Tutorial 1 Summary (B)

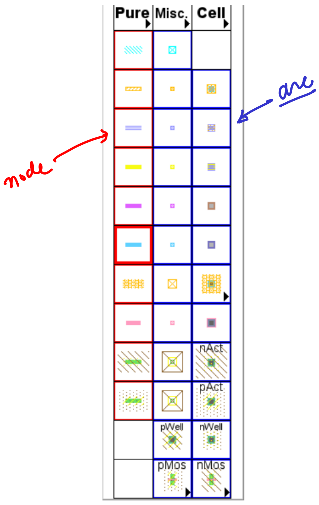
Based on
Electric VLSI Design System Tutorials from CMOSedu.com
By R. Jacob Baker

Modified by Young W. Lim

.

Copyright (c) 2015 Oracle andStatic Free Software.

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies. Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided also that they are labeled prominently as modified versions, that the authors' names and title from this version are unchanged (though subtitles and additional authors' names may be added), and that the entire resulting derived work is distributed under the terms of a permission notice identical to this one. Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions.



bar that has 3 tabbed sections, the *components menu*, the *cell explorer*, and the *layers*. You can move it to the right side with the **On Right** command (of menu **Windows / Side Bar**) and move it back with the **On Left** command. You can also request that the side bar always be on the right by checking "Side bar defaults to the right side" in the Display Control Preferences (in menu **File / Preferences...**, "Display" section, "Display Control" tab).

The cell explorer lets you examine the hierarchy, system activity, and error messages (see Section 4–5–2 for more).

The *components menu* shows a list of nodes (blue border) and arcs (red border) that can be used in design. The arrangement of the entries in the components menu varies with the different technologies. For MOS technologies, see Section 7–4–2; for schematics, see Section 7–5–1; and for artwork, see Section 7–6–1.

The top three entries in the components menu let you place pure–layer nodes (see Section 6–10–1), miscellaneous objects (see Section 7–6–3) and instances of cells (see Section 3–3).

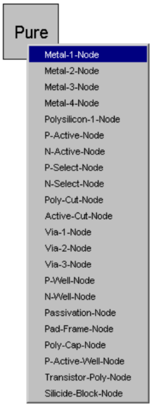
The layers tab lets you control which parts of the display are visible. See Section 4–5–3 for more on layer visibility.

6–10–1: Introduction to Outlines

For some primitive nodes, it is not enough to rotate, mirror, and scale. These primitives can to be augmented with an *outline*, which is a polygonal description.

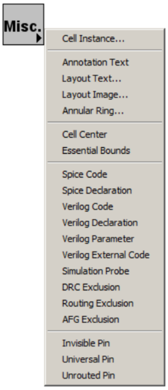
There are quite a few primitive nodes that make use of outline information. The MOS transistors use the outline to define the gate path in serpentine configurations (see Section 7–4–1). The Artwork technology has nodes that use outline information: Opened–Solid–Polygon, Opened–Dotted–Polygon, Opened–Dashed–Polygon, Opened–Thicker–Polygon, Closed–Polygon, Filled–Polygon, and Spline (see Section 7–6–1).

For arbitrary shapes on arbitrary layers, use the *pure–layer* nodes in the IC layout technologies. The pure–layer nodes are found under the "Pure" entry in the component menu. For example, the node called "Metal–1–Node" in the CMOS technologies looks like a rectangle of the Metal–1, until you add outline information. With an outline, this node can take any shape. It is even possible to have multiple disjoint outlines in a single pure–layer node (users cannot create this situation, but some tools such as GDS import can).



Special Nodes

There are also special nodes in the Generic technology. They are all available from the "Misc." entry of the component menu.



A special primitive, called *Cell Center*, defines the origin of any cell. Once the node is placed, its location is at (0,0) for the cell. Since instances of the current cell use the origin as the *anchor point* for cursor–based references, the location of this node defines the anchor. For example, if you place this node in the upper–right corner of a cell, then creation commands place instances such that their upper–right corner is at the cursor. See Section 3–3 for more information on cell centers.

A special primitive, called *Essential Bounds*, defines an alternate boundary of any cell. At least two of them must be placed in opposite corners, although 4 can be place to make it look better.

Note that the Cell Center and Essential Bounds nodes are made "hard–to–select" by default, which means that they can be selected only by using "Special Select" mode (see Section 2–1–5 for more).

The *Spice Code* and *Spice Declaration* entries create text for Spice decks (see Section 9–4–3). The *Verilog Code*, *Verilog Declaration*, *Verilog Parameter*, and *Verilog External Code* entries create text for Verilog decks (see Section 9–4–2). These entries actually create Invisible Pin nodes with appropriate text on them.

A special primitive, called *Simulation Probe* is recognized by simulators and visually modified to reflect whatever it is connected to. The simulators that reflect the state of the circuit by drawing lines along arcs also fill–in these probe nodes. It provides a visual display of simulation activity, and works especially well with the VCR controls in the waveform window. See Section 4–11 for more.

The *DRC Exclusion* node is used by the design–rule checker (see Section 9–2–3). The *Routing Exclusion* node is used by routers to tell them to avoid certain layers under this node (see Section 9–6–1). Currently only the Sea–of–Gates router handles this. The *AFG Exclusion* node is used by the auto–fill generator (see Section 9–8–2).

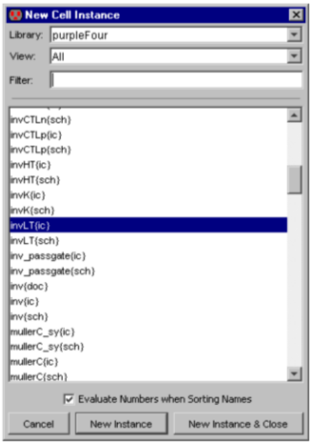
The *Spice Code* and *Spice Declaration* entries create text for Spice decks (see Section 9–4–3). The *Verilog Code*, *Verilog Declaration*, *Verilog Parameter*, and *Verilog External Code* entries create text for Verilog decks (see Section 9–4–2). These entries actually create Invisible Pin nodes with appropriate text on them.

A special primitive, called *Simulation Probe* is recognized by simulators and visually modified to reflect whatever it is connected to. The simulators that reflect the state of the circuit by drawing lines along arcs also fill–in these probe nodes. It provides a visual display of simulation activity, and works especially well with the VCR controls in the waveform window. See Section 4–11 for more.

The *DRC Exclusion* node is used by the design–rule checker (see Section 9–2–3). The *Routing Exclusion* node is used by routers to tell them to avoid certain layers under this node (see Section 9–6–1). Currently only the Sea–of–Gates router handles this. The *AFG Exclusion* node is used by the auto–fill generator (see Section 9–8–2).

3–3: Creating Instances

To place an instance of a cell in another cell, use the "Cell" button in the component menu. After choosing a cell from the popup list, click in the edit window to place the instance.



Library: purpleFour
View: All
Filter:

invCTLn(sch)
invCTLp(ic)
invCTLp(sch)
invHT(ic)
invHT(sch)
invK(ic)
invK(sch)
invLT(ic)
invLT(sch)
inv_passgate(ic)
inv_passgate(sch)
inv(doc)
inv(ic)
inv(sch)
mullerC_sy(ic)
mullerC_sy(sch)
mullerC(ic)
mullerC(sch)

[✓] Evaluate Numbers when Sorting Names

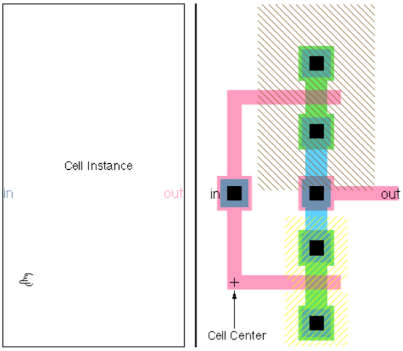
Another way to place an instance of a cell is to use the **Place Cell Instance...** command (in menu **Cell**). You will be shown a list of cells that are available for creation. After selecting one, click to create an instance in the current cell.

The cell selection dialog has three controls at the top for viewing cells. The "Library" popup lets you choose which library to examine. You can choose "ALL" to see cells from all libraries. The "View" popup lets you see only those cells in the specified view. Again, you can choose "All" to see all views. The "Filter" field contains a regular expression that must match a cell name in order to list it. For an explanation of the "Evaluate Numbers when Sorting Names" checkbox, see Section 3–7–1.

If you place an instance from a different library, that library will be linked to the current one. Linked libraries are read from disk together, and form a single hierarchy that spans multiple files. See Section 3–9–1 for more on libraries.

An alternate way to create a cell instance is to duplicate an existing one on the screen. This requires that an instance of that particular cell already exist. Select the existing cell and use the **Duplicate** command (in menu **Edit**). Then move the cursor to the intended location of the new instance and click to create the copy. Note that this command copies all attributes of the original node including its orientation.

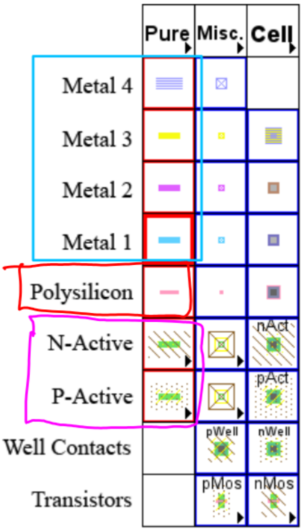
When a cell instance is being created, the cursor points to its *anchor point*. The anchor point is that point inside of the cell where the coordinate space has its origin. This is often defined by the location of a *cell−center* node inside of the cell (see Section 7−6−3).



Most cells have a cell−center node placed automatically in them. If there isn't one and you want it, click on the "Misc" button in the component menu on the left, and choose "Cell Center". A cell−center node, placed inside of the cell definition, affects the anchor point for all subsequent creation of instances of the cell.

The cell−center is always at the origin of the cell. If you move it, then the origin moves (in other words, moving the cell center is really like moving everything else in the cell). Note that the cell center is "hard to select" and can only be moved in "special select" mode (see Section 2−1−5). You can move the cell center to the center of the selected objects by using the **Cell Center to Center of Selection** command (from menu **Edit / Move**).

7-4-2: The MOSIS CMOS Technology

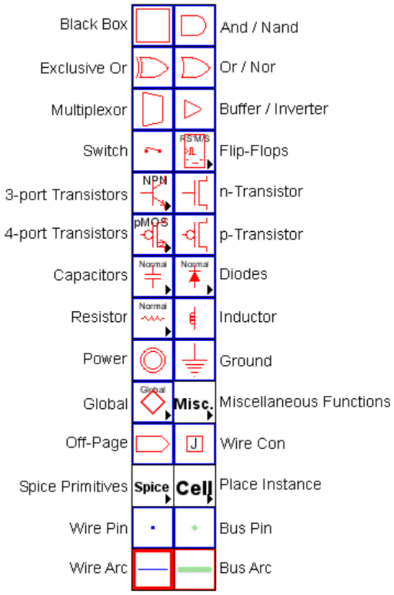


The MOSIS CMOS technology describes a scalable CMOS process that is fabricated by the MOSIS project of the University of Southern California. To obtain this technology, use the popup menu at the top of the component tab (in the side bar) and select "mocmos".

This technology can have from 2 to 6 layers of metal (4 are shown here, 6 is the default). It has 1 polysilicon layer but can be changed to use 2. The technology can be set to use either standard rules (SCMOS), submicron rules, or deep rules. You can choose whether to allow stacked vias and whether or not to use alternate contact rules. Finally, you can set the technology into "Analog" mode, which provides an NPN transistor, a Polysilicon Capacitor, and many resistors. This is done with the Technology Preferences (in menu **File / Preferences...**, "Technology" tab).

The default orientation of transistors (both in the menu, and when first placed) can be rotated by checking "Rotate transistors in menu" in the Technology Preferences.

7–5–1: Introduction to Schematics



The Schematic technology allows you to design using digital and analog schematic components. To obtain this technology, use the popup menu at the top of the component menu and select "schematics".

There are two arcs in the Schematic technology: the wire (blue) and the bus (green). These arcs can be drawn at 45 degree angles. One typically names busses with array names (for example "insig[0:7]"), and then names wires with scalar names (for example "insig[1]"). See Section 6–9–3 for more on bus naming.

To make a physical connection from a wire to a bus, the bus pin can connect to either, so it acts as a tap. In addition, the Wire Con node connects wires to busses, or connects busses of different width, replicating the narrower side to make it as wide as the wider side. Use the **Rip Bus** command (in menu **Edit / Arc**) to automatically add taps to a bus

•

9–4–3: Spice

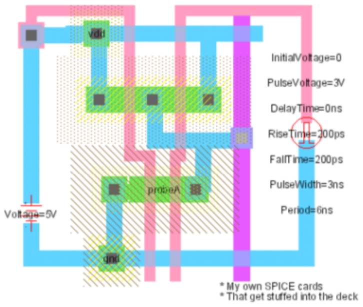
• Electric can produce input decks for Spice simulation with the **Write Spice Deck...** command (in menu **Tools / Simulation (Spice)**). Since there are may formats of Spice output, you must first set the "Spice Engine" field of the Spice/CDL Preferences (in menu **File / Preferences...**, "Tools" section, "Spice/CDL" tab). After the Spice deck has been written, you must run Spice externally to produce a simulation output file. Note that the Electric distribution does not come with a Spice simulator: you must obtain it separately.

After Spice has finished running, use the **Plot Simulation Output, Guess File** command (in menu **Tools / Simulation (Spice)**) to read the Spice output and plot the waveforms. If the file cannot be guessed from the cell name, you can use **Plot Simulation Output, Choose File...**, to select the desired Spice output file. The Spice simulation information is shown in a waveform window (see Section 4–11 for more).

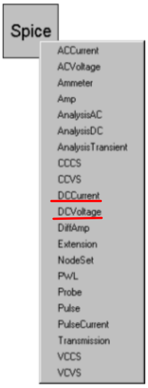
Special Spice Nodes

There are many powerful facilities for running Spice with Electric. The example shown here illustrates some of these facilities. This example is available in the Samples library as cell "tool–SimulateSpice" (you can read the library with the **Load Sample Cells Library** command, in menu **Help**).

All input values to Spice are controlled with special nodes, found in the "Spice" component menu entry. Note that the first time any Spice node is placed, the library of Spice parts is loaded into Electric, so there may be a delay.



The Spice nodes described here are Electric's default set. However, additional sets can (and have) been written. To choose another set, use the Spice/CDL Preferences (in menu **File / Preferences...**, "Tools" section, "Spice/CDL" tab). Under the setting "Spice primitive set", choose another set. A second set of nodes, called "SpicePartsS3", is tailored towards special Spice3.



In this example, there is a 5−volt supply on the left. It was created by using the "DC Voltage" entry under "Spice" entry of the component menu. Once placed, the text that reads "Voltage=0V" can be selected and modified (either with **Object Properties...** or by double−clicking on it). The Pulse input signal on the right is created with the "Pulse" entry under "Spice" (it has 7 parameters).

There are both voltage and current sources, in AC and DC form. There is a piecewise−linear (PWL) source, and two pulses (voltage and current). A set of "two−gate" devices are also available: "CCCS", "CCVS", "VCCS", "VCVS", and "Transmission".

It is possible to specify Transient, DC, or AC analysis by using the "Transient Analysis", "DC Analysis", and "AC Analysis" subcommands. The "Probe" lets you graphically specify signals of interest to Spice. Only one such element may exist in a circuit.

For advanced users, there are two special Spice nodes: "Node Set" and "Extension". The Node Set may be parameterized with an arbitrary piece of Spice code. Truly advanced users may create their own Spice nodes by modifying the cells in the Spice library (see next Section).

Spice Text

This example also shows the ability to add arbitrary text to the Spice deck, as shown in the lower–right. To create this text, use the "Spice Code" or "Spice Declaration" entries under the "Misc." button in the component menu. These command create text that can be modified arbitrarily. Whatever the text says will be added to the Spice deck (declarations go near the top).

Another option that can be used when modeling transistors and other component is to set a specific Spice model to use for that component. To set a node's model, select it and use the **Set Spice Model...** command (in menu **Tools / Simulation (Spice)**).

The **Add Multiplier** subcommand places a multiplier on the currently selected node. Multipliers (also called "M" factors) scale the size of transistors inside of them.

Another piece of text that can be added to a circuit is for separate flattened analysis files. This is useful for Nanosim timing assertions, hierarchical measurements, etc. The **Add Flat Code** subcommand places a piece of text in the circuit that will be flattened and written to a separate file with the "flatcode" extension. Flattening adds global scope to these statements. For example, if you place a Nanosim timing assertion in a cell with the flat code

```
    tv_node_setuphold $(clk) rf $(in) rf 100p 100p
```

and there are 3 instances of the cell, then there will be 3 flattened assertions in the flatcode file:

```
    tv_node_setuphold xtop.xflop1.clk rf xtop.flop1.in rf 100p 100p
    tv_node_setuphold xtop.xflop2.clk rf xtop.flop2.in rf 100p 100p
  tv_node_setuphold xtop.xflop3.clk rf xtop.flop3.in rf 100p 100p
```

If `clk` is actually a single signal that comes from the top level, it is smart enough to recognize this:

```
    tv_node_setuphold clk rf xtop.flop1.in rf 100p 100p
    tv_node_setuphold clk rf xtop.flop2.in rf 100p 100p
    tv_node_setuphold clk rf xtop.flop3.in rf 100p 100p
```