REX - A VLSI Parasitic Extraction Tool for Electromigration and Signal Analysis

Jerry P. Hwang
Digital Equipment Corporation
77 Reed Road
Hudson MA 01749

ABSTRACT

REX is a program that extracts parasitic resistance and capacitance values for nodes in VLSI layouts. REX also performs network serial and parallel simplifications. Two types of netlists are available, the SPICE-format for general purpose circuit simulation and a specially designed format for easy interface to internally developed analysis programs. The generated netlist contains information specifying where each resistor and capacitor is physically located in the original artwork. A graphical representation of the resistance and capacitance network is generated for overlaying with the mask layout for visual verification. REX is used in conjunction with a suite of analysis programs to perform engineering checks. Electromigration, power bus resistance, clock skew, noise analysis, and latch-up of CMOS VLSI chips are among the checks that utilize REX results.

1 Introduction

As VLSI CMOS design advances to sub-micron technology, there is an apparent need for tools to verify electromigration phenomena, power bus voltage drop, and noise analysis. Those phenomena are due to the parasitic resistances of the circuit, which is often a neglected factor in the VLSI layout verification process. It is also desirable to verify clock and signal lines for possible timing skews, due to the RC delay of long wires, directly from the layout.

Parasitic resistances can substantially affect circuit performance, but are difficult to extract efficiently from the layout. For chips with millions of devices, hundreds of thousands of load points on the power busses can be expected, where each load point is a connection to a MOS device. The complexities of the resistance network formed by the power busses create a need for an effective parasitic extraction algorithm.

The most accurate algorithm for resistance extraction is to solve Laplace's equation with boundary conditions numerically [Chaw70] [Bark85]. But this approach is very slow and, therefore, not appropriate for VLSI layout extraction purposes. The simplest approach is the lumped approximation method: each node in a circuit is assigned a resistance by approximating the node's layout perimeter and area using simple arithmetic calculations [Scot85]. Although the lumped approximation method is fast, it is less accurate and not suitable for applications that require representation of the electrical node as a resistor network instead of one resistance element.

Another approach is the polygon decomposition method, which represents a tradeoff between speed and accuracy [Horo83] [Star87] [Hall87]. Although somewhat expensive for extracting a large design, it yields reasonable results provided that good optimization algorithms and suitable data structures are utilized.

A polygon decomposition algorithm is outlined in [Horo83], but no implementation is reported. The resistance extraction program in [Star87] is suitable for electromigration analysis of power and ground nodes, but non-Manhattan geometry extraction is not supported, and it yields a different extraction result if the geometry is rotated. The simplified extraction algorithm in [Hall87] is effectively used in an interactive design environment for designing a chip section, but its suitability for extracting a full chip layout is not stated.

This paper presents a resistance extraction method based on a polygon decomposition approach. The heuristic algorithm is simple enough to extract large circuits within a reasonable time, yet it is sophisticated enough to be able to extract nodal parasitics in a complicated, million device chip with sufficient accuracy. The restriction to Manhattan geometries typical of previous work also is eliminated.

2 REX System Overview

An integrated environment of resistance/capacitance extractor, circuit analysis, and graphical display is essential for analyzing electromigration phenomena and signal integrity of long interconnects. Fig. 1 shows the system diagram of an integrated VLSI design engineering checking system in use at Digital's Semiconductor Engineering Group.

Node and device layout geometries and the chip transistor netlist are extracted from the chip layout by our Interconnect Verifier, IV. REX uses the node geometries (and optionally, the device geometries) together with the technology information to generate a resistor and

capacitor network and other information for circuit analysis. A schematic-like diagram of the RC network can be optionally generated by REX for display by our layout editor. Page 718

MEGAN. A set of circuit analyzers, including VIP for electromigration checking and clock skew analysis, TALENT for power/ground noise analysis and latch-up checks, and SPICE for circuit simulation, utilizes the RC network generated from REX and the chip netlist from IV to analyze potential circuit problems. These include the current density and voltage drop on various critical points of the power and ground nodes, or RC delay on specified signal nodes. Each RC element generated by REX contains physical coordinates so the analyzer can report the locations of the spots that violate the electrical design rules. The locations can be found using the layout editor on either the extracted resistor/capacitor schematic or the original chip layout.

[Figure 1]

Fig. 1 An integrated engineering check system

REX is used for extracting the resistance and capacitance on a node of a VLSI chip. The types of nodes under consideration are power and ground, clocks, and other complex nodes that the circuit analyzer has determined require further RC analysis. In addition to the RC network, REX's netlist file also contains transistor and well association information. The transistor association information specifies a node in the RC network as a source/drain or gate terminal for a specific transistor of the chip. Well association groups together all the nodes within a well region so a latch-up analysis can be performed by the latch-up analyzer.

The program consists of the following modules: (1) technology file reader, (2) layout file reader, (3) polygon fracturing module, (4) polygon search and processing module, (5) RC calculation module, and (6) netlist/geometry output module.

The technology file contains the resistivity and capacitance data for each layer. It also contains information about the contact layer connectivity and link resistance information.

The layout reader translates the layout DLF file (Digital Layout Format, an extension of CIF format) into internal polygon data structures. It also constructs KD-tree [Bent75] [Rose85] data structures for polygons to facilitate searches.

The polygon fracturing module decomposes a complex polygon into path elements. A path element is a polygon without convex corners. Subnodes are inserted into path elements prior to and after fracturing depending on the type of subnode. A subnode is a node of the extracted RC network, and is created where discontinuities in the geometry are found, such as a junction or a contact site.

The polygon search and processing module provides routines to search for abutment or overlapping conditions between polygons, and it performs polygon intersection and merging operations. Since KD-tree data structures are attached to polygons, these polygons with bounding boxes not touching the object under consideration are filtered out during abutment, overlapping, and merging operations, thus speeding up polygon search and processing.

The resistance and capacitance calculation module determines resistances of path elements and contact sites, and capacitance on the subnodes. Two types of resistance exist: the path element resistance due to the sheet resistivity of the conducting layer and the resistance of the contact layer due to the link resistance of the contact. The two are modeled differently in the technology file and are calculated accordingly.

The output module generates the RC netlist and the network schematic. It also performs series and parallel reduction of the RC network.

The resistance and capacitance extraction procedure of REX is as follows:

```
RC_extraction(node,devices) {
Read_technology_file();
Read_node_geometry(node);
Read_device_geometry(devices);
Construct_KD_tree();
Merge_polygons();
Mark_device_abutment(node,devices);
Fracture_polygon();
Mark_contact_subnode();
Calculate_resistance();
Calculate_subnode_capacitance();
Output_netlist();
Output_RC_schematics();
}
```

The program starts by reading the geometry files and constructing polygon and KD data structures. The polygons are merged before they

are fractured to avoid dangling elements due to split polygons. The subnodes are marked at various stages: the device subnodes are identified before fracturing, the break line subnodes are inserted during fracturing, contact subnodes are located afterwards. After the polygons are fractured and subnodes inserted, the resistance and capacitance can be calculated and a netlist generated. The routines outlined outlined in the above procedure will be further explained in the following sections. Page 719

3 Polygon Fracturing

The fracturing procedure adds break lines at convex corners of the polygon such that each break line is perpendicular to the direction of current flow direction in the polygon. After all the break-lines are inserted in the polygon, the resistance of the fractured path elements can be constructed from the break-lines and the other sub-nodes marked prior to the polygon fracturing stage.

The accuracy of the resistance calculated for the fractured element can be maximized if the break line is made on an equi-potential line, which is perpendicular to the current flow of a path element but may not always be a straight line [Horo83]. The initial current flow direction is determined by the device abutment condition, which was marked prior to the fracturing stage. It is further determined by the length of the two edges joining the convex corner and the distance of the two virtual lines reaching the two opposite edges from the corner. The device abutment condition takes precedence to the edge distance condition when determining the current flow.

The polygon fracturing algorithm utilized by REX is as follows:

```
Fracture_Polygon() {
  sort polygon edges into vertical, horizontal, and angled
edges.
  sort vertices into concave and convex vertices lists
  if the polygon doesn't have a convex corner, return.
  Set current_flow=PERPENDICULAR to device
subnodes.
  for each convex_vertex {
    get the two edges that join the convex_vertex,
    extend edges from vertex to opposite edge of polygon
to form two virtual segments.
    if current_flow of any edge has been set to PERPEN-
DICULAR,
      make corresponding virtual segment a break line.
    else if both edges are same length then
      make shorter virtual segment a break line,
    else make virtual segment corresponding to shorter
edge a break line.
  }
  for each break line {
      intersect break line with polygon edges.
      split the polygon edge at intersection.
  }
  for each pair of intersected break lines
      split the break lines at intersection.
  current_segment = first edge of polygon.
  traversal_direction = FORWARD
  do until all polygon edges are traversed once and all
break lines are traversed at both directions {
      next_segment = segment forms most acute angle
with current_segment and not traversed in current direc-
tion.
      if segments form a closed polygon {
         store the polygon as a path element.
         current_segment = next edge not yet traversed
or next break line not yet traversed at both directions.
         traversal_direction = to be traversed direction
of current_segment.}
    }
}
```

Note that in a closed polygon, the number of convex corners is less than the number of concave corners. When classifying vertices, polygon edges are arranged as directional vectors. The direction of an edge is determined by the direction in which the edge is traversed. The sign of the product term of the two vectors can be used to classify the vertex. The two edges joining vertex (x0,y0) illustrated as directional vectors, are shown in Fig. 2. The cross product of the two vectors z = dx1 * dy2 - dx2 * dy1 is calculated, and the sign of z is used to determine the class of the vertices. The vertices with negative cross product sign are separated from those with positive sign. By counting the two classes, the vertices can be classified accordingly.

[Figure 2]

Fig. 2 Edges of a vertex

Shown in Fig. 3 is a complex polygon fractured into path elements. The subnodes, including contacts and break lines, are marked on each path element.

[Figure 3]

Fig. 3 Example polygon fractured into path elements

4 Subnode Insertion and Association

There are four types of subnodes: device subnodes, breakline subnodes, contact subnodes and dummy subnodes. Device subnodes are inserted prior to polygon fracturing, breakline subnodes are inserted during the fracturing process, contact subnodes are inserted after a polygon is completely fractured, and dummy subnodes are inserted during the resistance calculation stage.

The device subnodes are identified for the polygons on polysilicon and diffusion layers where abutment with device Page 720 regions occur. The fracturing routine uses this information to determine the current flow direction in a polygon.

The contacts that link two conducting layers are located and grouped if they are within the same overlapping region of two conducting layer path elements. For each contact site, two subnodes are marked on the same coordinate, one on each of the two conducting layers connected by the contact. The center of the contact or the contact group is marked as the subnode location.

The following algorithms show the procedures for determining the device subnode prior to fracturing, and for marking contact subnodes on the path elements of conducting layers.

```
Mark_device_abutment(node,devices) {
For each node polygon do
  device polygons = KD_search (device geometries with
bounding boxes intercepting the node polygon)
 For each device polygon returned by KD_search do
    if check_abutment(node polygon, device polygon)
    mark the edge on node polygon as a device subnode.
}
Mark_contact_subnode() {
For each contact layer do {
  i=upper conducting layer linked by the contact layer
  j=lower conducting layer linked by the contact layer
  overlap regions = AND (element (layer[i]), element
(layer[j]))
  Do KD_search to get all contacts within the overlap
regions
  Mark the geometry center of the contacts to element
(layer[i]) and element(layer[j]) as a contact subnode.}
}
```

With KD tree data structures and KD searches, a lot of unnecessary polygon operations are eliminated, since only those elements whose minimum bounding boxes intersect each other are returned by the KD search routine for further detailed operations, such as AND, OR, CHECK_ABUTMENT, and so forth.

REX performs device association with all of the devices on a chip if the device information is given. The device file contains the geometries of the MOS transistor region (overlapping of diffusion and polysilicon layer) with the assigned device name marked as text on each device geometry. For the device subnode, REX uses the device text from the device geometry instead of the coordinate of the subnode. Thus a device subnode can be distinguished with other types of subnodes and the analyzer can use those names to look up the proper devices from the chip netlist. REX performs the device association by locating the abutment of device polygons with diffusion and polysilicon polygons. The device text on diffusion layer resistors is interpreted as source/drain nodes of the device, whereas the device text on polysilicon layer resistor is interpreted as a gate node of the device.

The analyzer uses the chip netlist for its source of device information to model the loading information. The loadings translated from the devices are connected to the REX extracted RC network through the device association subnodes in the RC network.

In order to analyze the latch-up phenomena, the voltage difference of any plugs within the same well are checked against a threshold voltage. By grouping together the subnodes on the well plug layer within the same well, the analyzer's comparison of the voltage drop within the same group is made easier, without the need to read the layout file.

5 Path Element Resistance/Capacitance Calculation

There are two types of resistance to be calculated independently. The path element resistances are the "horizontal" resistors due to sheet resistivity of the conducting layers; the contact resistances are the "vertical" resistors that connect two conducting layers.

After fracturing polygons into path elements with subnodes attached to them, the resistance of each element can be calculated depending on the number of subnodes. For elements with only one subnode, another dummy subnode is created on the furthest opposite edge to form a two-subnode element. The resistance of two-subnode path elements is calculated by the equation: R = [rho]L/W where [rho] is the sheet resistivity of the layer, and L and W are the length and width of the element, respectively. Length L is the distance between the center of two subnodes. Width W is the length of a line segment passing through the mid point and perpendicular to the line that connects the centers of the two subnodes. Fig. 4 shows the examples of width and length calculation.

[Figure 4]

Fig. 4 W and L of a two-subnodes path element

For the contact resistance that connects conducting layers, the resistance is calculated by the following equation: R = Rl / n where Rl is the link resistance for an individual contact, and n is the number of contacts on the site. REX has to determine the conducting layers being joined before calculating the actual resistance of a contact site. The link resistances are different not only among contact layers, but also by the layers they connect. For instance, the link resistance for metall contact is different for metall-diffusion and metall-poly connections.

Path elements with more than two subnodes are dealt with differently. In this case, dummy subnodes are inserted between the existing subnodes so that a ladder-type resistance network is generated. The resistance calculation algorithm Page 721 is shown below, with the detailed multiple-subnodes resistance calculation scheme.

```
Calculate_resistance() {
 If element has more than two subnodes {
    For each breakline or diffusion subnode of a path ele-
ment edge do {
      If the subnode is parallel to the current flow direc-
tion of the path element,
         Create a dummy subnode perpendicular to the
current flow direction extending from the center of the
subnode to the edge of the path element.  Create a resistor
from the center of the breakline to the center of the
dummy subnode.}
    For each contact subnode do {
      Create a dummy subnode segment perpendicular to
the current flow direction, passing over the center of the
contact and extending from edge to edge.
      If the distance between the center of the subnodes is
not zero
         Create a resistor from the contact subnode to the
dummy subnode.}
    If no dummy subnode has been generated so far,
      Create one at the center of the path element.
    If the path element current flow is vertical
      Sort the dummy subnodes based on y values
    else
      Sort the dummy subnodes based on x values
    Eliminate the duplicate dummy subnodes.
    Create resistors to connect every two neighboring
dummy subnodes based on the order of the sorted dummy
subnode list.
    For the rest of the subnodes on the path element,
      Create a resistor from the subnode to the nearest
dummy subnode.}
 If element has one subnode,
      Create a dummy subnode at furthest edge from the
subnode
 If element has two subnodes,
    Simple_resistance_calculation().
}
```

[Figure 5]

Fig 5. Ladder-type resistance network from multiple subnode element

Fig. 5 shows a ladder-type resistor network generated from a multiple subnode path element. Multiple subnode elements occur mostly near the bending and junction portion of a complex polygon. The small resistances on those elements are usually optimized away by merging with the neighboring bigger resistances if series simplification is performed.

The capacitances are calculated for each path element and then distributed to all subnodes within the element weighted on the size of the resistor areas. Both capacitances due to area and fringe factors are taken into account. The following equation is used to calculate path element capacitance: C = Carea + Cfringe = area * fa + perimeter * ff where fa is the per unit area capacitance for the polygon layer, and ff is the per unit length capacitance value. These values come from the technology file.

The path element capacitance is distributed among the subnodes weighted according to their resistance value. The capacitance on a subnode is the sum of all the capacitances distributed to the subnode.

6 RC Network Optimization and Output

In order to reduce the enormous amount of data extracted from a complex chip layout, it is desirable to simplify the RC network by simple serial and parallel reductions. However, when a RC netlist is used for electromigration analysis, the element width and contact count information associated with a resistance element are to be kept intact. In this case, only series path element resistances of same width are combined.

REX generates two types of netlists: a SPICE netlist for circuit simulation purposes, and a custom format tailored to analyzers developed in-house.

A schematic representation in DLF format of the extracted RC network can also be generated for overlaying with the original layout for visual inspection of the parasitics using the layout editor.

[Figure 6]

Fig 6. Overlaying an RC network schematic to original layout

Shown in Fig. 6 is an example of the extracted RC network. The SPICE format netlist is included in Fig. 7. Page 722

[Figure 7]

Fig. 7 REX generated SPICE-format netlist

It can be seen from the netlist that the layer and coordinate information are maintained not only on the resistance and capacitance elements, but they are also used to specify subnodes. Note that M1 and M2 subnodes of resistors R_CPCH_4350_5650 and R_CPCH_4650_5650, respectively, are referred to the chip device names. They are the so-called device association subnodes. Those two devices are associated with resistors on the diffusion layer. The analyzer knows they are source/drain connections. The width and length of a sheet resistor and contact count of a contact resistor are written as comments prior to each resistor description statement.

7 Experimental Results

Table 1 REX example results

[Table 1]

The tool has been used to extract parasitics of critical nodes of various chips designed by Digital's Semiconductor Engineering Group. A test case of various special geometries was created and the RC network extracted to verify the correctness of the extraction results. There were no measurements of the subnode capacitance, but the accuracy of extracted resistance is within 10% of measured values.

Shown in Table 1 are the memory and CPU requirements to extract RC networks of several chips.

8 Conclusions

A resistance and capacitance extraction program for VLSI layouts has been presented. The extraction results are used to analyze electromigration phenomena of power/ground nodes, signal skew of clock and signal nodes, latch-up analysis, signal noise analysis, etc. The heuristic algorithm used by REX is fast enough to handle complex VLSI layout while generating reasonably accurate data.

9 Acknowledgments

The author would like to thank Neil O'Sullivan for his efforts in implementing the fracturing module, Kent Dalton for his capacitance estimation routines, Chris Hersman for some geometry processing routines, and Christina Chieng for refinement of the custom format output and maintenance of the program. The author would also like to thank Bill Grundmann for his suggestions of algorithmic improvements, Bill Wheeler for the custom format proposal, and the VIP implementation team for their feedback during the development of the program.

10 References

[Bark85] E. Barke, "Resistance Calculation from Mask Artwork Data by Finite Element Method," *Proc. 22nd DAC,* 1985.

[Bent75] J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," *Comm. Ass. Comput. Mach., Vol. 18, No. 9.* Sept., 1975.

[Chaw70] B. R. Chawla, H. K. Gummel, "A Boundary technique for Calculation of Distributed Resistance," *IEEE Trans. Electron Devices, Vol. ED-17, No. 10,* Oct. 1970.

[Hall87] J. E. Hall, D. E. Hocevar, P. Yang, M. J. McGraw, "SPIDER -- A CAD System for Modeling VLSI Metallization Patterns," *IEEE Trans. CAD, Vol. CAD-6, No. 6.* Nov. 1987.

[Horo83] M. Horowitz, R.W. Dutton, "Resistance Extraction from Mask Layout Data," *IEEE Trans. CAD, Vol. CAD-2, No. 3.* July 1983.

[Rose85] J. B. Rosenberg, "Geographical Data Structures Compared: A Study of Data Structures Supporting Region Queries," *IEEE Trans. CAD, Vol. CAD-4, No. 1.* Jan. 1985.

[Scot85] W. S. Scott, J. K. Ousterhout, "Magic's Circuit Extractor," *Proc. 22nd DAC,* 1985.

[Star87] D. Stark, M. Horowitz, "REDS: Resistance Extraction for Use in Digital Simulation," *Proc. 24th DAC,* 1987.