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.

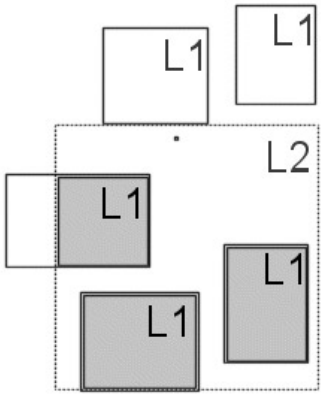

Figure 1. Calculated area in DEVICE_AREA(“L1”,”L2”)

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.

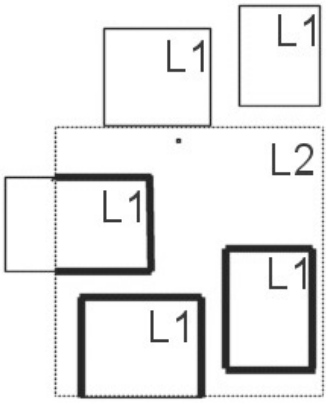

Figure 2. Edges in DEVICE_PERIMETER (REL_INSIDE,“L1”,”L2”).

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

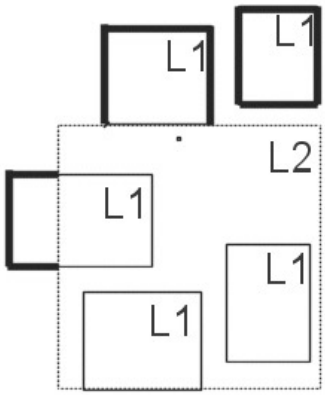

Figure 3. Edges in DEVICE_PERIMETER (REL_OUTSIDE, “L1”,”L2”)

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

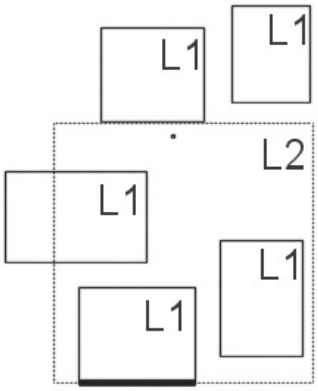

Figure 4. Edges in DEVICE_PERIMETER (REL_COINCIDENT, “L1”,”L2”)

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

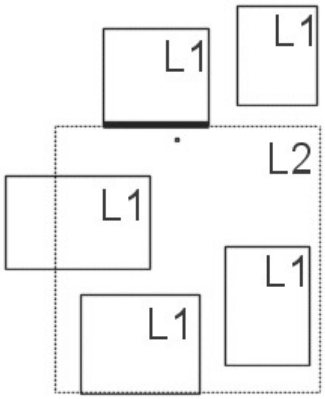

Figure 5. Edges in DEVICE_PERIMETER (REL_BUTTING,“L1”,”L2”)

- 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∞.

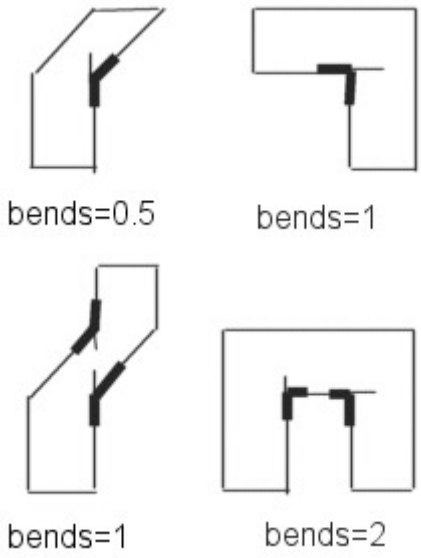

Figure 6. Bends computation.

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.