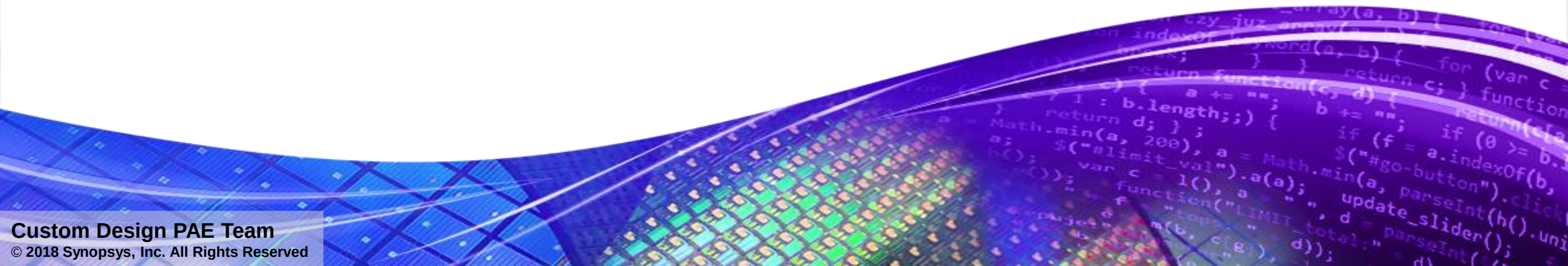
SYNOPSYS®
Silicon to Software™

Custom Compiler

Schematic Editor (SE)

Schematic Entry

O-2018.09

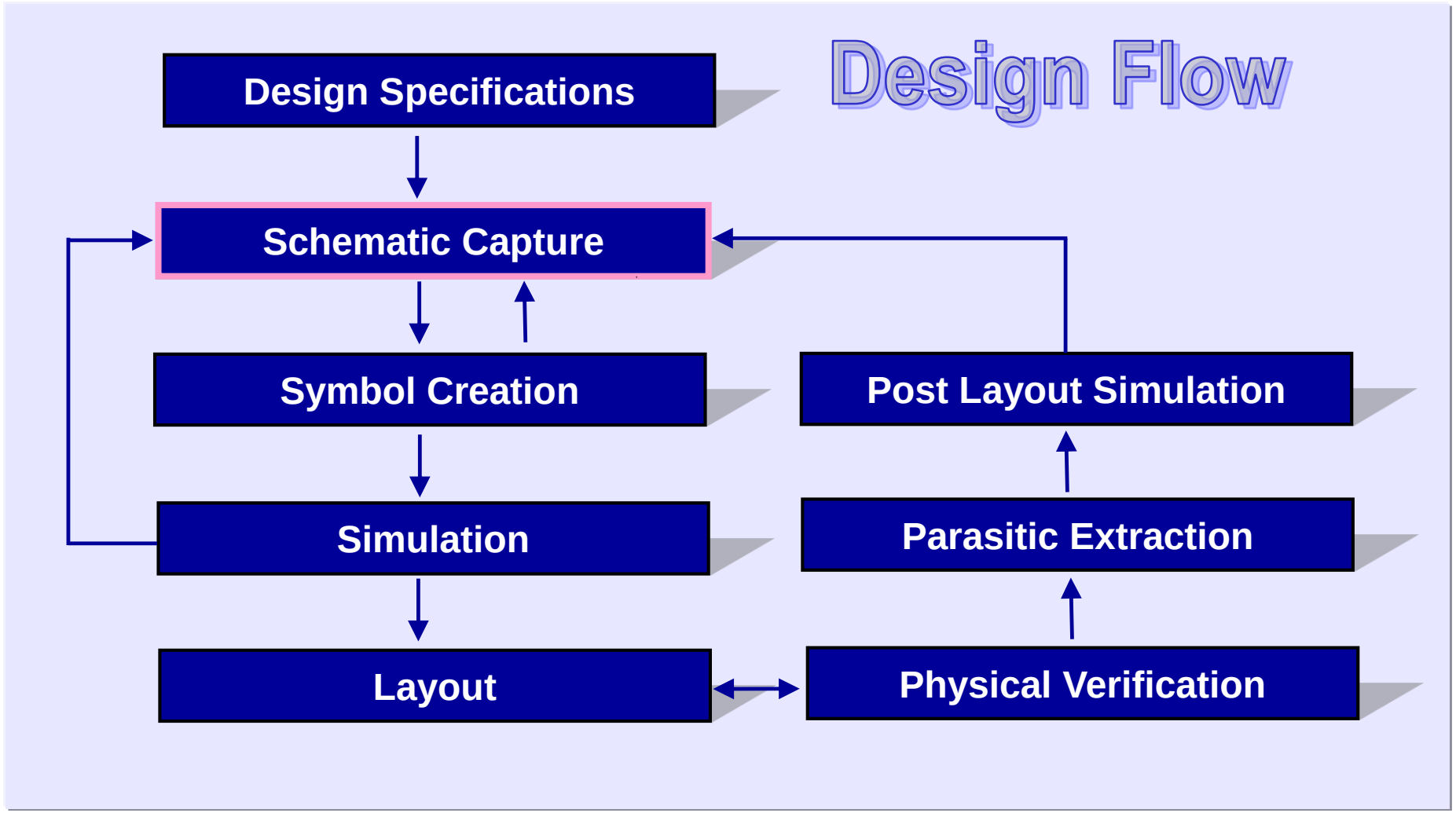


Custom Design PAE Team

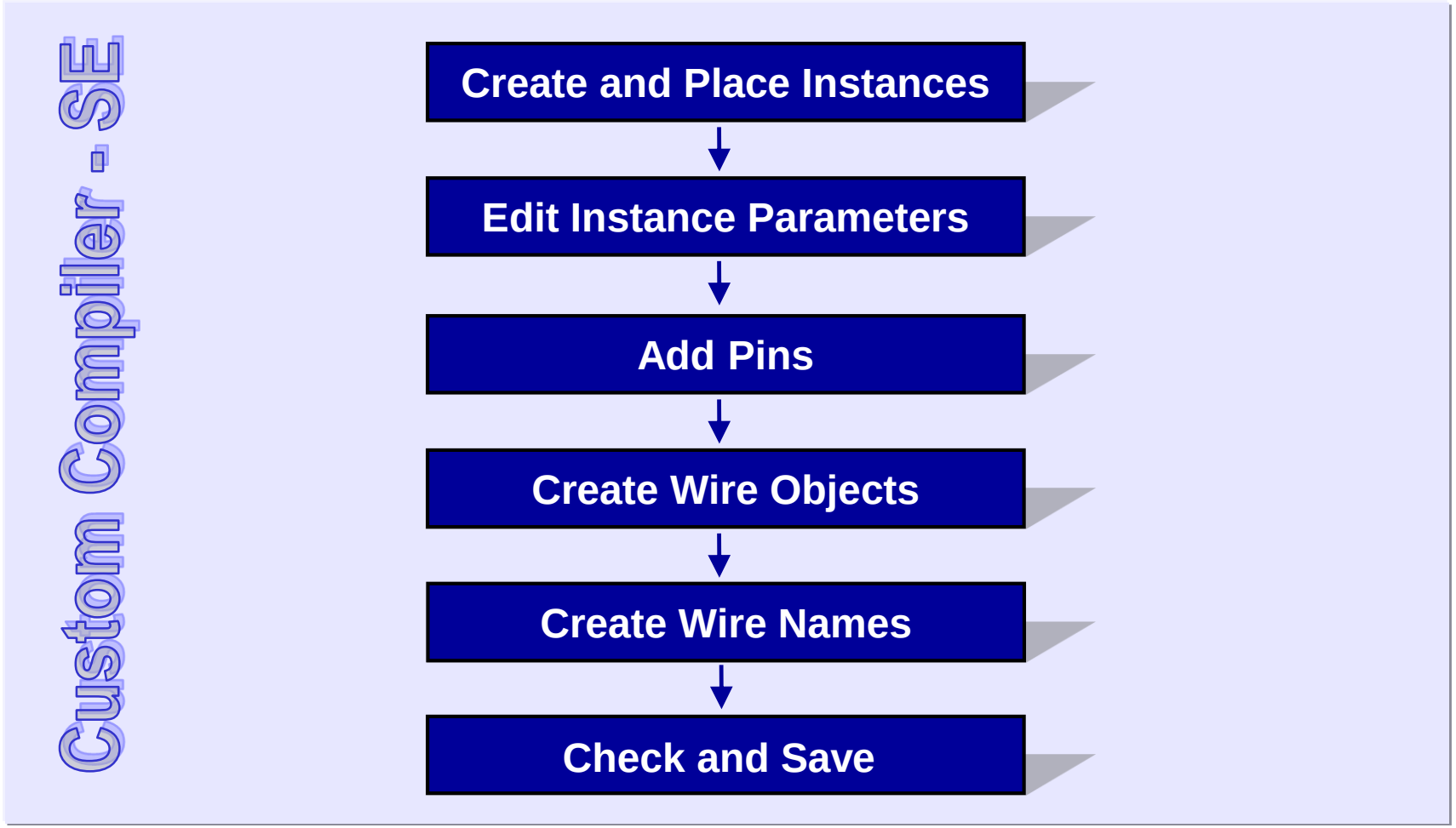
Unit Objectives

- **After completing this unit, you should be able to:**
  - Use a standard Custom Compiler components library
  - Create and place Instances
  - Edit Instance parameters
  - Add pins
  - Wire the design
  - Name the nets
  - Check the schematic for errors

Full Custom Compiler Flow

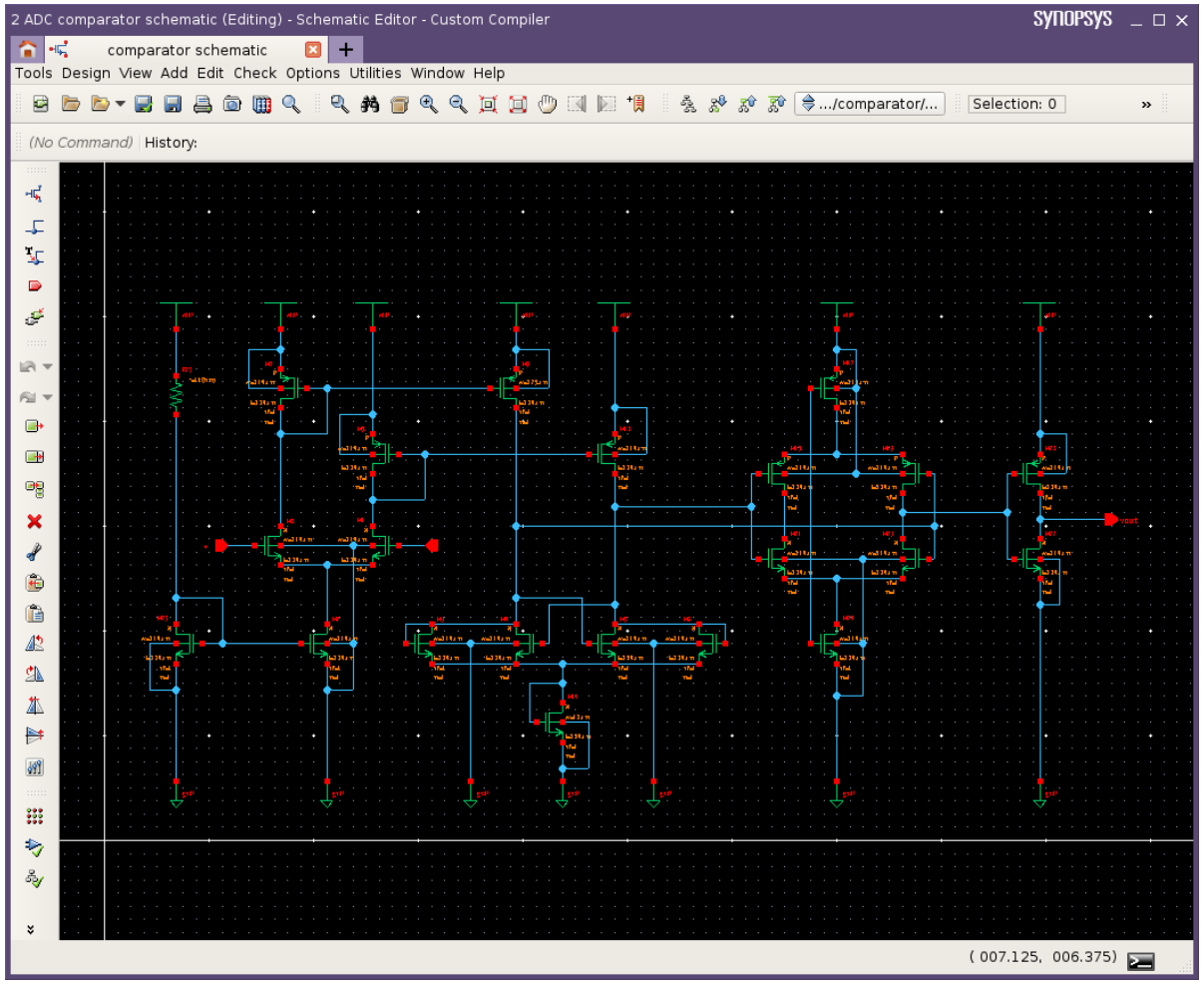


Schematic Capture Flow

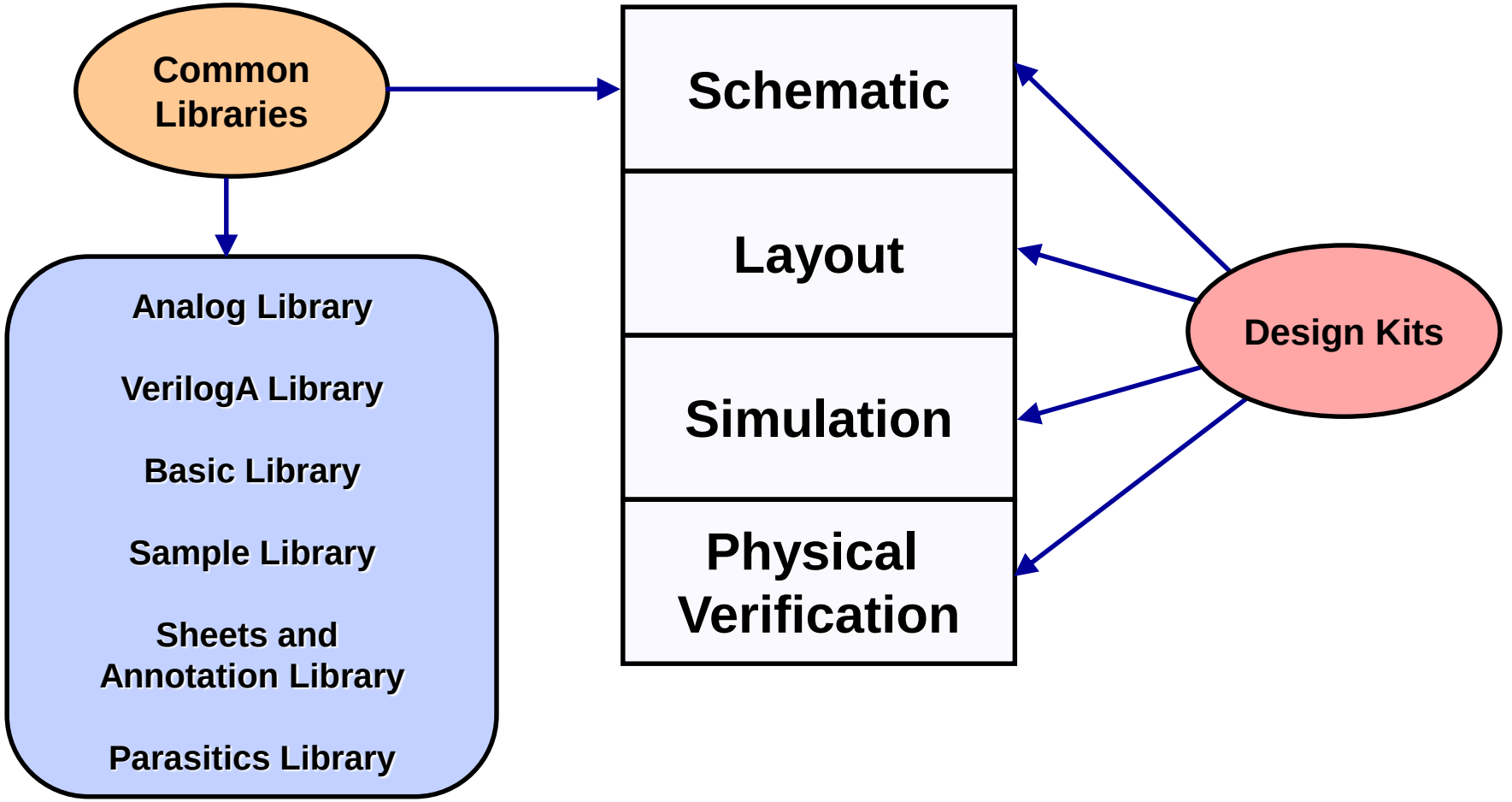


Purpose of Schematic Editor

- **Enables custom IC design entry**
- **Provides commands for placing devices & pins, drawing wires, object manipulation**
- **Launching point for integrated applications**
  - Simulation



Libraries and Design Kits

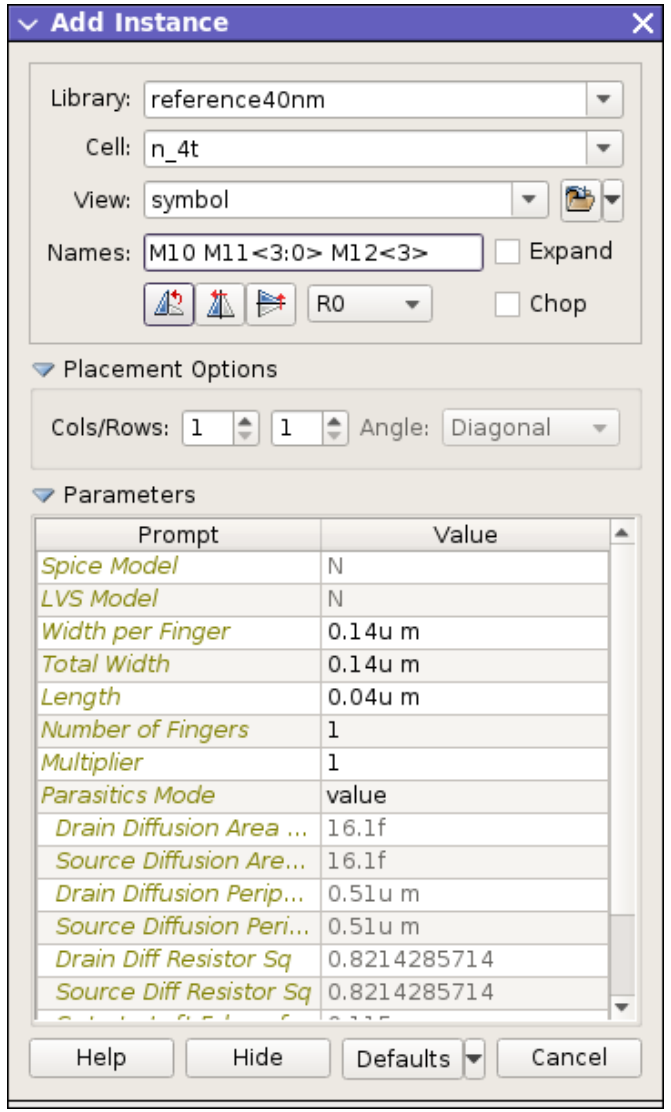


Instance Creation

- **Instances are design objects which reference another symbol or schematic design**
- **Represented by:**
  - Scalar instances
  - Vectored/Iterated instances
- **Added to a design by:**
  - Using Add → Instance (i) command
  - Copying instances
  - Pasting instances from the clipboard

Instance Creation

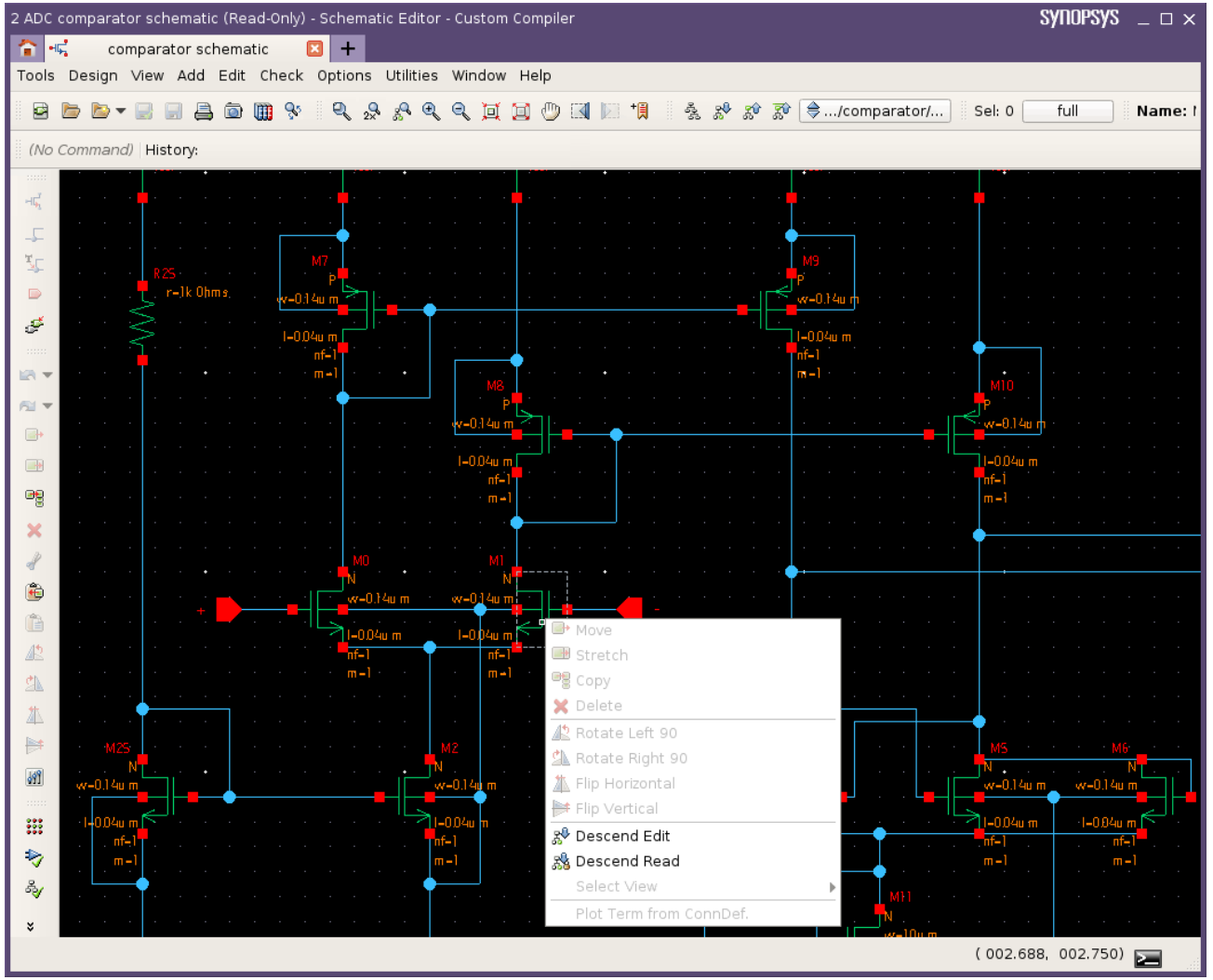
- **Use Model**
  - Invoke the command
  - Specify the master LCV from which the instance has to be added
  - Optionally change parameters
  - LMB on the canvas to place the instance
- **Supports**
  - Instance array (1-D and 2-D) creation by Cols/Rows specification
  - Instance naming
  - Default naming
    - Controlled by user preference
  - Name stacking



Context Sensitive Menu (CSM)

- **Menus that apply to specific objects**
  - Only invoked from active/selectable objects (with exception of the canvas)
- **Bound to binding on Mouse buttons**
- **Active Object vs. Selected Set**
  - CSM always reflects the current active object, even if the active object is not part of the current selected set
  - Why? Allows user to modify an object or to issue canvas commands without disturbing the selected set

Context Sensitive Menu Behavior



Creation / Editing Paradigm

- **All data creation commands repeat until aborted**
- **All edit commands that support canvas interaction, support:**
  - Infix Mode - controlled by a preference
  - Pre / Post Selection Mode
- **Examples:**
  - Editing commands support infix (Edit → Move)
  - Commands that typically require modification of ‘options’ will not support infix (Add → Instance)

Direct Object Manipulation

- **Bound to LMB dragStart-dragStop binding**
- **Allows user to enter a command using drag start**
- **Results in:**
  - Stretch/Move operation
    - Instances
    - Schematic pins, wires
    - Labels
    - Notes
  - Wiring operation
    - Instance pins
- **SE Specific Default Requirements**
  - Only active if no other command is active
  - Works on selected set and on active object
  - Drag start must occur over a selected object or on an active object
- **If drag start is not over a selected/active object then drag start / stop will do an area selection**

Pin Creation

- **Pins are the design objects which are used to define the interfaces of the design**
- **Pins can be created using Add → Pin (P) command**

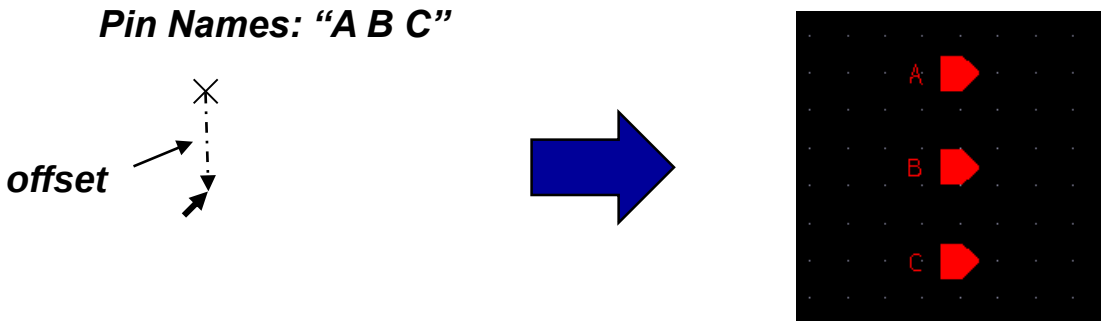
- **They are classified as:**
  - Schematic pins
  - Symbol pins

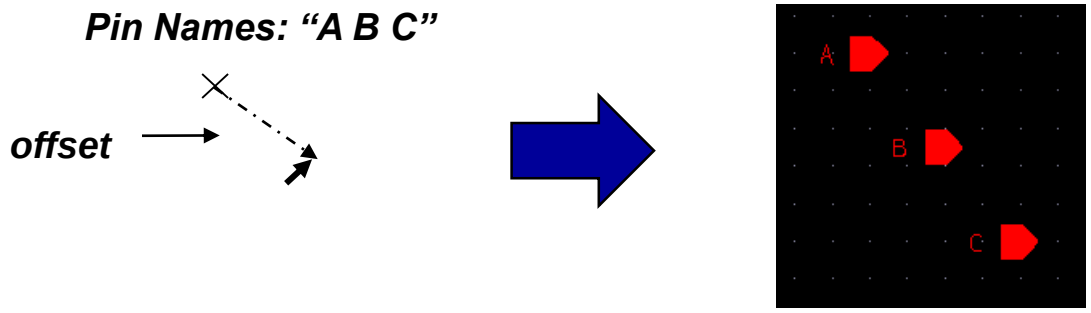
Pin Creation

- **Supported pin types for schematic and symbol:**
  - Input
  - Output
  - InputOutput
  - Switch
  - Tristate
- **Supports**
  - Name stacking
  - Pin array creation
  - Expand bus (vector) and bundle names

Pin Creation (Example)

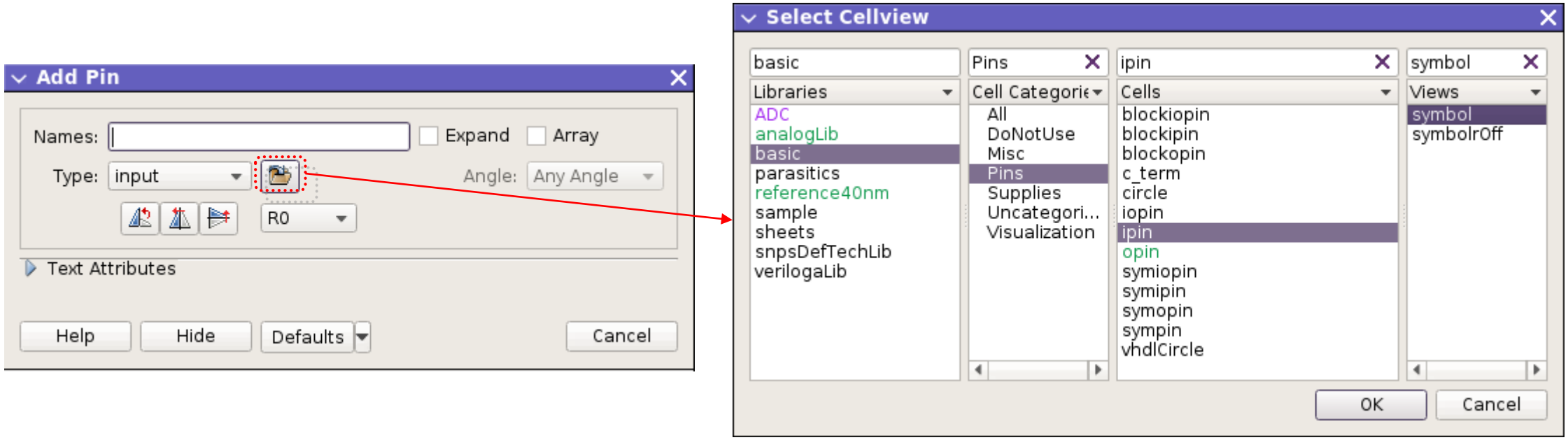
- **Schematic Pin Array UI**





Pin Creation (Custom Pin Shape)

- **User could custom pin shape by select a specific symbol view from COD**



Test for Understanding

- What are the two ways to represent instances in schematic
- What is “Name Stacking”
- CSM stands for _____
- “Direct object manipulation” on instance pins results in which operation
- Add→ Instance supports infix mode: True / False
- All editing commands follow pre-post selection model: True / False

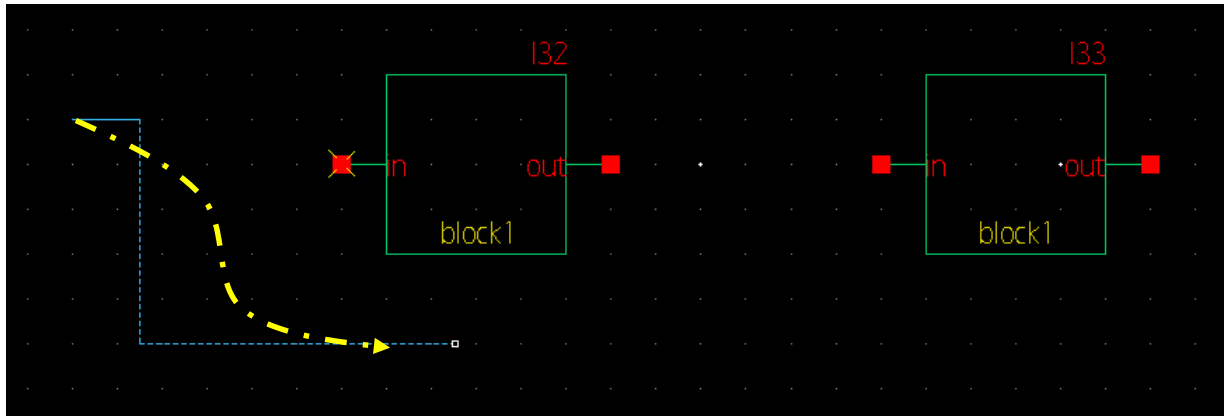
Manual Wiring

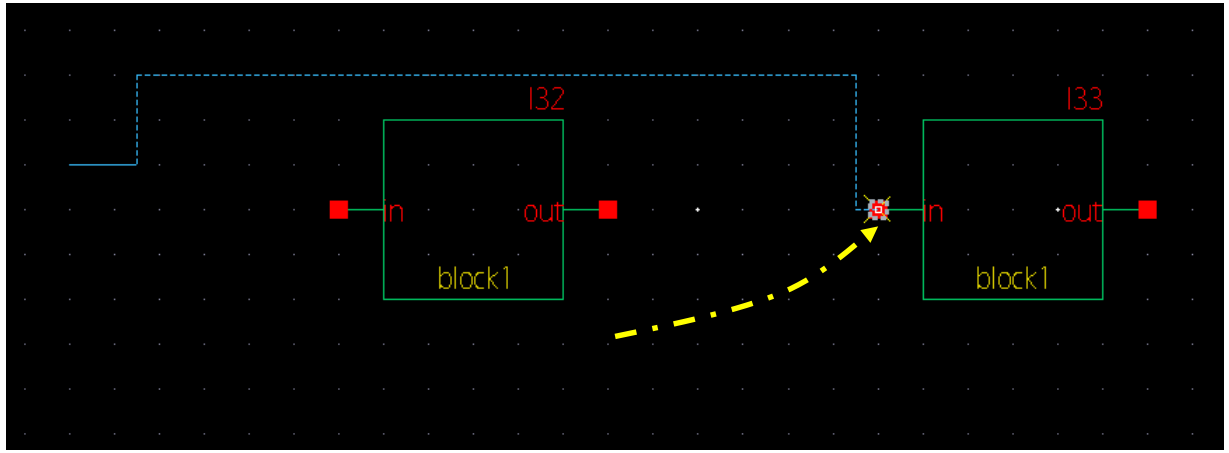
- **Main capabilities are:**
  - Creation of single net wires
  - Creation of bus/bundle net wires
- **Manual router supports the following routing modes:**
  - Auto
  - Guided (i.e. “Follow Mouse”)
  - Orthogonal (Manhattan) wiring
  - Diagonal (45-degree) wiring
  - Any (any-angle) wiring

  } Manual
- **Supported routing options avoid shorts and avoid symbols for Auto and Guided mode**
- **Bus/Bundle wiring support includes:**
  - Wiring all bits of a bus/bundle
  - Tapping/Patch-cording some bits from the bundle

Manual Wiring

- **Wiring with Symbol Avoidance (Auto Mode)**

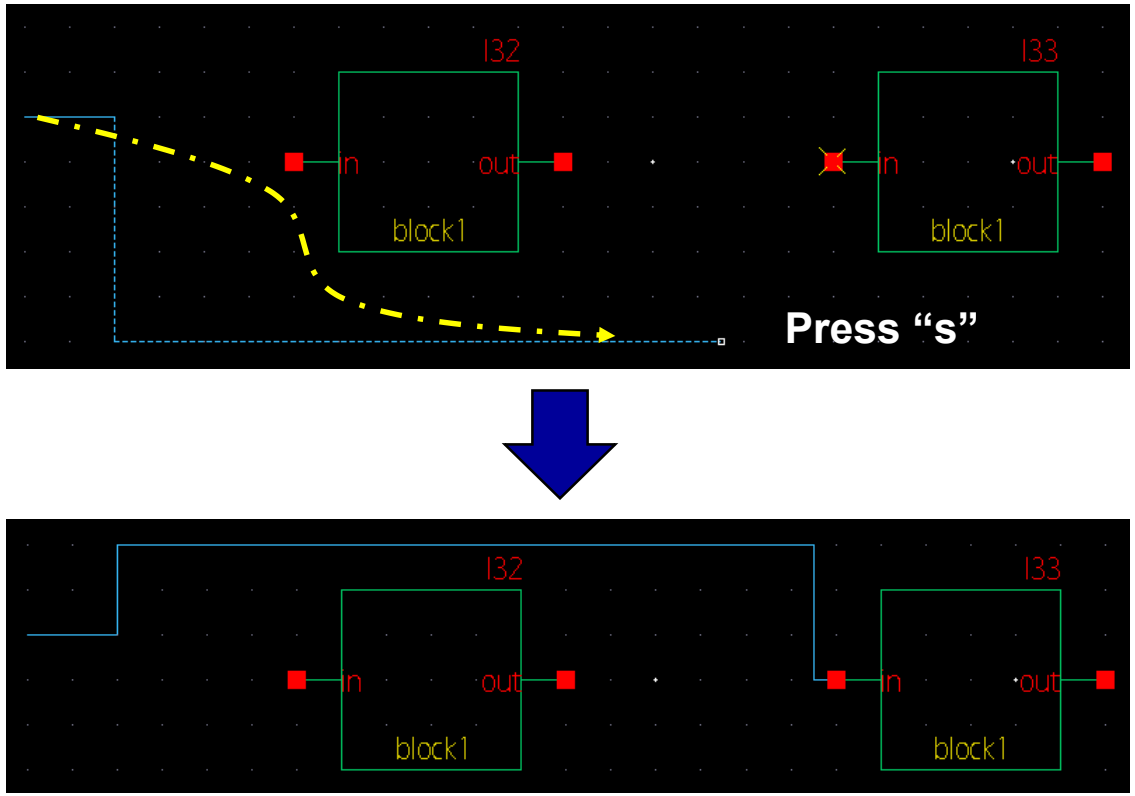




Tip: “y” key changes the wiring path.

Wiring: “Smart Connect”

- Accelerates wiring
- Can snap wiring to closest wire point using “s” bindkey

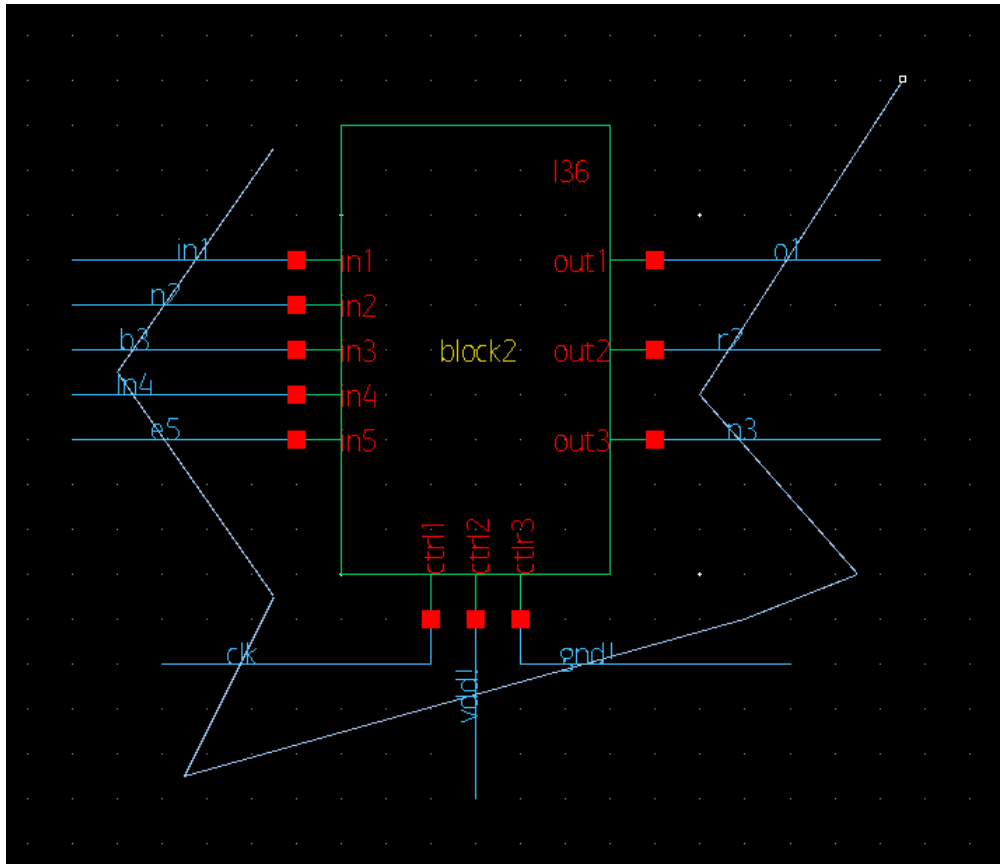


Wire Name Creation

- **Wire Name can be created**
  - Directly during wire creation
  - Using Add → Wire Name (L) command
  - Bringing mouse cursor on the wire and pressing “Enter”
- **Command doesn't support infix mode**
- **Supports**
  - Name Stacking
  - Array naming
  - Expand bus (vector) and bundle names

Wire Name Creation

- **Wire Naming using Polyline**

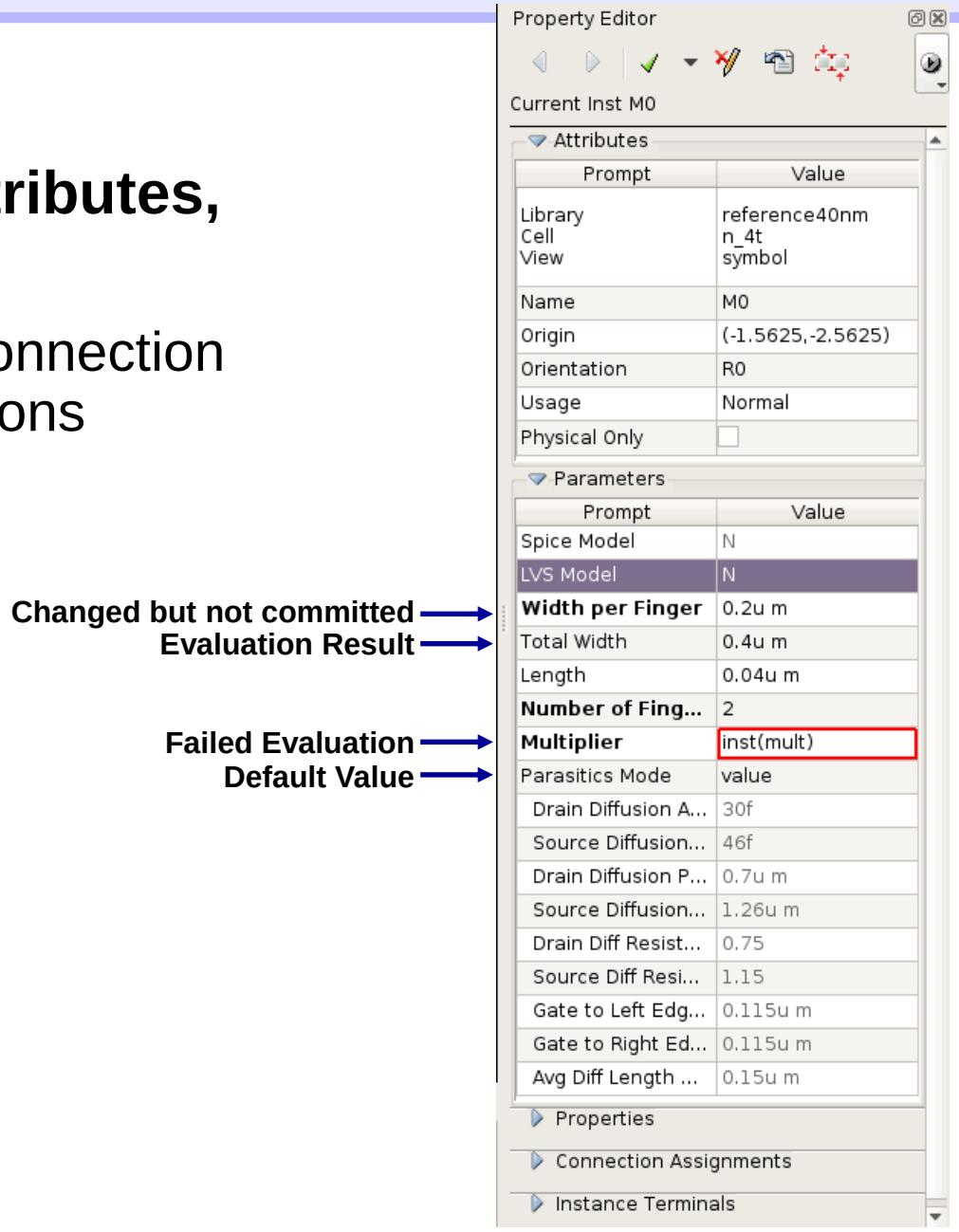


Connectivity

- **Physical connections**
  - Connections only at endpoints
  - Wiring objects must intersect pin bounding box
  - Pins overlapping pins
  - Etc…
- **Connect by name**
  - Done through the use of explicit wire names – this is how user controls the name of a net
  - Any nets with same name in a cellView considered connected
- **Globals**
  - Custom Compiler supports a “promote-to-global” symbol

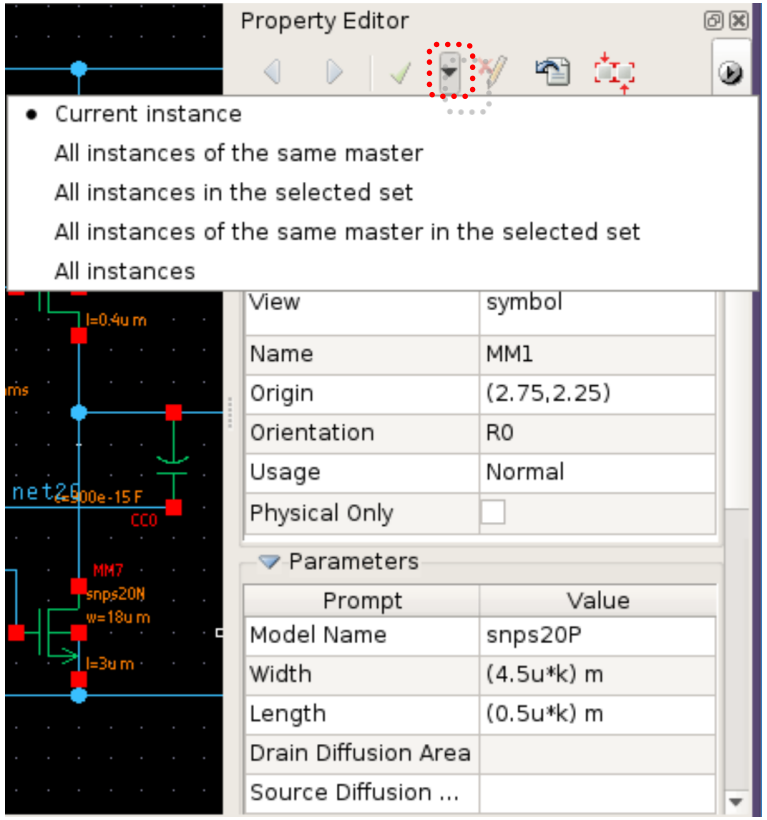
Property Editor Assistant (PE)

- **Main tool for managing object attributes, properties**
  - Instance parameters, properties, connection assignments and terminal connections



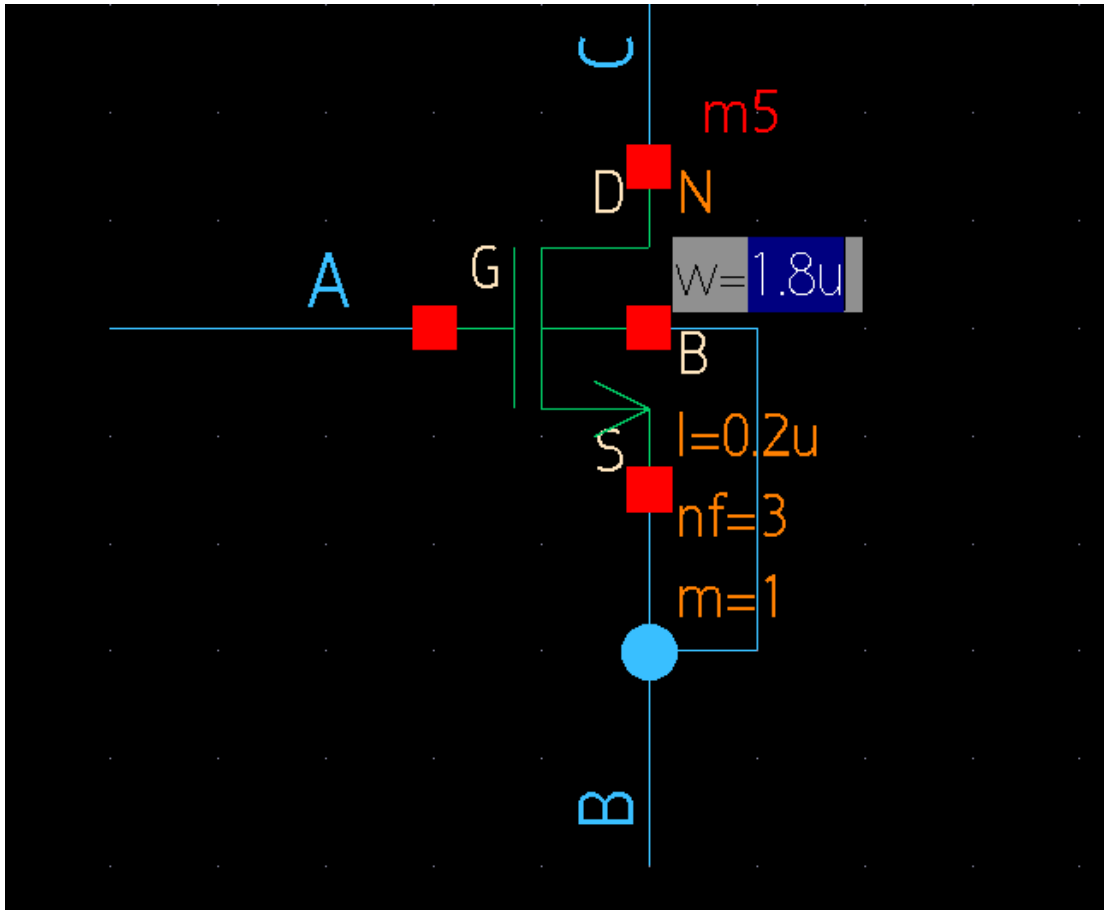
Property Editor Assistant (PE)

- **Using pull-down menu to apply property change for multiple objects**



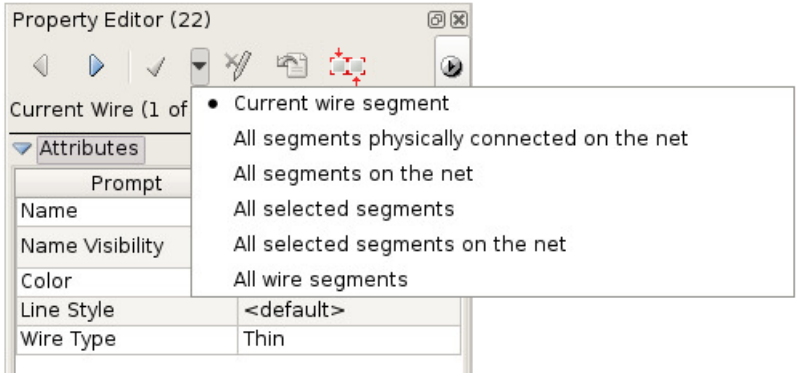
On-Canvas Editing

Supported for device property labels, wire names, etc.

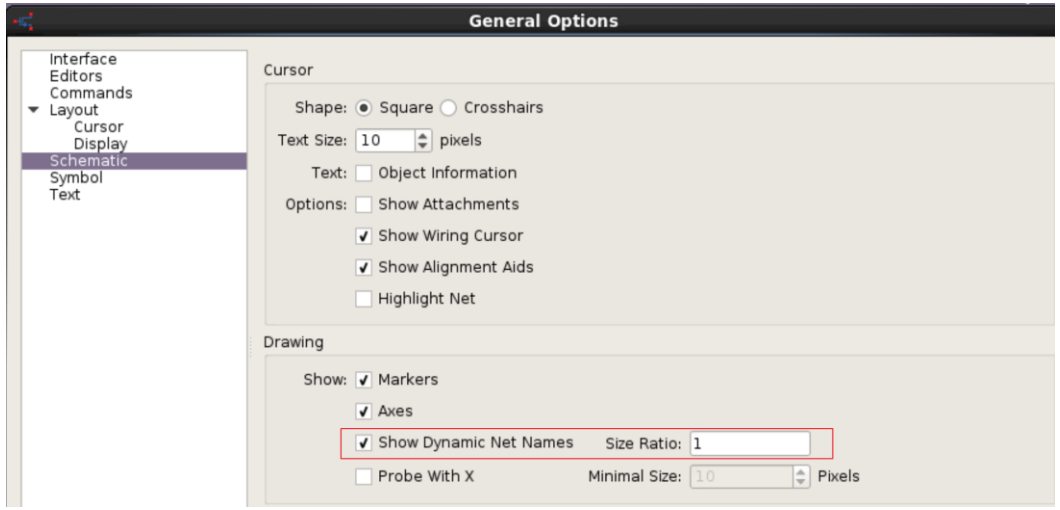


Wire Name Manipulation Using PE

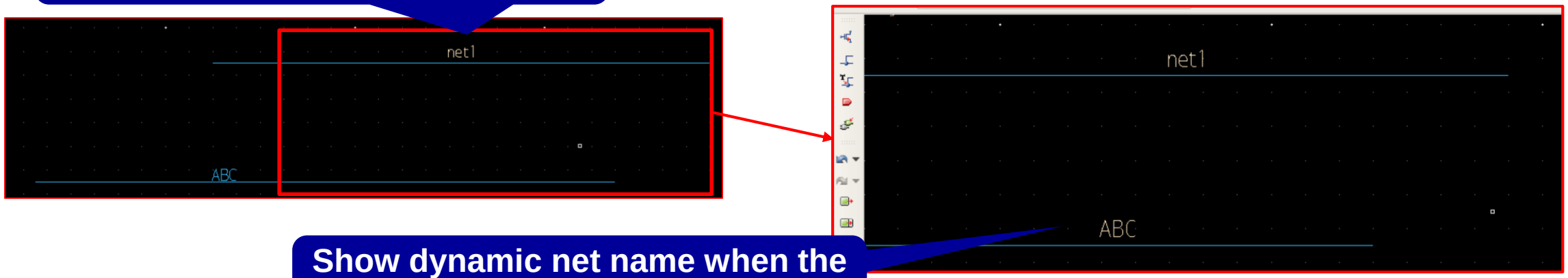
- Current wire segment
- All segments physically connected on the net
- All segments on the net
- All selected segments
- All selected segments on the net
- All wire segments



Dynamic Net Names



Non label wire still show wire name



Show dynamic net name when the wire label is not in view window

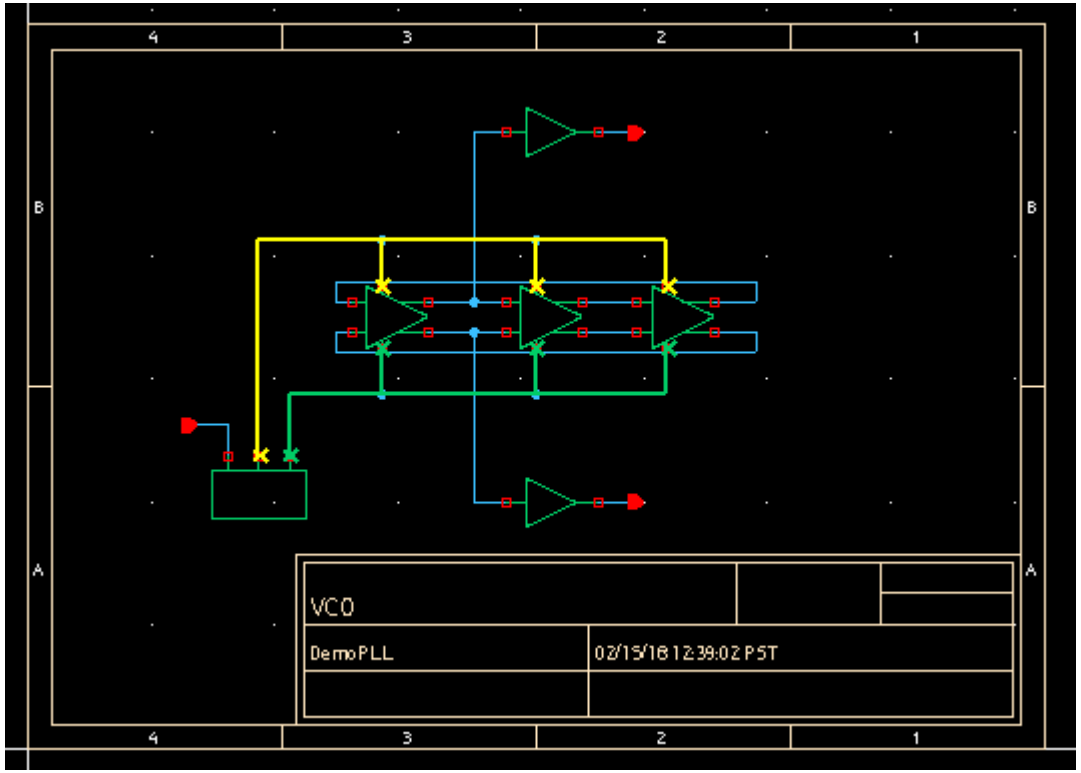
Highlighting Connections

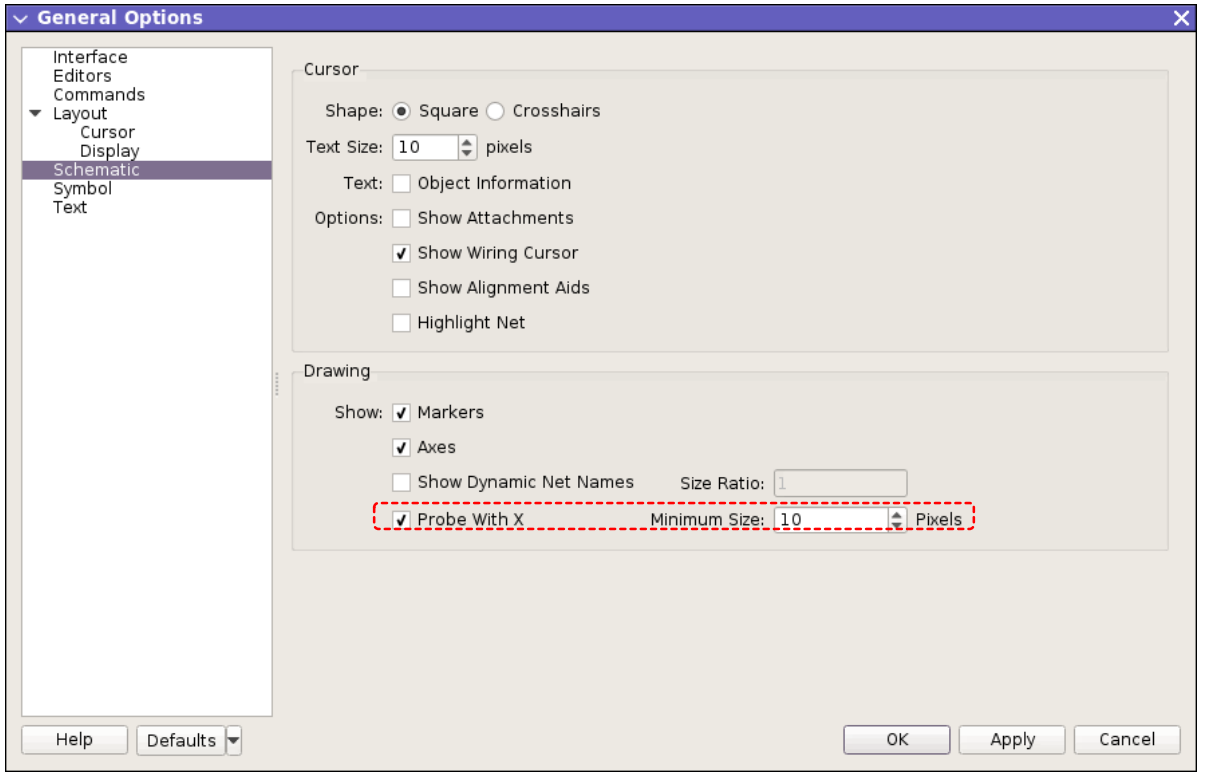
- **Visualize hierarchical connections between blocks**
- **Invoke the command Add → Probe…(Shift+P)**

- **Type**
  - Net
  - Terminal
  - Instance
  - Instance terminal
  - Any

Highlighting Connections

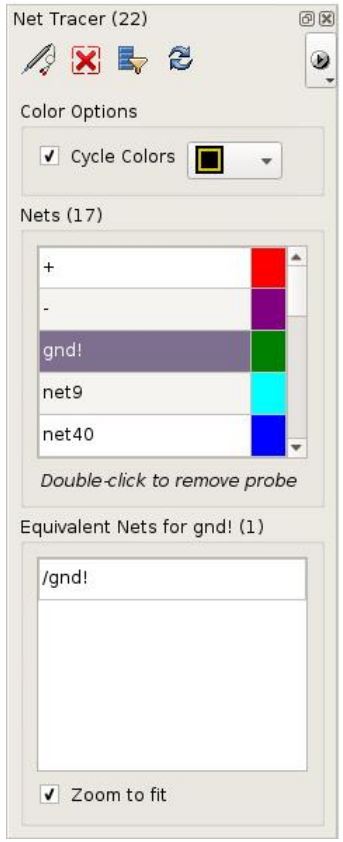
- **Highlight Small Shapes with X Symbol**

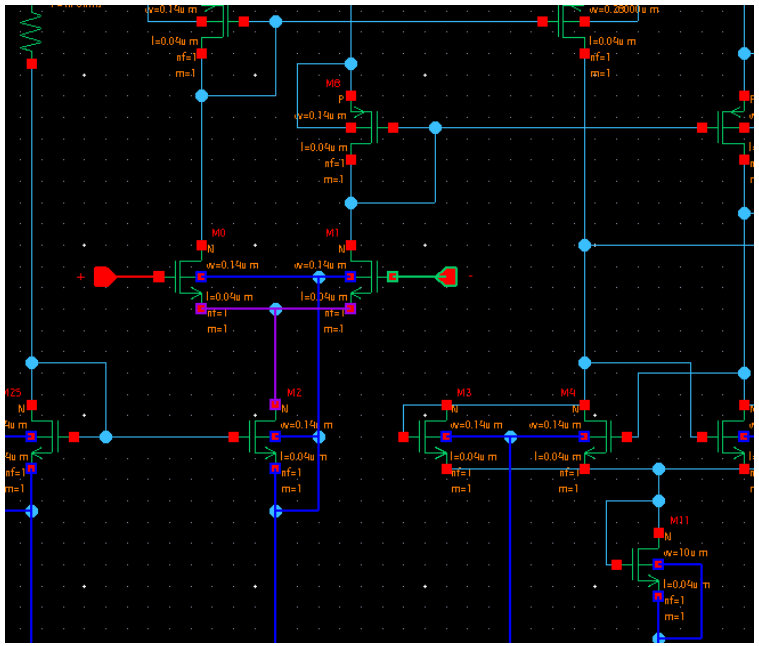




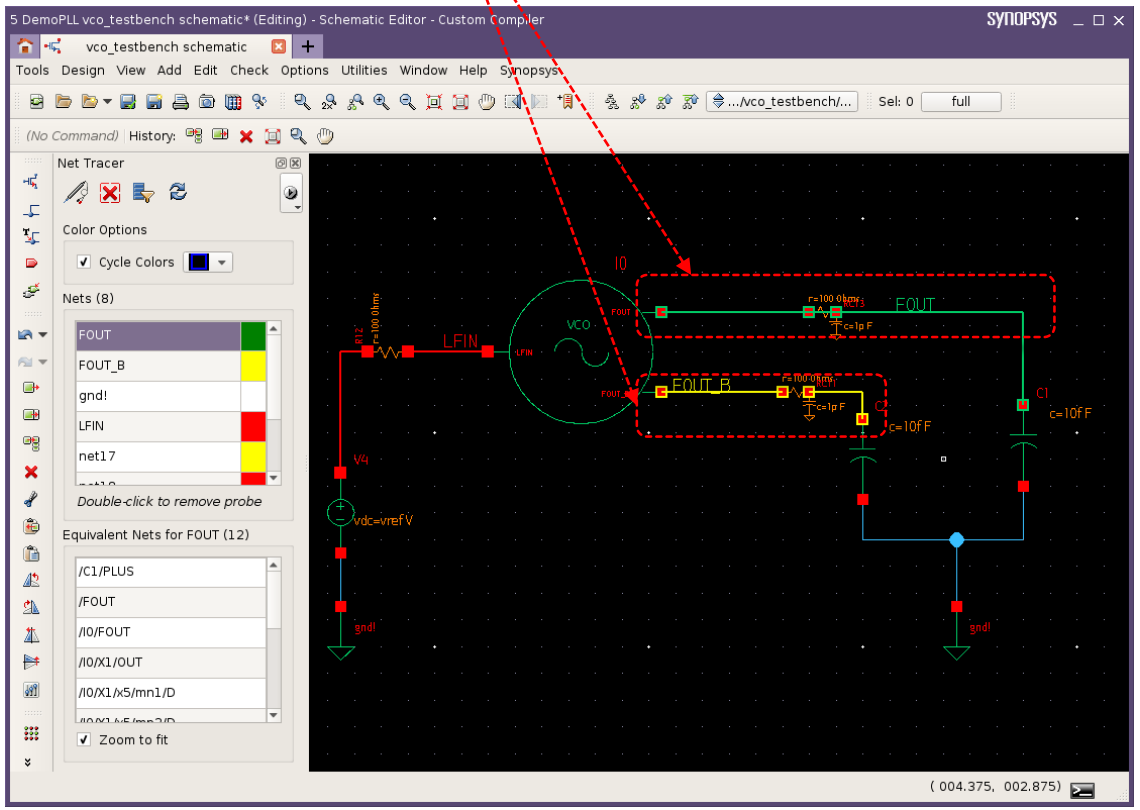
Net Tracer Assistant

- **Visualize hierarchical connections between blocks and list all equivalent nets through hierarchy**





- **Support parasitic devices trace through**



Test for Understanding

- What are the three different modes of routing
- Connect by name is an invalid logical connectivity: True / False
- What is default behavior when changing wire name from Property Editor

Checking the Design: SRC/ERC Rules

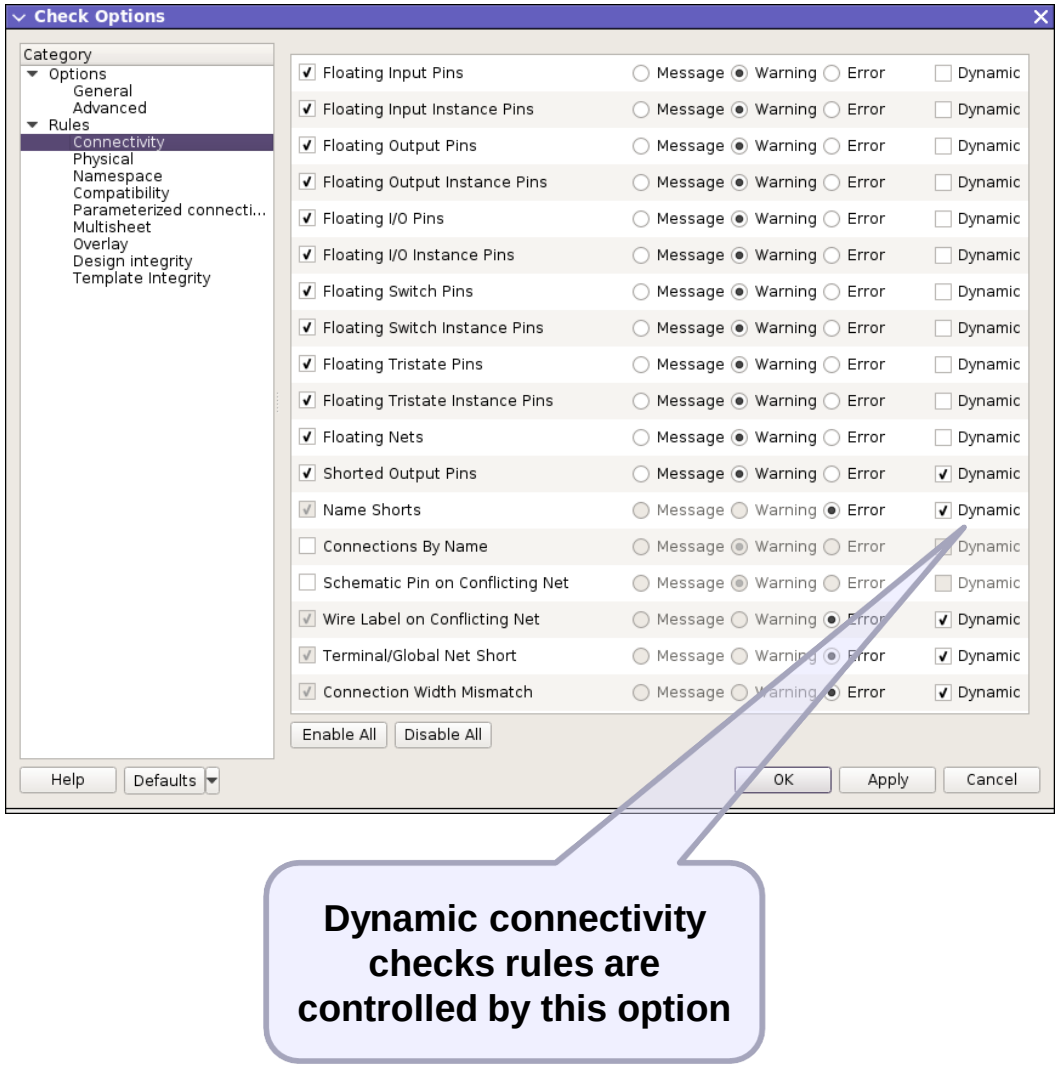
- **Allows user to verify the connectivity**
- **Can check current cellView or full hierarchy**
- **Cross-check views**
- **Comprehensive sets of checks**
  - Connectivity: floating inputs/outputs, ambiguous connections, name shorts
  - Physical: unconnected wire segments, solder dots on crossing wires, off grid objects
  - Name-space specific: Verilog, HSPICE

Marker Creation

- Markers are the visual objects to flag an ERC/SRC violation with an informational message in the console
- Markers are created enclosing violating objects
- Created in the design for the rules with severity level of warning and error
- Managed by an assistant called “Marker Browser”

Dynamic Connectivity Checks

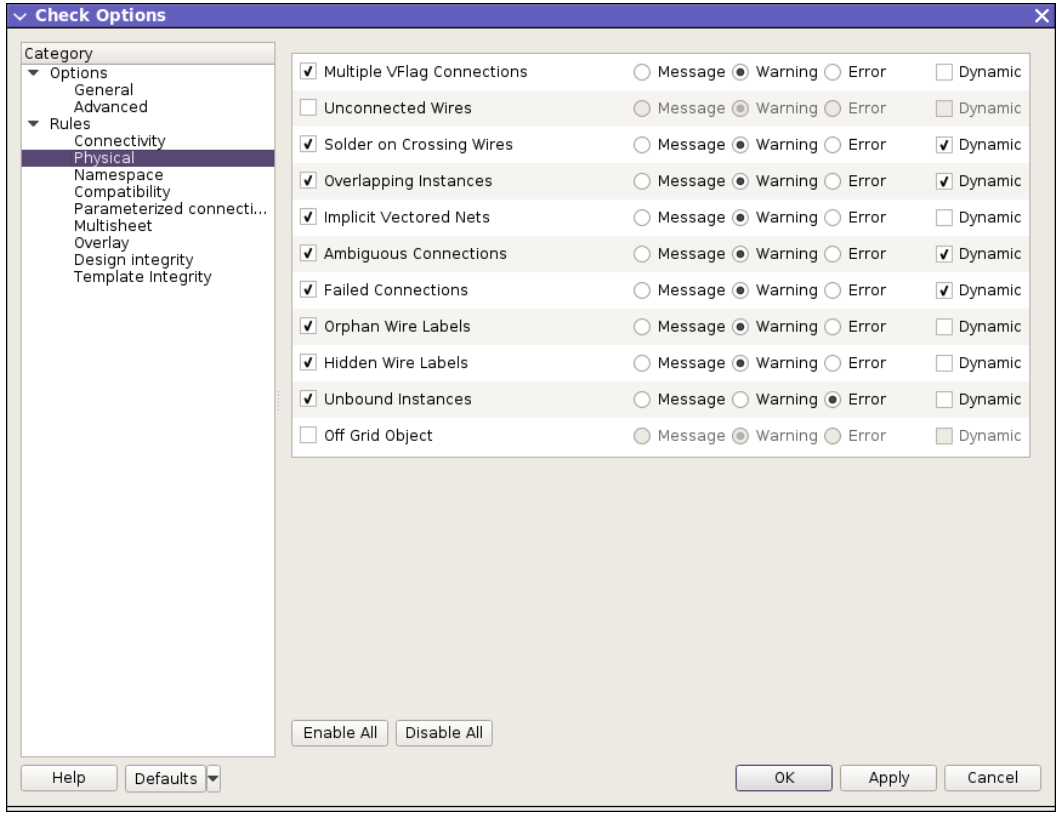
- User-Configurable
- Default rules are:
  - Floating Pins
  - Floating Instance Pins
  - Floating Nets
  - Shorted Output Pins
  - Name Shorts
  - Wire Label on Conflicting Net
  - Terminal/Global Net Short
  - Connection Width Mismatch



Dynamic connectivity checks rules are controlled by this option

Physical Checks

- **Default rules are:**
  - **Solder on Crossing Wires**
  - **Overlapping Instances**
  - **Implicit Vectored Nets**
  - **Ambiguous Connections**
  - **Failed Connections**
  - **Orphan Wire Labels**
  - **Hidden Wire Labels**
  - **Unbound Instances**



Test for Understanding

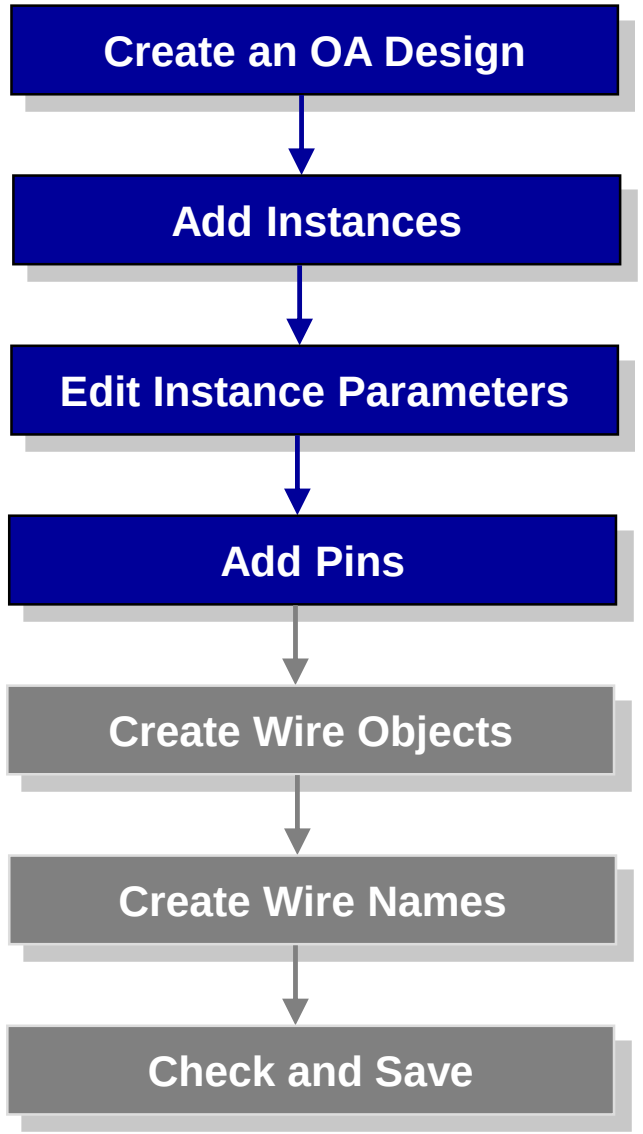
- ERC/SRC checks allows you to _____
- Marker objects are created for the rules with severity level “Message”: True / False
- How to enable dynamic checks for ERC/SRC rules?

Lab 1A: Basic Schematic Entry

20 minutes

Goals:

- To become familiar with Custom Compiler Schematic Environment
- To understand Schematic connectivity object
- To understand editing functions

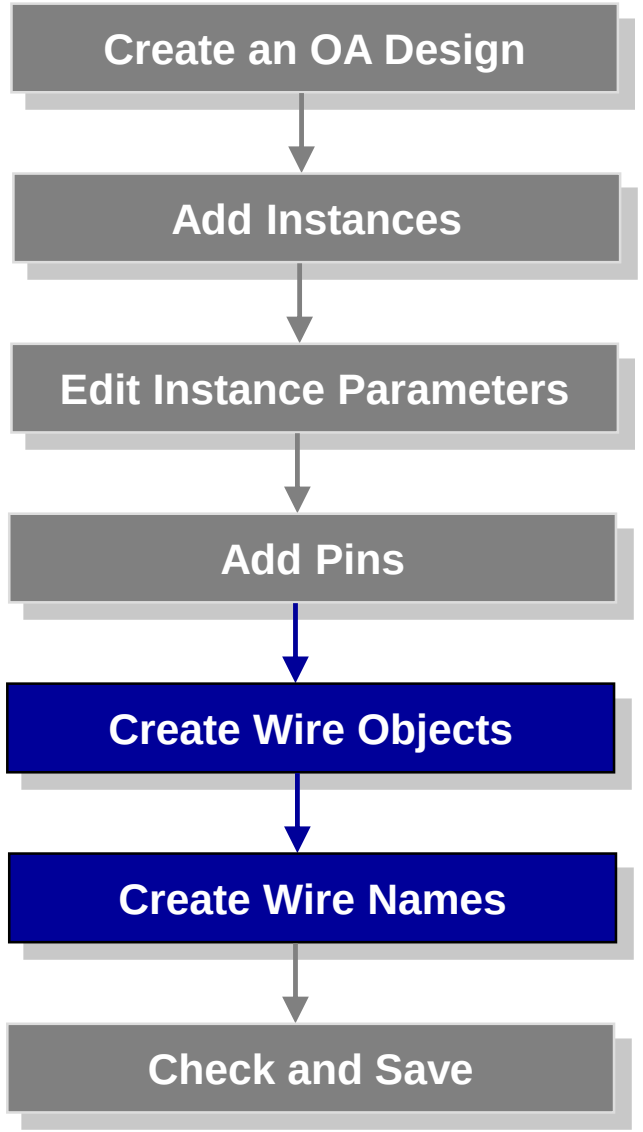


Lab 1B: Basic Schematic Entry

15 minutes

Goals:

- To understand Schematic wiring creation
- To understand connectivity concepts



Lab 1C: Basic Schematic Entry

10 minutes

Goals:

- To understand the concept and importance to verify the connectivity