Simulation Standard

Connecting TCAD To Tapeout

A Journal for CAD/CAE Engineers

EDIF Conversion Flow on Gateway

EDIF has been a vital part of the Electronic Design Automation (EDA) industry for many years and Gateway allows users to convert edif200 formatted files of other tools vendors into ***Gateway***'s schematics and symbols.

STEP 1. Importing EDIF

You can import an edif from other tools into ***Gateway*** with a menu, File>>Import. To enable this menu, you have to load a workspace file that includes library paths before importing. We recommend you to make a directory for this flow for example, name "test1" to the directory then copy your edif file and following files from the installation directory.

C:\Silvaco\examples\gateway\2.2.7.R (This is an installation directory)

examples.sws:	workspace file includes library path
spicelib:	basic symbol library from Silvaco
subcircuit:	sample library from Silvaco

In ***Gateway***, you load examples.sws like Figure 1 and then load your edif file with the following menu File>>Import>>EDIF200 as Figure 2. Click the Open, starts a conversion.

Figure2. EDIF import.

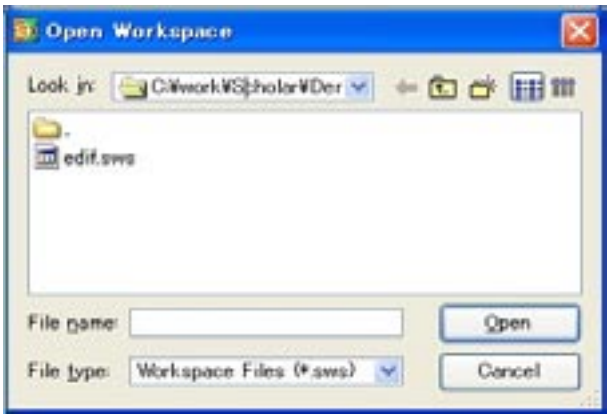

Figure1. Open Workspace.

When the conversion is OK, the successful message will appear. You can see that there are (is) some new libraries, check the test1 directory and you can see that there are (is) new libraries here. Finally, you save the workspace with File>>Save Workspace.

Figure3. Successful

Continued on page 2 ...

INSIDE

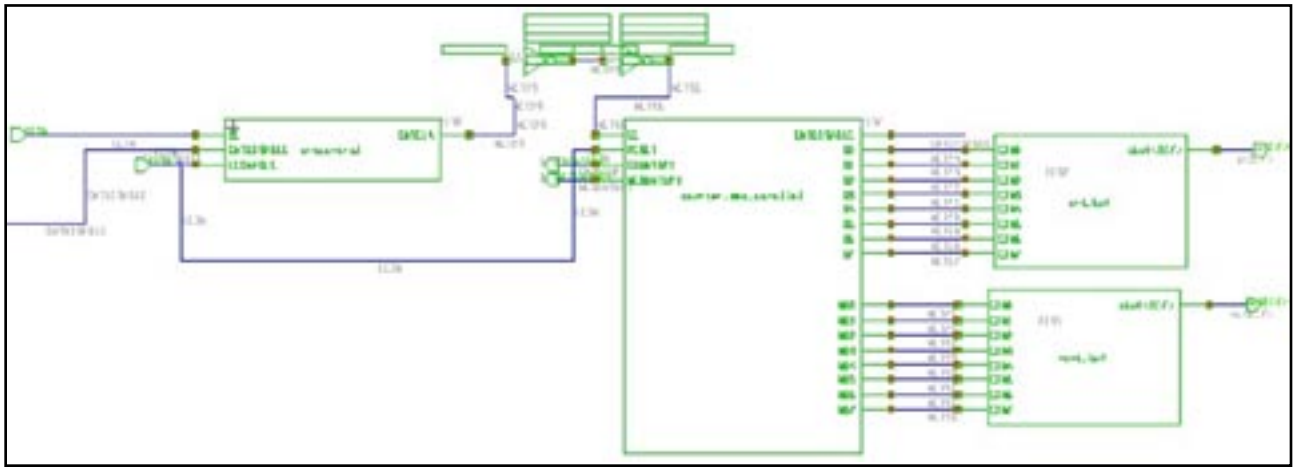

Figure.4 Imported top level schematic

STEP 2. Checking Imported Schematic and Symbols

Generated libraries include symbol files (.body) and schematics (.schlr). Open the top level schematic with File>>Open>>Schematic, and check with your eyes whether its connections, no lack of symbols and inheritance of symbol parameters appear correctly.

STEP 3. Modifying Parameters

You can understand Library directories include symbols, and they are used and referenced in schematics. Each symbol has its own parameters. For example, a RES symbol has a resistance parameter, a MOSFET has W and L and so on. In ***Gateway***, a parameter in a symbol consists of a set of a parameter name and a value like "Name = Value", parameter names are usually set in its symbol file and parameter values are in schematics. Since the symbol after translating into Edif creates by omitting parameter names, users need to add it by hand. An example of nmos4 symbol in the analoglib shows below.

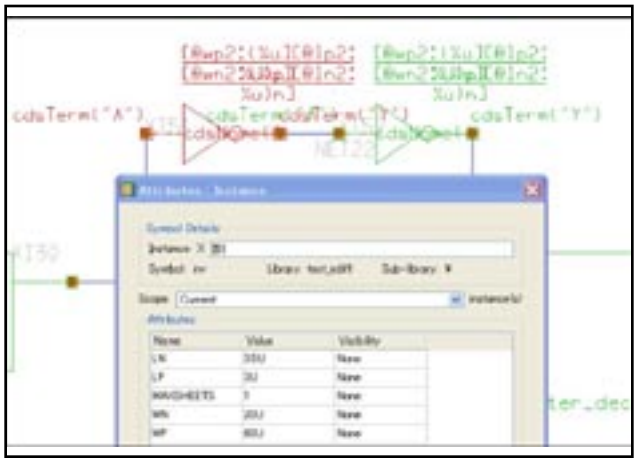

Figure 5.

Open two files, nmos4.body in analoglib and nmos4.body in ***SpiceLib*** with Notepad and so on. and might see less or no "P" lines in analoglib's one. Cut and paste the red descriptions in ***SpiceLib*** 's to at the end of line in analoglib's and save it.

```
V 1
C 40 30 "d" 40 30 0 0 12 0 L
C 40 0 "b" 40 0 0 0 12 0 L
C 0 0 "g" 0 0 0 0 12 0 L
C 40 -30 "s" 40 -30 0 0 12 0 L
L 44 26 36 26 -1 0
L 44 34 44 26 -1 0
L 36 34 44 34 -1 0
L 36 26 36 34 -1 0
L 44 -4 36 -4 -1 0
L 44 4 44 -4 -1 0
L 36 4 44 4 -1 0
L 36 -4 36 4 -1 0
L 4 -4 -4 -4 -1 0
L 4 4 4 -4 -1 0
L -4 4 4 4 -1 0
L -4 -4 -4 4 -1 0
L 44 -34 36 -34 -1 0
L 44 -26 44 -34 -1 0
L 36 -26 44 -26 -1 0
L 36 -34 36 -26 -1 0
L 40 -15 40 -30 -1 0
L 20 -15 40 -15 -1 0
L 20 0 40 0 -1 0
L 15 15 15 -15 -1 0
L 0 0 15 0 -1 0
L 40 -15 30 -20 -1 0
L 30 -10 40 -15 -1 0
L 20 -15 20 15 -1 0
L 40 15 40 30 -1 0
L 20 15 40 15 -1 0
      <-- Paste red descriptions here.
```

Figure 6. nmos4.body in analoglib.

```
L 20 15 40 15 -1 0
L 40 -15 40 -30 -1 0
L 40 15 40 30 -1 0
L 20 -15 20 15 -1 0
L 15 15 15 -15 -1 0
P "SMART_SPICE" "@PREFIX@PATH %D %G %S %B
@MNAME $L $W $AD $AS $PD $PS $NRD $NRS &OFF#
IC=&VDS##, &VGS##, &VBS# $TEMP $DTEMP $M $GEO
$DELVTO" 40 40 0.00 0.00 12 0 0 0 0 0 0 1 0
P "WL" "(W\/L)" 60 -32 0.00 0.00 10 0 0 0 0 0 1 1 0
P "PATH" "?" 62 0 0.00 0.00 12 0 0 0 0 0 1 0 0
P "MNAME" "?" 62 -16 0.00 0.00 10 0 0 0 0 0 1 0 0
P "PREFIX" "M" 40 0 0.00 0.00 12 0 0 0 0 0 0 1 0
P "L" "" 54 -32 0.00 0.00 10 0 0 0 0 0 0 0 0
P "W" "" 54 -48 0.00 0.00 10 0 0 0 0 0 0 0 0
P "AD" "" 62 -47 0.00 0.00 10 0 0 0 0 0 1 0 0
P "AS" "" 62 -63 0.00 0.00 10 0 0 0 0 0 1 0 0
P "PD" "" 62 -79 0.00 0.00 10 0 0 0 0 0 1 0 0
P "PS" "" 62 -95 0.00 0.00 10 0 0 0 0 0 1 0 0
P "NRD" "" 62 -111 0.00 0.00 10 0 0 0 0 0 1 0 0
P "NRS" "" 62 -127 0.00 0.00 10 0 0 0 0 0 1 0 0
P "OFF" "" 62 -143 0.00 0.00 10 0 0 0 0 0 1 0 0
P "VDS" "" 62 -159 0.00 0.00 10 0 0 0 0 0 1 0 0
P "VGS" "" 62 -175 0.00 0.00 10 0 0 0 0 0 1 0 0
P "VBS" "" 62 -191 0.00 0.00 10 0 0 0 0 0 1 0 0
P "TEMP" "" 62 -207 0.00 0.00 10 0 0 0 0 0 1 0 0
P "DTEMP" "" 62 -223 0.00 0.00 10 0 0 0 0 0 1 0 0
P "M" "" 62 -239 0.00 0.00 10 0 0 0 0 0 1 0 0
P "GEO" "" 62 -255 0.00 0.00 10 0 0 0 0 0 1 0 0
P "DELVTO" "" 62 -271 0.00 0.00 10 0 0 0 0 0 1 0 0
```

Figure 7. nmos4.body in spicelib.

In ***Gateway***, open the workspace saved on the Step 1 and then open nmos4.body in analoglib. With the menu Edit>>Properties, you open the edit symbol properties panel and you will see that there are many parameters you added in the body file. In this case, you need to set a model name for MNAME because you have known the fixed model name. Gateway allows users to set a prefix value instead of "?" character for a value. The end of this flow, final checking or netlisting detects this character a value error. You need to do it in this stage.

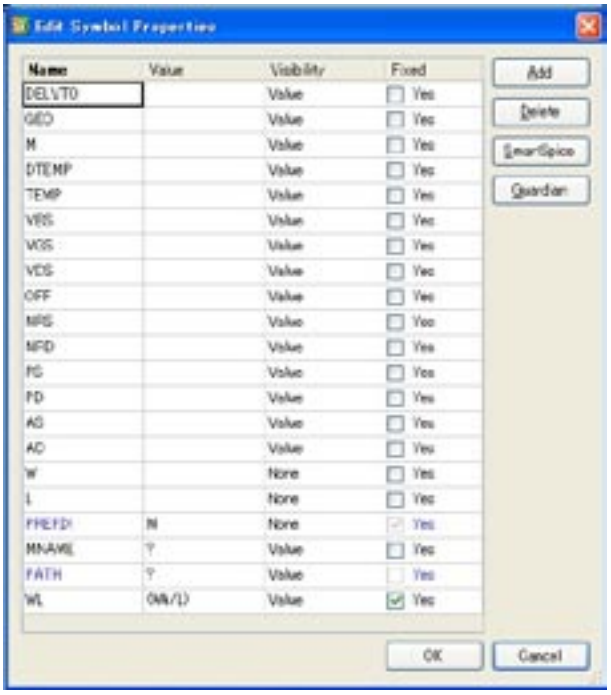

Figure 8. Edit Symbol Properties panel.

Also, click the ***SmartSpice*** button on the panel, and the ***SmartSpice*** String Editor shows you netlist syntax description for SmartSpice.

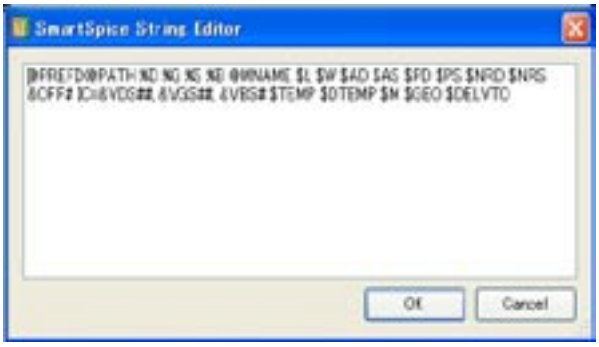

Figure9. SmartSpice String panel.

STEP 4. Open the Top Schematic Again

Now you can reference modified symbols, you open the top level schematic with the workspace file and run a menu Simulation>>View>>Spice Netlist. If no error you get a netlist and create a control deck for simulation.

Conclusion

EDIF200 conversion flow finishes here. The STEP 3 might take a long time for manual modifying, however once modifying whole symbols in the analoglib from competitors, you will reuse them from now on.

HIPEX-CRC Parasitic RC-Network Reducer

Introduction

Design of large scale chips requires precise knowledge of interconnect delays. However, detailed analysis of interconnects may quickly become computationally too expensive due to the distributed nature of the networks, and the large number of internal nodes extracted.

HIPEX-CRC is a parasitic RC-network reduction tool able to reduce the huge number of elements produced by major EDA parasitic extractors, including coupling capacitors, and supports main industry standards parasitic formats.

Advanced algorithm behind ***HIPEX-CRC*** enables to maintain accuracy within a few percents of Spice. ***HIPEX-CRC*** can be directly plugged to the ***HIPEX*** suite, but also can be used as a stand-alone tool.

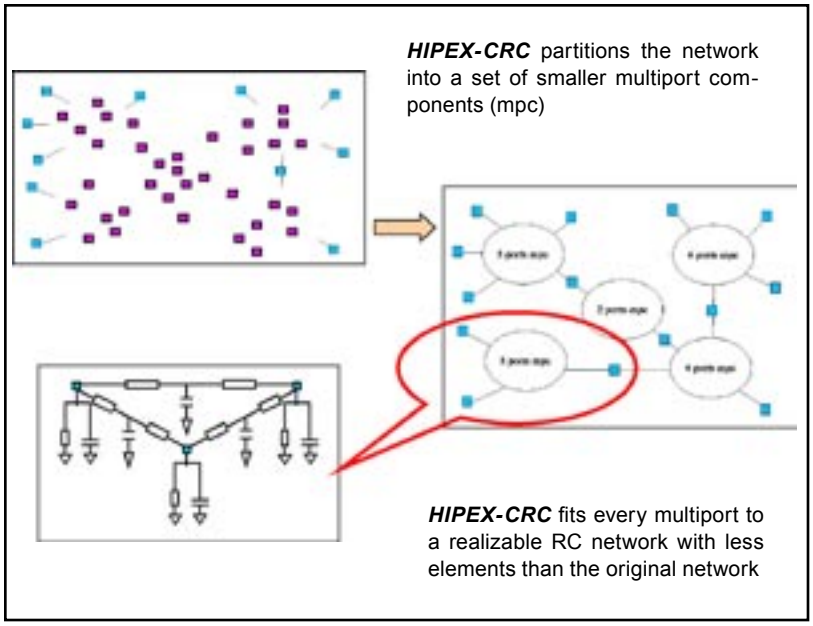

Figure 1.

Scattering-Parameter-Based Macromodeling

- ***HIPEX-CRC*** allows the user to preprocess the circuit, making a first **equivalent reduction** by merging serie-parallel elements and removing dangling elements

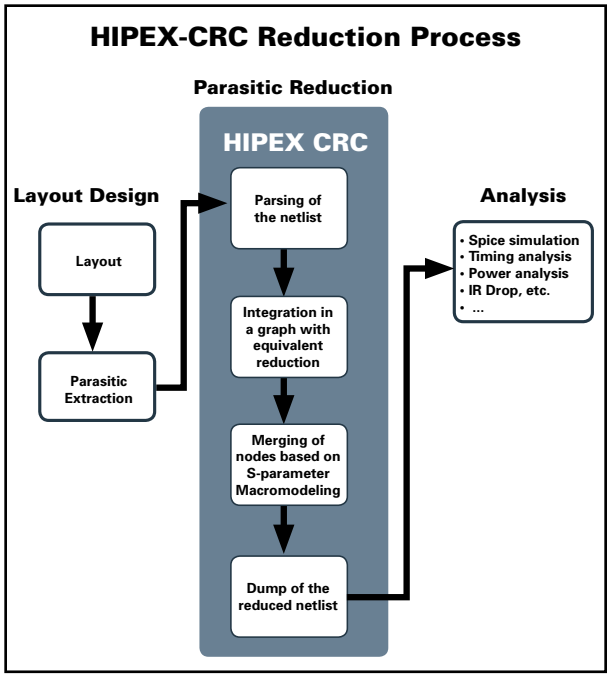

Figure 2. HIPEX-CRC flow.

- Once the preprocessing is done, ***HIPEX-CRC*** relies on a powerful **Scattering-Parameter-Based Macromodeling** [1] reduction technique. The advanced node merging rules within ***HIPEX-CRC*** lead to the partitioning of the original network into a set of several N-port component, each of which is then modeled by a reduced RC circuit characterized by the same set of S-Parameters (Figure 1)
- This permits the analysis of interconnect models other than RC-trees, and therefore, coupling capacitors and resistor loops can be handled without loss of generality
- Also, partitionning of the circuit into small multiport components leads to smaller size matrix computation, saving time and memory, while increasing accuracy
- Output of HIPEX-CRC is thus a realizable, simulable reduced network. Figure 2 depicts the ***HIPEX-CRC*** reduction flow

Key Features

- ***HIPEX-CRC*** supports **SPICE**, **DSPF** and **SPEF** input formats and outputs SPICE, DSPF and SPEF reduced netlists. Possibility is given to the user to output SPEF from DSPF input, as well as output DSPF starting from a SPEF output

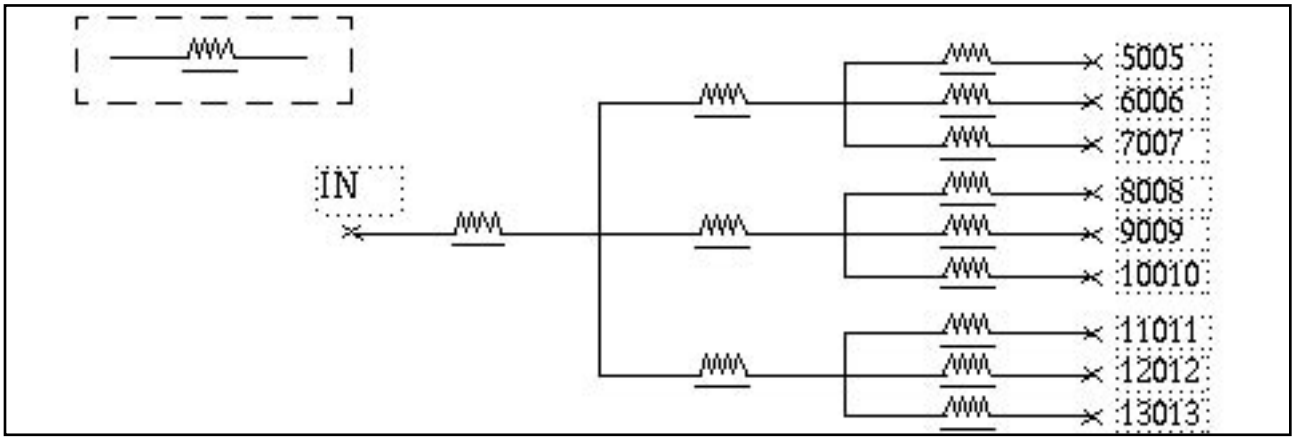

Figure 3.

- ***HIPEX-CRC*** is **fast**: SPICE netlist ranging about 1.5 million parasitic elements can be processed within some 5 minutes on a 64bit-Linux standard machine
- For **enhanced reduction**, ***HIPEX-CRC*** may ignore all parasitic resistances and/or capacitances lower than a user-specified threshold. Also, ***HIPEX-CRC*** may ignore coupling capacitors present in DSPF or SPEF netlists
- For **selective reduction**, ***HIPEX-CRC*** may ignore user specified subcircuits and/or SPF nets.
- For **custom reduction**, ***HIPEX-CRC*** enables the user to specify unreducible nodes, to control topology of the circuit
- Any reduction step performed by ***HIPEX-CRC*** is reported to a summary file (detailed or simple, on the user choice)
- ***HIPEX-CRC*** is easy to use, thanks to user-friendly graphical interface, and flexible ***LISA*** scripting language
- ***HIPEX-CRC*** is available for Unix, Linux32-64bit, Windows

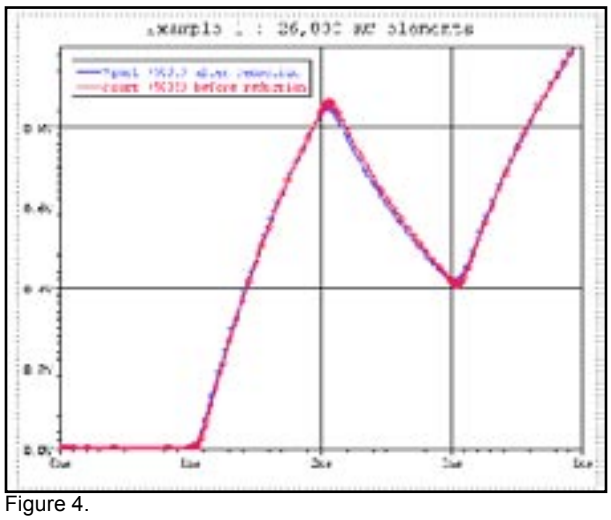
Figure 4.

Examples and Validation

1. Example #1

The RC tree network shown on Figure 3 is made of 26,000 elements (13,000 R and 13,000 C). A set of 10 external ports was specified to carry out the reduction.

Symbol in dashed inset stands for a lumped RC segment of 1000 resistances of 0.16 Ohm, and 1000 capacitances of 5e-4 pF.

A reduction of 99.7 % was achieved on this circuit, with 42 resistances and 32 capacitances in the reduced network.

Figure 4 is the snapshot of a comparative ***SmartSpice*** transient simulation of the original RC tree network versus its reduced equivalent.

Worstcase voltage precision of the reduced circuit lies within 2.5 % from that of original, while simulation time was divided by 400.

2. Example #2

The RC tree network shown on Figure 5 is made of 80,000 elements (40,000 R and 40,000 C).

A set of 6 external ports was specified to carry out the reduction.

Symbol in dashed inset stands this time for a lumped RC segment of 4000 resistances of 0.25 Ohm, and 4000 capacitances of 15e-6 pF.

Again, a reduction of more than 99 % was achieved on this circuit, with only 16 resistances and 15 capacitances in the reduced network.

The comparative ***SmartSpice*** transient simulation of the original RC-tree network versus its reduced equivalent is shown on Figure 6.

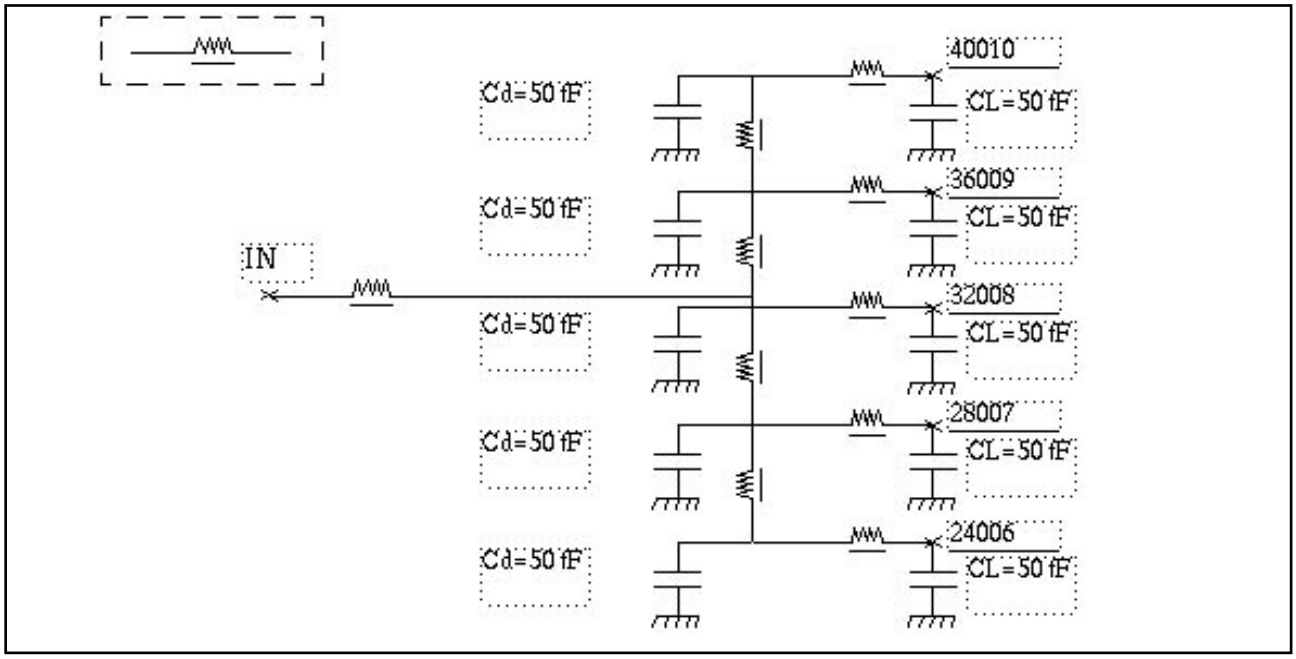

Figure 5.

Though only 6 external ports were specified, the worst-case voltage precision was no less than 2.5 % again, and simulation time was divided by 3000.

3. Example #3

Figure 7 is the snapshot of ***SmartSpice*** transient simulations of reduced and original DSPF formatted industrial networks.

The original network is made of 560 nets, for a total number of 34,464 RC elements.

HIPEX-CRC performed the reduction net by net ; percentage of reduction per net varies from 48% to 82%, according to the topology of the net (regularity, fan-out of internal nodes, ...etc), and the number of instance pins or pins connected to it (the external ports).

Percentage of reduction for the global circuit was found to be 71.6 %, and simulation time could be reduced by 50%.

Conclusion

HIPEX-CRC is an accurate,fast and flexible tool. Based on Scattering Parameter Macromodels, the technique behind guarantees the resultant equivalent network to be fully compatible with general-purpose simulators. Simulation time can be reduced by two or three orders of magnitude, while the response of the reduced circuit is within a few percents of that of the original network.

References

[1] `Partitioning and Reduction of RC Interconnect Networks Based on Scattering Parameter Macromodels', H.Liao and W. Wei-Ming Dai, Computer Engineering, University of California, Santa Cruz.

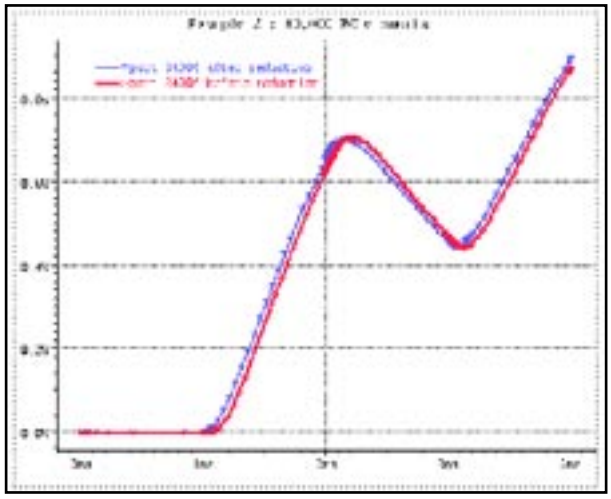

Figure 6.

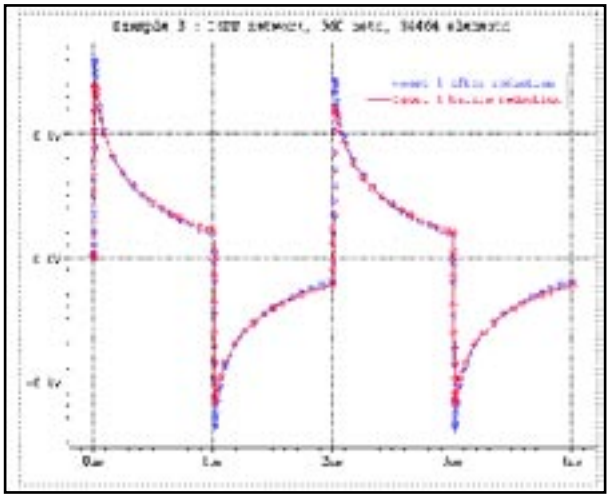

Figure 7.

Generic Devices - The New *HIPEX-NET* Feature for Extraction of Custom Devices

1. Introduction

HIPEX-NET is SILVACO's powerfull hierarchical layout netlist extractor. Built-in extracting rules permit to make fast and accurate parameter extraction for basic set of devices as MOS, BJT, diode, capacitor and resistor. But these rules can't descibe all devices features and parameters that appeare in new submicron or RF layout designs. To resolve this problem the new ***HIPEX-NET*** command ***HIPEX*** GENERIC_DEVICE has been introduced. This new ***HIPEX-NET*** feature gives to user the capability to extract not only additional parameters for standard devices but also custom defined devices with arbitrary set of parameters. This article describes the new ***HIPEX-NET*** command.

2. Generic Device Command

The ***HIPEX*** GENERIC_DEVICE command is intended to define devices that parameters are extracted by user-defined ***LISA*** function. The command has the following definition:

```
HIPEX GENERIC_DEVICE <body>
  /ELEMENT_NAME=<string>
  /MODEL_NAME=<string>
  [/AUX=<aux_layer1>, <aux_layer2>, ... ,<aux_layerN>]
  /PINS={{<pin_layer1>[, <pin_name1>]},
      {<pin_layer2>[, <pin_name2>]},
      ...,
      {<pin_layerM>[, <pin_nameM>]}}
  /FUNC=<string>
  [/FPARAM=<param1>,...,<paramL>]
  [/BY_SHAPE | BY_NET]
  [/SUBCKT_FILENAME=<string>]
```

body is the device recognition layer. Unlike It can be several ***HIPEX*** GENERIC_DEVICE statements with the same recognition layer. These statments should be distinguish by auxiliary and pin layers.

ELEMENT_NAME is device name. It can be PMOS, NMOS, D, and other devices, or also can be arbitrary name. It defines how the device will be reported to netlist. In most cases, generic devices are reported as custom ***SmartSpice*** devices. To report generic device as the specific device into netlist the number and names of pins, have the following values:

- for MOSFET (PMOS, NMOS, MN, MP, ME) and MESFET (PMF, NMF) devices have 3 or 4 pins with names: "D", "G", "S", "SUB". Fourth "SUB" pin is optional.
- for JFET (PJF, NJF) devices have 3 pins with names: "D", "G", "S".
- for BJT (PNP, NPN) devices have 3 or 4 pins with names: "C", "B", "E", "SUB". Fourth "SUB" pin is optional.
- for resistor (R), capacitor (C), diod (D) and inductor (L) devices have 3 pins with names "POS", "NEG" and "SUB". Third "SUB" pin is optional.

If device name is CUSTOM_SUBCKT, the device will be reported as subcircuit instance call.

MODEL_NAME is the model that is output to SPICE. It can be several MODEL_NAMES for one <body> layer.

AUX is the list of auxiliary layers.

PINS is the list of pin layers with pin names. You can omit the pin names for generic devices that should be recognized as transitors, diodes, resistors, capacitors devices. In this case, the order of pin layers is important and the pins will get the following names:

- for MOSFET and MESFET devices, the order is "D","G","S","SUB" that correspond to Drain, Gate, Source, Substrate(Bulk).
- for JFET devices, the order is "D","G","S","SUB" that correspond to Drain, Gate, Source,Substrate(Bulk).
- for BJT devices, the order is "C","B","E","SUB" that correspond to Collector, Base, Emitter, Substrate(Bulk).
- for two pins devices (diodes, resistors, capacitors) with optional third substrate pin, the order is "POS","NEG","SUB" that correspond to first (positive) pin, second(negative) pin, and Substrate.

FUNC is the user-defined ***LISA*** function for device parameter calculation.

FPARAM is a list of FUNC ***LISA*** function parameters.

BY_SHAPE or BY_NET is the option that defines how to separate pins.

SUBCKT_FILENAME is a file name with subcircuit definition. This option works for device with ELEMENT_NAME=CUSTOM_SUBCKT. In this case, the .INCLUDE statement is output to netlist instead of .MODEL statement for this device.

There are several built-in ***LISA*** functions, that you can use for device parameter calculation in your ***LISA*** procedure.

3. Built-In LISA Functions

DEVICE_AREA(<layer1_name>|<pin1_name>, <layer2_name>|<pin2_name>)

This function calculates the overlaping area between shapes from two layers or pins. If second parameter is empty line (“”), the function calculates area of the shape from first layer or pin.

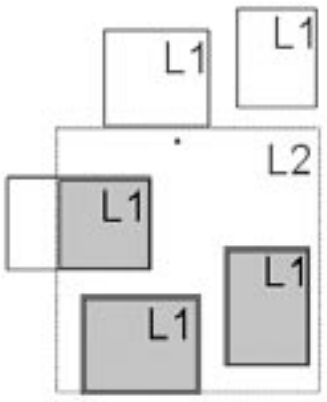

Figure 1. Calculated area in DEVICE_AREA(“L1”,”L2”)

DEVICE_PERIMETER(RELATION, <layer1_name|pin1_name>, <layer2_name>|<pin2_name>)

This function calculates the length of the boundaries for the shapes from layers or pins that are inputted as parameters.

RELATION defines the relation between shapes from two layers and it can take the following predefined values:

- REL_NONE: The second layer or pin should be missed (empty string “” is used), and the function calculates perimeter of the shape from the first layer or pin.
- REL_INSIDE: The length of edges of shape from first layer that is strictly the INSIDE shape from second layer.

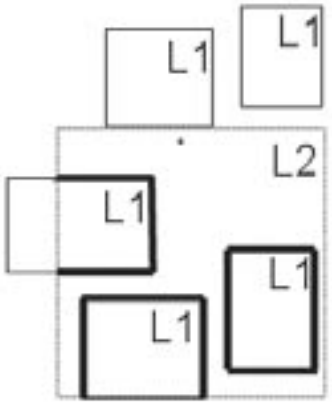

Figure 2. Edges in DEVICE_PERIMETER (REL_INSIDE,“L1”,”L2”).

- REL_OUTSIDE: The length of edges of shape from first layer that is strictly the OUTSIDE shape from second layer.

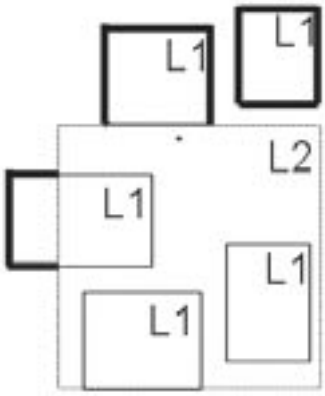

Figure 3. Edges in DEVICE_PERIMETER (REL_OUTSIDE, “L1”,”L2”)

- REL_COINCIDENT: The length of common edges of shape from first layer that is strictly the INSIDE shape from second layer.

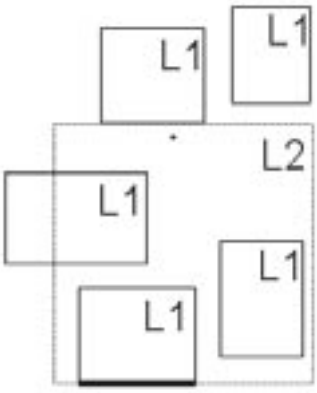

Figure 4. Edges in DEVICE_PERIMETER (REL_COINCIDENT, “L1”,”L2”)

- REL_BUTTING: The length of common edges of shape from first layer that is strictly the OUTSIDE shape from second layer.

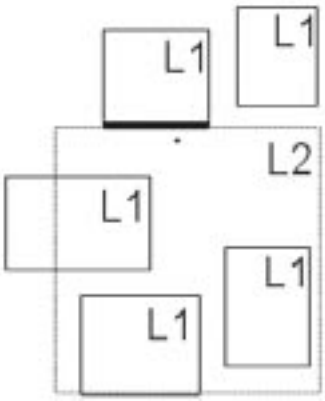

Figure 5. Edges in DEVICE_PERIMETER (REL_BUTTING,“L1”,”L2”)

- REL_COINCIDENT+REL_BUTTING: This is used for calculation the length of common edges for both shapes.

DEVICE_BENDS(<layer_name>|<pin_name>)

This functions calculates the bends value for the shape. The bends value is the summing angle in degrees by which the perimeter changes direction in all concave vertices of the shape and dividing by 90∞.

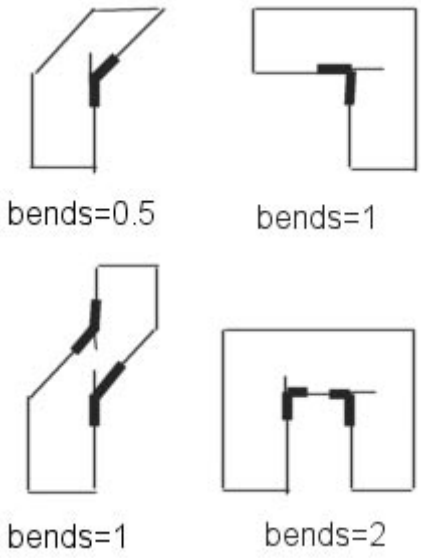

Figure 6. Bends computation.

DEVICE_COUNT(<layer_name>|<pin_name>)

This function returns number of separate shapes from aux or pin layer or pin.

DEVICE_PIN_NET(<pin_name>)

This function returns net name of <pin_name>.

DEVICE_SET_PROPERTY(<property_name>, <property_value>)

This function set the device attribute in output netlists in form <property_name>=<property_value>. <property_name> is the string and <property_value> is the number.

4. Examples of Generic Device Definitions.

In example below, one layer P-Gate is used for recognition two PMOSes. First, one will be recognized only when P-Gate layer shape has overlapped or touched the Aux_gate layer shape. The pin names will have default values. It will be reported to SPICE netlist as PMOS with model name P0. The procedure PGATE calculates geometrical parameters for this PMOS: L, W, PD, PS, AD, AS that will be reported to SPICE file too. Second, one will be recognized when P-Gate layer shape does not overlap or touch the Aux_gate layer shape. The model name for it is P1 and pin names have explicit definition.

PMOS device definitions:

```
hipex generic_device P-Gate
    /element_name=MP
    /auxs=Aux_gate
    /pins= P-SD, Poly, P-SD, NWell
    /func=PGATE0
    /model_name=P0;


hipex generic_device P-Gate
    /element_name=MP
    /pins=({"Poly", "G"}, {"P-SD","S"},
     {"P-SD","D"}, {"NWELL, "SUB"})
    /func=PGATE
    /model_name=P1;
```

Example of Lisa Function:

```
define procedure PGATE
do begin
    W = 0.0;
    L = 0.0;
    PD = 0.0;
    PS = 0.0;
    AS = 0.0;
    AD = 0.0;
! Calculation of gate area
    AREA = device_area("P-Gate", "");
    W1 = device_perimeter(REL_BUTTING, "P-Gate", "S");
    W2 = device_perimeter(REL_BUTTING, "P-Gate", "D");
! Calculation of gate width and length
    W = (W1 + W2) / 2 ;
    IF (W NEQ 0.0) THEN (L = AREA / W);
! Calculation of source, drain areas and perimeters
    AD = device_area("D", "");
    AS = device_area("S", "");
    PD = device_perimeter(REL_NONE, "D", "");
    PS = device_perimeter(REL_NONE, "S", "");
! Transfer of calculated parameters to c-code
    device_set_property("L", L);
    device_set_property("W", W);
    device_set_property("PD", PD);
    device_set_property("PS", PS);
    device_set_property("AD", AD);
    device_set_property("AS", AS);
end;
```

In the following example, the ***LISA*** function is defined with parameters. Therefore, you can use the same function for a set of devices that contain different layers in its definitions. L and W will be output as MOS parameters.

Device Definition:

```
hipex generic_device "nmos"
    /pins={{"nsd", "S"}, {"pc_nodev", "G"},
    {"nsd", "D"}, {"psub_analog", "SUB"}}
    /auxs={"rx", "gate_dc"}
    /element_name="NMOS"
    /model_name="nmos"
    /func="MOS_PROPERTIES"
    /fparams={{"nmos","nsd",0}};
```

LISA Function:

```
DEFINE PROCEDURE MOS_PROPERTIES
PARAMETER mos_seed
PARAMETER sd_seed
PARAMETER bend_effect
DO BEGIN
    AREA = device_area((mos_seed), "");
    W = device_perimeter(REL_BUTTING, (mos_seed),
    (sd_seed)) / 2;
    IF (W GTR 0) THEN (L = AREA / W)
    ELSE (L = SQRT(AREA));
    bends = device_bends((mos_seed));
    IF (bends GTR 0) THEN ( W = W - bends *
    bend_effect * L ) ;
    device_set_property("L", L);
    device_set_property("W", W);
END;
```

In the example below, the inductor will be recognized and reported to netlist as the instance of "spiral_std" subcircuit with four parameters NR, W, S, and RAD.

Device Definition:

```
hipex generic_device "inddev"
    /pins={{"METAL", "PIN1"}, {"indpin", "PIN2"},
    {"substrate", "PIN3"}}
    /auxs={"ind_hole", "ind_bottom", "gaps_not_end",
    "segs_not_end", "segs"}
    /func="SPIRAL"
    /element_name="CUSTOM_SUBCKT"
    /model_name="spiral_std";
```

LISA Function:

```
define procedure SPIRAL
do begin
      nr = 0.0;
      W = 0.0;
      S = 0.0;
      rad = 0.0;
      nr = device_count("ind_bottom") - 0.5;
      rad = device_perimeter(REL_NONE, "ind_hole",
      "") / 8;
      segs_cnt = device_count("segs");
      segs_peri_int = device_perimeter(REL_INSIDE,
      "segs", "inddev");
      W = segs_peri_int/(2*segs_cnt-1);
      gaps_cnt = device_count("gaps_not_end");
      gaps_peri_sp = device_perimeter(REL_NONE,
      "gaps_not_end", "") -
      device_perimeter(REL_BUTTING+REL_COINCIDENT,
      "gaps_not_end", "inddev");
      S=gaps_peri_sp/(2*gaps_cnt);
      device_set_property("NR", nr);
      device_set_property("W", W);
      device_set_property("S", S);
      device_set_property("RAD", rad);
end;
```

5. Conclusion

The new ***HIPEX-NET*** feature extents the capability of code. Generic device command gives users the possibility to extract new parameters and new custom devices for submicron or RF layouts.

Calendar of Events

September

1	
2	SISPAD - Munich, Germany
3	SISPAD - Munich, Germany
4	SISPAD - Munich, Germany
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	ESSDERC - Leuven, Belgium
21	ESSDERC - Leuven, Belgium
22	ESSDERC - Leuven, Belgium
23	ESSDERC - Leuven, Belgium
24	ESSDERC - Leuven, Belgium
25	
26	
27	
28	
29	
30	
31	

October

1	
2	
3	
4	IEEE SOI Conf. - Charleston, SC FSA Modeling WS - Santa Clara, CA
5	IEEE SOI Conf. - Charleston, SC FSA Modeling WS - Santa Clara, CA
6	IEEE SOI Conf. - Charleston, SC FSA Suppliers Expo - San Jose, CA
7	IEEE SOI Conf. - Charleston, SC
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	IEEE Compound Symposium - Monterey, CA
25	IEEE Compound Symposium - Monterey, CA
26	IEEE Compound Symposium - Monterey, CA
27	IEEE Compound Symposium - Monterey, CA
28	
29	
30	

Bulletin Board

Silvaco Joins Accellera

Due to the new focus on the Verilog Hardware Description Language with the acquisition of the assets of Simucad Inc., Silvaco has joined Accellera. The SILOS Verilog Simulator, ***Hyperfault*** Mixed-Level Fault Simulator and ***Harmony*** Mixed-Signal Simulator all rely on the stability of Verilog.

To improve designers' productivity, the electronic design industry needs a methodology based on both worldwide standards and open interfaces. Accellera was formed in 2000 through the unification of Open Verilog International and VHDL International to focus on identifying new standards, development of standards and formats, and to foster the adoption of new methodologies.

Accellera's mission is to drive worldwide development and use of standards required by systems, semiconductor and design tools companies, which enhance a language-based design automation process. Its Board of Directors guides all the operations and activities of the organization and is comprised of representatives from ASIC manufacturers, systems companies and design tool vendors.

If you would like more information or to register for one of our our workshops, please check our web site at ***http://www.silvaco.com***

Hints, Tips and Solutions

SILVACO PDK Development Group

Q. How can I become a more efficient user of *Expert*?

A. Maximize my layout viewing area Default screen layout with many toolbars around is sometimes inconvenient; layout area can be maximized as desired by the user. You may switch control bars on/off. Layer Bar is resizable, both in height and in width. Also toolbars can be docked/undocked or dragged into another location or completely turn off depending upon user's desire.

Window Arrangements To simply the management of opened windows, the Window submenu provides features that automatically arrange windows to suit your needs into different arrangements i.e Cascade, Tile, Tile Vertically and Tile horizontally.

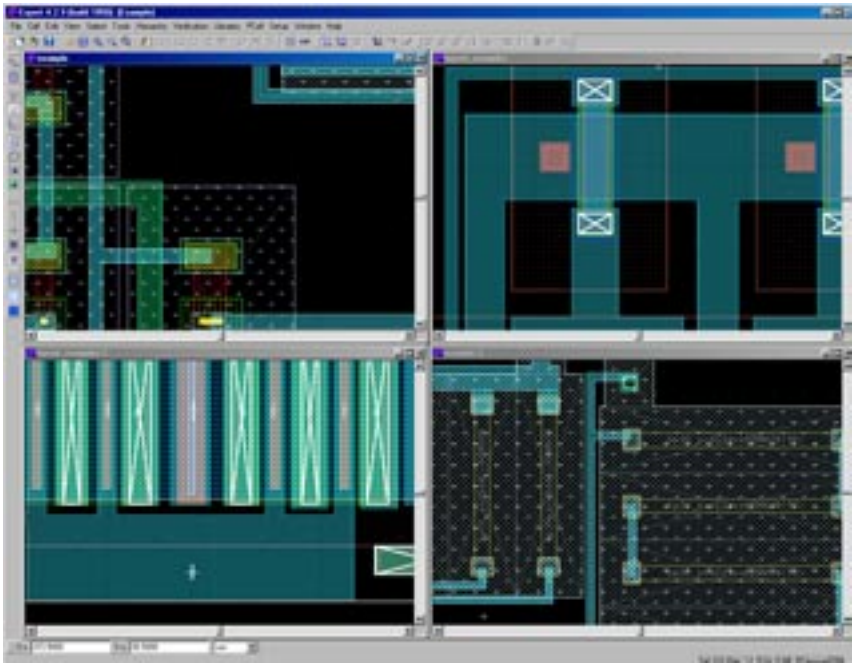

Figure 1. Tile Method.

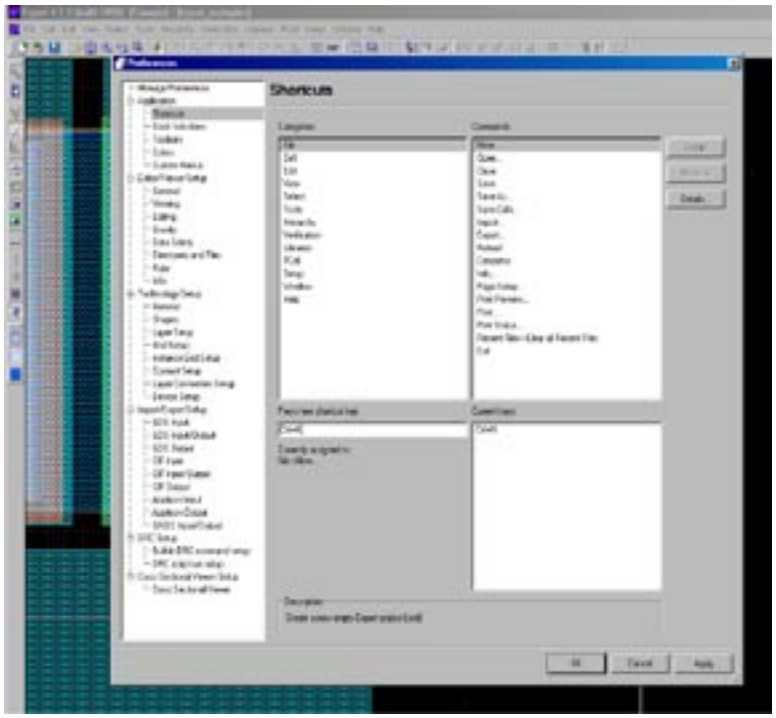

Figure 2. Shortcuts Menu.

Shortcuts It is possible to assign keyboard shortcuts to all menu commands, so that the commands maybe quickly executed by hitting the associated key instead of selecting them from Menu. Also if you click the right-mouse button inside the layout window, the custom menu appears and it contains a user-definable list of commands.

Scripting repetitive tasks Repetitive tasks or a series of commands can be easily be scripted using ***LISA*** and can be executed when desired, hence saving time by automating the execution of those commands rather than individually executing them every time.

Edit in Place allows the user to select and edit an instance of a cell at any level of hierarchy within the current cell which is open in ***Expert***. During Edit in place session you may enter into a cell instance, then enter into another instance what belongs to first instance and so on, going deeper and deeper into hierarchy.

Q. How can I navigate through my layout?

a. Project Tree:

Project Tree (Hierarchy→ Explorer→ Proj ect Tree) helps to view the libraries being used in addition to the actual hierarchy of a specific cell being used in a layout. This is a convenient tool for fast loading, editing, or inspecting any desired cell instance or primitives at any level of hierarchy.

b. Cell hierarchy/explode heirarchy features/create cell in place:

Cell Hierarchy is a powerful tool in EXPERT that makes it possible for a user-created logical categorizing of layout into subparts thus giving it a full hierarchical mode of control. The key operations are recursive Edit-in-place, hierarchical view settings, hierarchy flatten (or explode), and grouping into cells. Edit-in-place gives an easy method of editing selected objects within the surroundings of any of its instance.

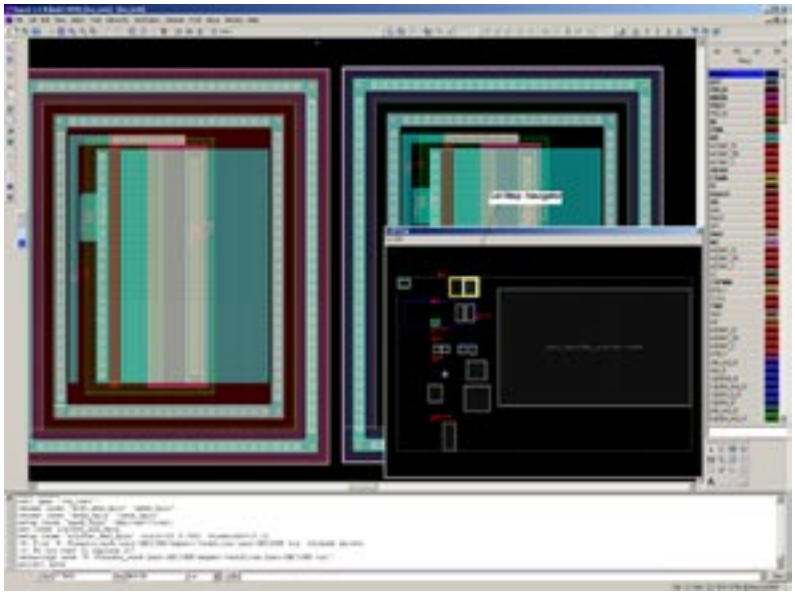

Figure 3. Navigating using Cell Map 1.

c. Flat view/lazy view:

The **View→ Cell** view submenu controls the degree of details in the layout being edited. There are several options provided by ***Expert*** to change the contents of the layout's view or to make parts of the geometry hidden. In **View→ Cell view→ Flat** mode enables the total and real geometry of the cells to be displayed. The cell layout in fact looks like the hierarchy flattened. In **View→ Cell view→ Lazy** mode, only the primitives/instances/arrays are displayed as hatched boxes (with their insides hidden and only the top level of hierarchy displayed). Some other modes of view like the **Floorplan mode, primitives' mode, Edited Cell Frame, Instances/array Frame** etc.

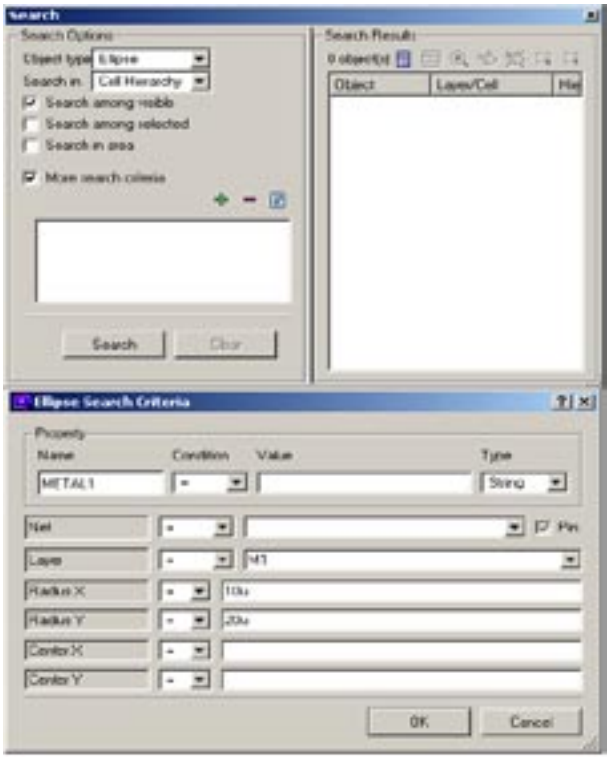

Figure 4. Using the Search Feature 1.

d. Cell map:

By enabling the cell map (View→ Dock windows→ cell map), a small additional window of the layout is shown. It has a small navigational box (as shown in Figure 3) that is convenient for resizing/focusing specific part of a big layout. It allows the user to view the layout at different resolutions and to globally monitor the changes made on the locally viewed area.

e. Search feature:

In order to better identify specific layers of any shape (box, eclipse, or region), or search for specific wire, text, instances, arrays in a Cell hierarchy: there is a *"Search"* option in the menu (Edit→ Search).

Call for Questions

If you have hints, tips, solutions or questions to contribute, please contact our Applications and Support Department
Phone: (408) 567-1000 Fax: (408) 496-6080
e-mail: support@silvaco.com

Hints, Tips and Solutions Archive

Check our our Web Page to see more details of this example plus an archive of previous Hints, Tips, and Solutions
www.silvaco.com

Your Investment is Safe

20 Years and Growing

Financially Rock-Solid

Fiercely Independent

We are NOT For Sale

SILVACO
INTERNATIONAL

USA Headquarters:

Silvaco International
4701 Patrick Henry Drive, Bldg. 2
Santa Clara, CA 95054 USA

Phone: 408-567-1000
Fax: 408-496-6080

sales@silvaco.com
www.silvaco.com

Contacts:

Silvaco Japan
jpsales@silvaco.com

Silvaco Korea
krsales@silvaco.com

Silvaco Taiwan
twsales@silvaco.com

Silvaco Singapore
sgsales@silvaco.com

Silvaco UK
uksales@silvaco.com

Silvaco France
frsales@silvaco.com

Silvaco Germany
desales@silvaco.com

*Products Licensed through Silvaco or e*ECAD*