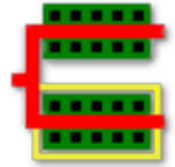
Create Layout



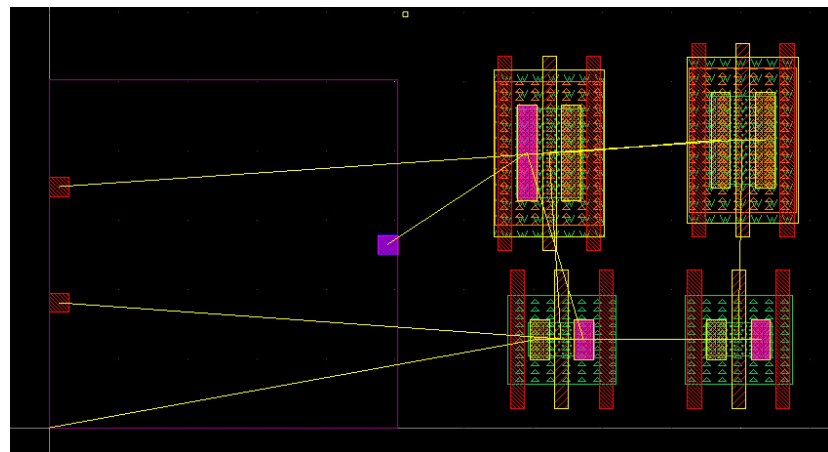
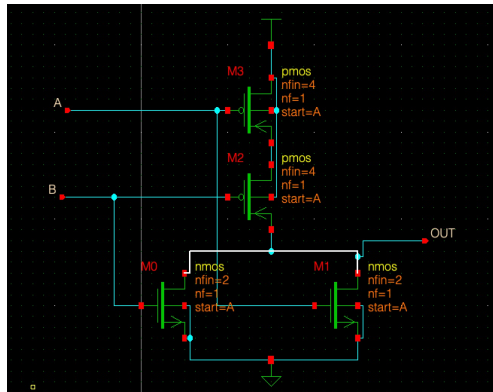
- Layout created has connectivity
 - Selection Options -> Show Connectivity displays flightlines
 - Assists placing devices



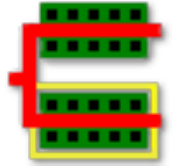
Cross probing



- Cross probe between schematic and layout
 - Nets
 - Instances
- Subsequently use 'Link to Layout' cmd to link schematic and layout views for cross probing.



Labs



- Create a inverter schematic
- Create a symbol for it
- Create a testbench for simulation
- Run simulation
- Create layout from the schematic