Printed Circuit Board Layout Verification and Circuit Extraction

Mark S. Basel
Michael B. Steer
Sasan H. Ardalan

Center for Communications and Signal Processing
Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, NC 27695-7911

SEPTEMBER 1987

CCSP-TR-87/14

Abstract

This technical report is a report on approaches to the verification and parasitic extraction for printed circuit boards. The work is aimed at determining approaches to post layout netlist extraction for printed circuit board simulation taking in to account proximity effects such as coupling of traces and EMI. The report is a product of phase II of an enhancement project entitled "Propagation of High Speed Digital Signals in Printed Circuit Board Systems" funded by Bell Northern Research through the Center for Communications and Signal Processing at North Carolina State University.

<u>Introduction to Layout Verification and Circuit Extraction</u>

It's one thing to plan something on paper and quite another to carry the plan into action in the real world. Generals and circuit designers have more in common than might be apparent. They both layout plans of action and try to preceive difficulties before they happen but as history has shown, there's always something unforseen that can bring everything to a halt. In the case of circuit designers, one of the traps is parasitic elements in the actual physical device. One can guess at these values and design around them but this leads to two potential problems. First, one might underestimate the parasitic effects and find that the resulting circuit doesn't work because of timing (RC) delays, crosstalk or a host of other problems brought on by unforseen elements. On the other hand, one might be conservative and estimate very high values for the parasitic parameters. Though this might guarantee that the circuit will work the tradeoff in performance and circuit density might result in the loss of a competative edge and thus lost sales. With increasing competition in the electronics field, no one can afford to isolate themselves from the marketplace completely.... even the circuit designer.

As circuits become increasingly complex, more and more of the physical design is done by the computer. The question then arises; did the machine layout the same circuit that I told it to ? If it did, the next question must also be answered; what is the effect of the wiring and physical structure on the circuit performance ?

These questions are at least partially answered by using another set of software-tools known generically as "circuit extractors" or "layout verification tools". This paper will look at the problem of circuit extraction and some of the methods that have arisen to provide the answers. Several topological techniques have been developed and incorporated into specific software programs. We will look at these methods in general and then their applications in specific. Finally, we will look at methods for obtaining the parasitic element values from this mask data and how these numbers are incorporated in the circuit model.

I. Topological methods used in IC and PCB artwork verification

In reviewing a mask set for verification and circuit extraction purposes, there are several errors that (hopefully) the algorithm will detect.

1. Geometrical design rule errors
   These rules are technology dependent but in general are concerned with minimum widths, minimum spacings, etc. A violation of such rules usually results in decreased yield or non-functional parts.

2. Topological errors
   Including electrical connection mistakes and improper structure of circuit elements. These are often detected as logical errors and are frequently fatal.

3. Electrical performance errors
   Failure to meet power dissipation, timing requirements etc. The former due to inadequate device dimensions and the latter to disregarding parasitic effects. The circuit often functions but fails to meet performance standards.

When performing circuit extraction and verification, it's obvious that the algorithm must be accurate and not miss design errors or rule violations. What's less obvious is that false errors are equally bad. If the algorithm repeatedly finds a "legal" shape in error, the program is probably doing more harm than good (the "boy who cried wolf" syndrom). Likewise in parasitic element extraction only critical areas should be modeled in detail since a full blown simulation of the entire circuit including parasitics becomes astronomically complex.

There are mainly two or three major groups of algorithms; (1) Shape based, (2) Edge based, and (3) Bit-map representations. The reason for the "two or three" is because of the fuzzy line between shape based and edge based algorithms that will become apparent later. In the following discussions, only two mask layers will be considered for simplicity and the total number of shapes (polygons) in those layers is N. The total number of edges needed to make these N polygons is denoted by n. We will also assume that the density of shapes across the masks is uniform and that there is no horizontal or vertical bias to the edges (i.e. the average number of horizontal or vertical edges is of $O(N^{1/2})$. The average number of unique x or y coordinates is assumed to be of $O(N^q)$ where q is less than 1 and approaching 0.5 for a uniform density, unbiased layout.

The simplest algorithm is a shape based one whereby all figures are compared by their enclosing rectangles as shown in Figure 1.1. Only shapes contained in overlaping rectangles are compared and thus the complexity of comparisons is of O(N) due to our assumption of uniform shape density. The total number of rectangle comparisons is of $O(N^2)$ and this establishes the total complexity for shape based algorithms. The following algorithms attempt to improve on this complexity figure of merit and thus provide faster execution times.

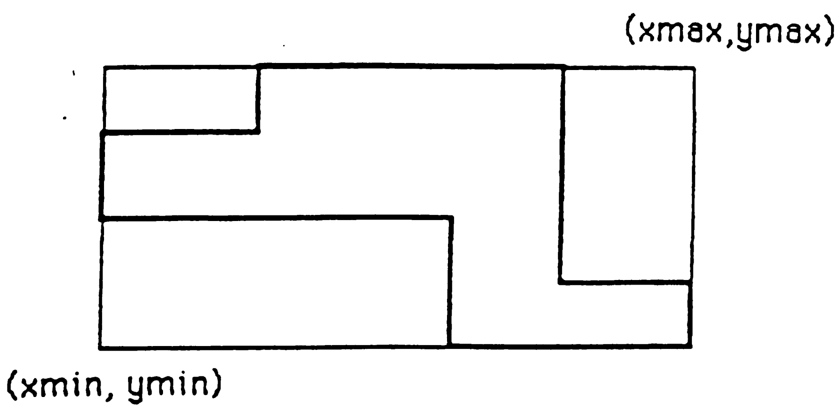


Figure 1.1

One of the first techniques used to reduce this complexity was to partition the chip into smaller areas. The shapes are first presorted by their location and then divided into K partitions. Following our assumption of uniform density, this produces and average of N/K shapes per partition. Next, an exhaustive comparison algorithm is applied to each partition resulting in an $O(N/K)^2$ number of comparisons per partition. With K partitions total, the algorithm has complexity of $O[(N/K)^2 K)]$. According to this, we can reduce the complexity to nearly zero by increasing the number of partitions forever. The catch is the shapes which fall on the boundary of a partition and must be compared against other shapes in both partitions. It can be shown that if M shapes per partition are expected to cross the boundary, then the complexity approaches $N^2/K + 2NM$. The number M is highly dependent on the vertical or horizontal bias of the design with $O(N^{1/2})$ for the unbiased case. In general, the complexity of such a partitioning algorithm is dependent on the second (2MN) term and is of $O(N^{3/2})$ regardless of the number of partitions. The tradeoff being that as the number of partitions K is increased, the number of border crossings M also increases (Wil 81).

Scan-Line Techniques

A variation on this static paritioning scheme is to use a moving partition or scan-line across the mask and consider only those boundary cases encountered after each repositioining. In this type of algorithm the mask is composed of three separate subsets of shapes at any position of the scan-line. The first is the subset of shapes that intersect the scan-line at its' current location (see Figure 1.2). This active worklist of current shapes is represented by the shaded polygons intersecting the vertical scan-line. Assuming that the scan-line is moving from left to right, the second subset is those shapes which have encountered the moving line and have been passed by. These shapes are deleted from the current worklist since they cannot intersect those shapes to the right of the line. The third subset is the group of shapes not encountered yet by the scan line but immediately adjacent to it. In this example, the scan-line has just encountered shape S and thus becomes the new "current shape". S is compared against each shape in the active worklist and thus the complexity of this step is just the size of the worklist. With our uniform density assumption, this operation is of $O(N^{1/2})$. All of the shapes (N) are entered into the worklist at one time or another and thus the total complexity is of $O(N^{3/2})$.

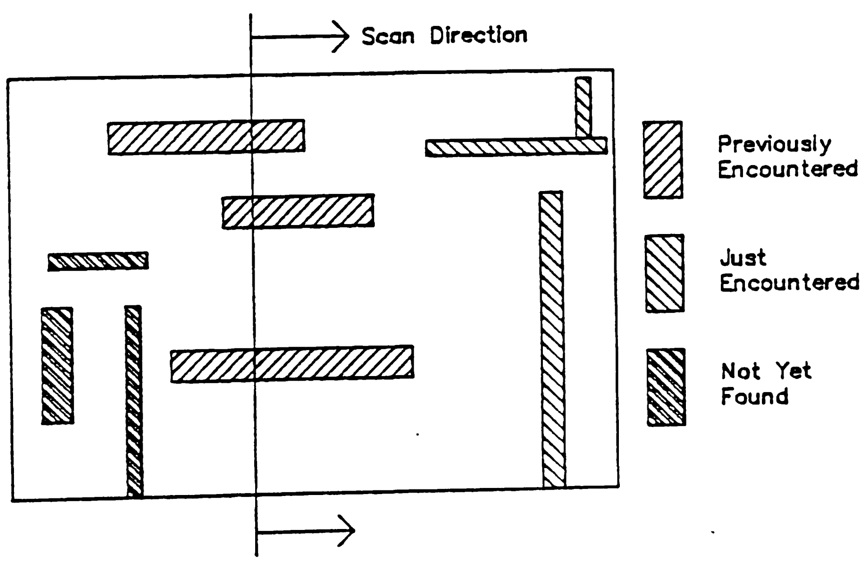


Figure 1.2

An increase in performance has been demonstrated (Mitsuhashi 80) for this method by using primitive shapes derived from the more complex polygons of the design. An example of the use of such primitives is shown in Figure 1.3. Though the theoretical complexity remains at $O(N^{3/2})$ the simplification of the detailed comparison of shapes often improves the performance speed significantly.

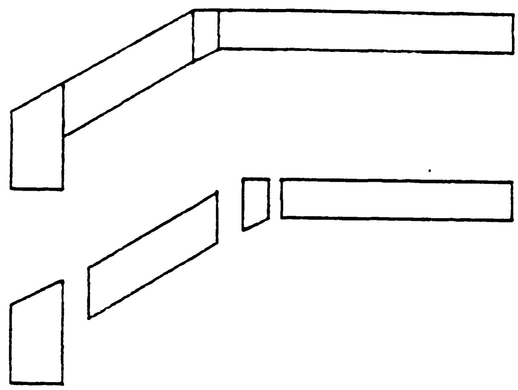
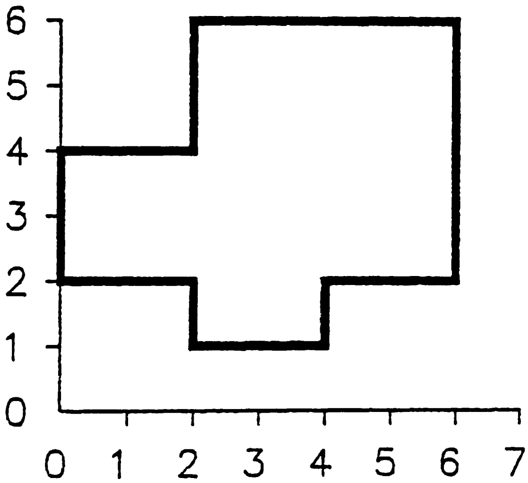


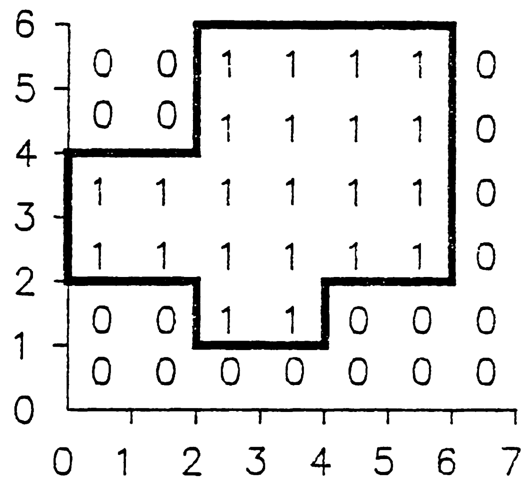
Figure 1.3

II. Essential bit-map representations

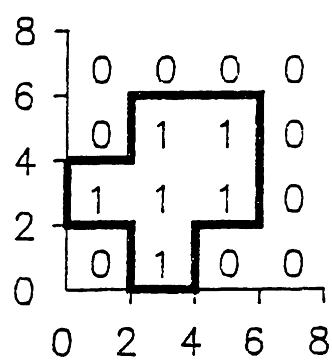
One of the simplest means (conceptually) to precisely represent mask data for Boolean operations is the use of bit maps. Here the entire chip is divided into grids cooresponding to the smallest dimension required. Opaque areas would be represented by ones and transparent areas by zeros as shown in Figure 2.1. This representation is fine for simple circuits with relatively coarse features but as might be suspected, it breaks down quickly for present day high density, small feature devices. If a grid of 1-micron spacing were used for a 1x1 cm chip, the entire 10^8 bits (1 bit = 1 grid square) would need testing for any type of Boolean operation. For practicality, this 1 to 1 grid map can be replace by an "essential coordinates" grid as shown in below. Part (a) shows the actual mask lay-out, part (b) shows the total bit map representation. To arrive at the "essential bit map" (c), draw vertical and horizontal vertical scan lines for all unique x,y coordinates. Regions outside the shapes (transparant) are denoted by 0 and the shapes (opaque) themselves by 1s. The limitation to this method is its' dependence on "manhattan" shapes, i.e. figures



(a) Polygon



(b) Simple Bit-Map



(c) Essential Bit-Map

Figure 2.1

made up of rectangular shapes parallel to either axis. With an essential bit map the total number of Boolean comparisons needed is just;

[no. of unique x values] x [no. of unique y values]

Studies of shown that the number of unique x or y values for an unbiased design is $O(N^q)$ where q has values between 0.6 & 0.8 (Wil 81). Therefore, operational complexity is of $O(N^{1.2})$ to $O(N^{1.6})$.

An improvement of this method of representation was proposed by Wilmore and is known as HIBAWL or Hierarchical Integrated Circuit Bit-map Artwork Language. HIBAWL is a combination of Essential Bit-map and Scan-line representations. For each unique y value, a horizontal scan line (HSL) is defined. The essential coordinates for a mask are referred to as "change-ys" (CHG-Ys) and the expected number of these CHG-Ys is of $O(N^q)$. For a given scan line and our assumption of a uniform density, unbiased mask, the expected number of change segments for a given CHG-Y is of $O(N^{1-q})$. Using a hierarchical database for the ordering of the data across the mask design reduces the search complexity assocaited with locating new changes and thus Boolean operations to a linear O(N) complexity. To better understand the HIBAWL approach the example shown in Figure 2.2 should be examined.

The mask is initial divided into coarse divisions and the HSL is traversed across the width of the mask and the resulting changes encountered are recorded in the level 1 file. The two bit information stored for each segment represents the following information;

10 = a change from transparent to opaque
01 = a change from opaque to transparent
00 = totally transparent
11 = totally opaque

Thus level 1 records a change from transparent to opaque somewhere between 0 and 16, totally transparent between 16 and 32, a change from transparent to opaque between 32 and 48 and finally, a change from transparent to opaque between 48 and 64. Each time a "change to" is encountered, a link to the next level is established and the process repeated but this time the spacing between divisions is reduced... in other words, a finer grid map is applied. Thus the change encountered between 0 and 16 is mapped to level 2 which divides the 0-16 interval into 4 finer sections. This new level shows that 0-4,4-8, and 8-12 are all totally transparent and from 12-16 the mask is totally opaque. Since level 1 has already indicated that the region from 16-32 is totally opaque, no further reduction in scale need be performed and the region from 0-16 is completely described in the two levels. This proceedure is repeated for each "change to" (i.e 01 or 10) with each pointing to the next level in the hierarchy until, if necessary, the finest dimensions of the technology limits are used. In other words, until each bit pair represents one grid square of the bit map.

Though this method can reduce Boolean operations to an O(N) complexity, the conversion of the mask data to this format is of O(N log N) complexity for most structures. This need be done only once and should be outweighed by the advantages gained.

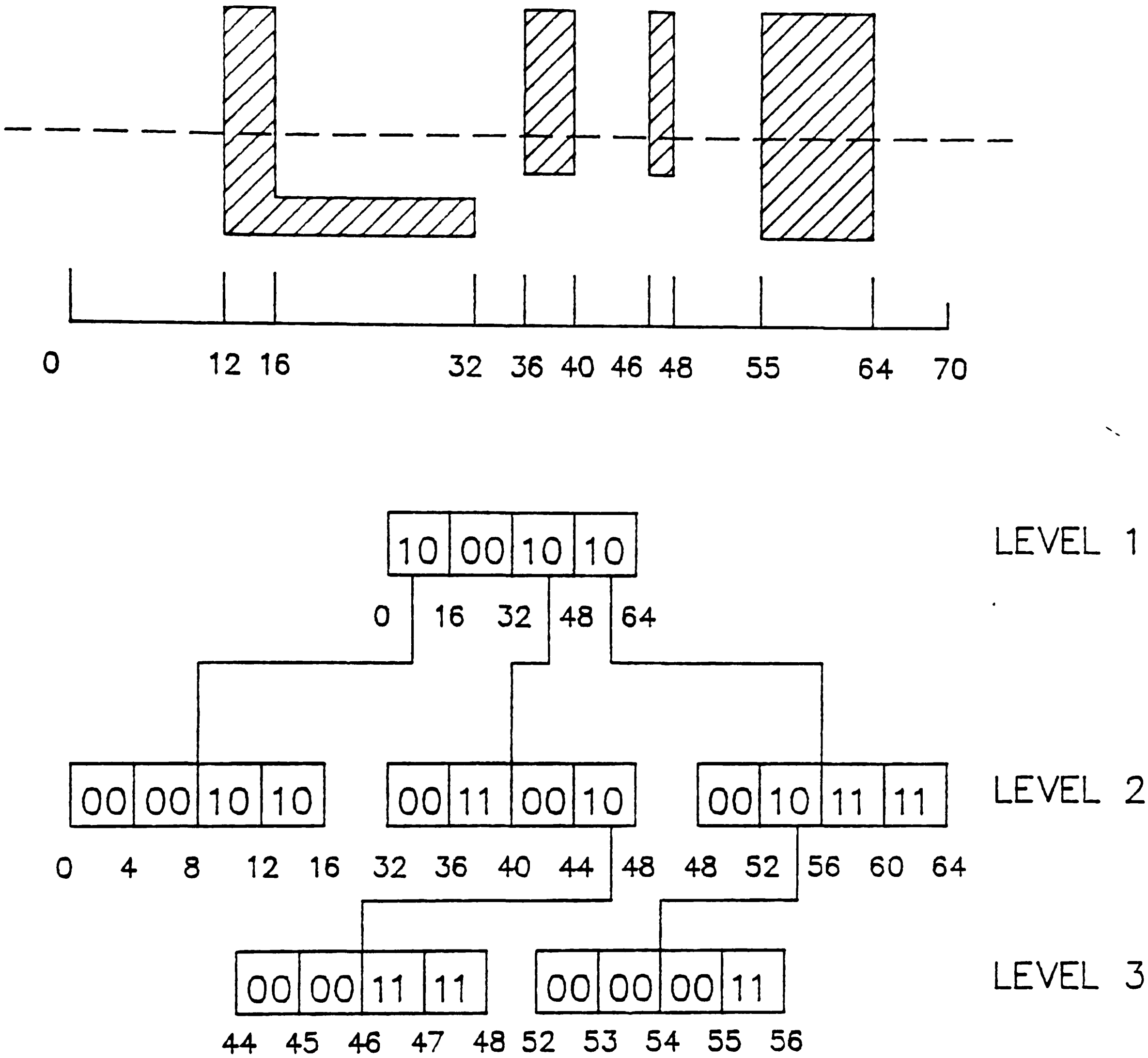


Figure 2.2 (Yos 86)

III. Corner-stitching as a circuit extraction technique

Corner Stitching is more of a database orginization than just a circuit extractor and is significantly different from other methods as to require a more detailed description. Corner stitching is a form of the neighbor pointer based data structure except that "all" tiles forming the design are accounted for. This means space tiles as well as solid tiles are provided with "corner stitches" allowing the system to locate empty spaces as well as solid shapes. Corner stitching is strictly limited to manhattan shapes, but in most design environments and technologies this is not very limiting. The space tiles are organized as "maximal horizontal strips" which means that no space tile has other space tiles immediately to its right or left. This simple requirement has the very important effect that there is one and only one decomposition of space for each arrangement of solid tiles.

All tiles (including space tiles) are linked by a set of pointers at their corners, two at the lower left corner and two at the upper right corner as shown in Figure 3.1. As corner stitching has been used almost exclusively by the team that created MAGIC, the details of some of the various operations performed with corner stitching will be left for the section of this paper that discusses the various circuit extraction programs around. Briefly, the use of corner stitching has several nice features, such as the ability to "plow" a a relatively large piece of the design thru other pieces and have those shapes move out of the way and make room. Much like plowing a field where the soil is moved out of the way of the plow but isn't "destroyed" or removed.

Because of the hierarchy and the data structure of a corner stitching database, the entire design need not be re-extracted whenever a particular cell is changed. The parents of a subcell will have to be re-extracted and the parents of those cells, etc. until the root is reached. This is a very nice feature of corner stitching, allowing incremental changes or "tweeking" to the design without major CPU costs for each re-extraction.

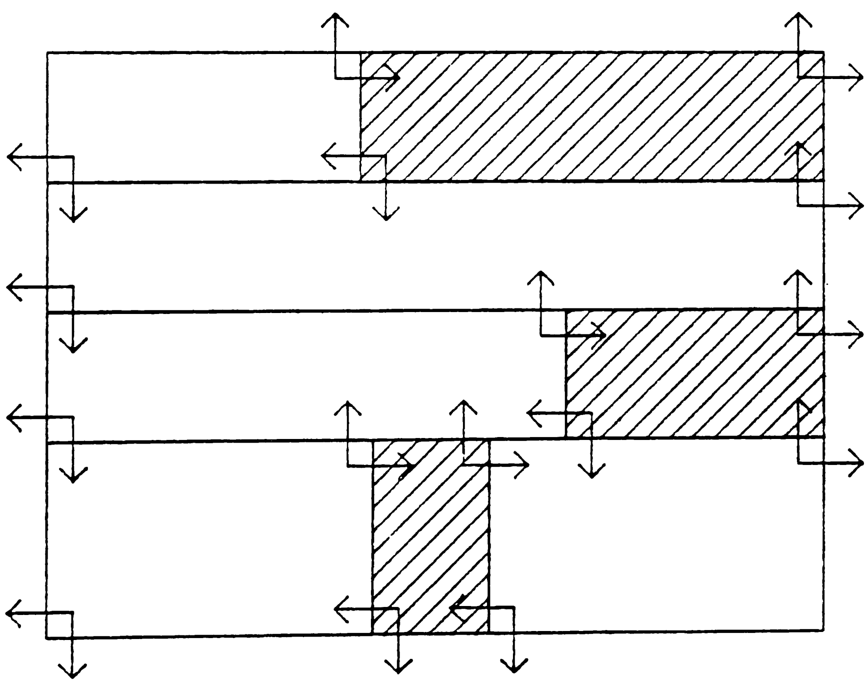


Figure 3.1

IV. Parasitic circuit element extraction

This section deals primarily with general methods of determining parasitic resistance and capacitance directly from mask layout information. Many of the circuit extraction/verification systems in use today are equipt to use several different techniques for determining parasitics. The more general of these methods will be discussed here and only briefly mentioned in the sections dealing with specific systems. On the other hand, techniques that are unique to a given system will be discussed in the appropriate section and will be given minimial attention here.

Resistance extraction

Interconnection resistance has come under scrutiny of late because of its' affect on the performance of high speed digital devices. High interconnect resistance usually means long RC time constants and thus long rise and fall times. Several methods are being used at the current time to calculate or predict this resistance and the more important ones will be discussed here. As might be expected, there is a tradeoff between accuracy and speed for all of the methods and a given algorithm should use a combination of them as an accuracy to performance tradeoff.

Laplace's Equation

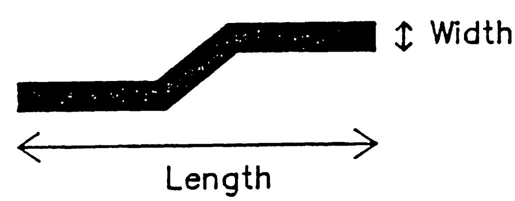
The most basic technique for calculating the resistance of any transmission line system is to use Laplace's equation and solve for div grad(V) = 0 over the resistive region.

A very complete treatment of this technique is found in (Bar 85) where the solution of Laplace's equation is combined with finite elements method to provide a 2D solution to finding the resistance of a general polygon. Though very complete, this particular solution is still somewhat computationally intensive and is probably only necessary for special applications such as encountered in analog IC designs.

Sheet resistivity

For long straight resistive regions common to many IC designs, the resistance can be calculated from:

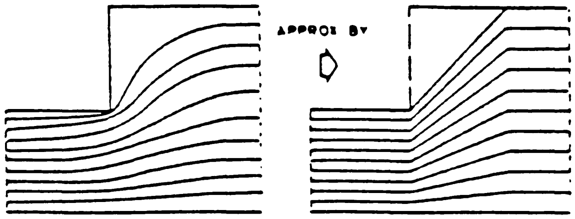
$$R = \rho_{sh} \frac{length}{width}$$



where ρ_{sh} is the sheet resisitivity for the conductor.

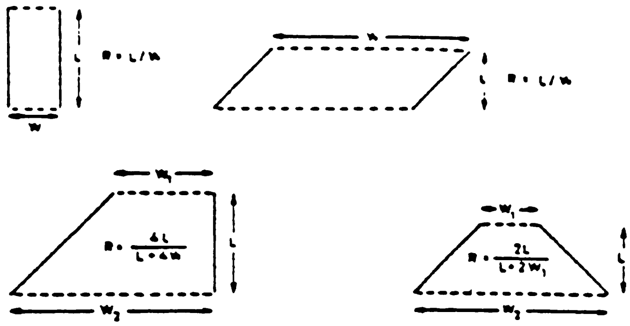
Table Look-up

This method provides for very accurate, fast "computation" of nodal resistances by spotting common shapes and looking up the correct resistance model in a library (McC 85). EXCL uses this method in combination with others and an example of the kinds of shapes catagorized is show in Figure 4.1. Note that these values are technology specific and were the result of calculation and actual measurements.



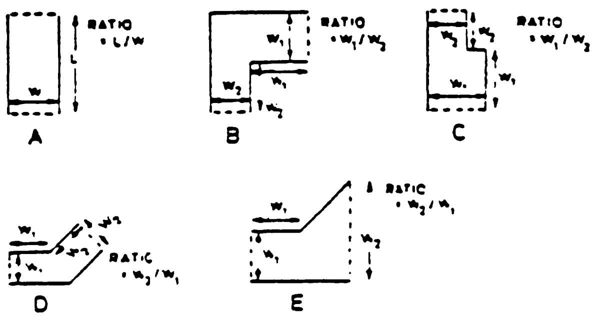
Current spreading from a small contact.

Figure 4.2



Region shapes and their resistances.

Figure 4.3



(a)

Shape	Ratio	Exact Resistance	Extracted Resistance	% Error
A	1	1	1	0
A	5	5	5	0
B	1	2.5	2.5	0
B	1.5	2.55	2.45	4
B	2	2.6	2.55	2
B	3	2.75	2.83	3
C	1.5	2.1	2.11	0
C	2	2.25	2.3	2
C	3	2.5	2.66	6
C	4	2.65	2.97	11
D	1	2.2	2.25	2
D	1.5	2.3	2.28	1
D	2	2.3	2.43	5
D	3	2.6	2.74	5
E	1.5	1.45	1.44	1
E	2	1.8	1.8	0
E	3	2.3	2.33	1
E	4	2.65	2.71	2

(b)

Test shapes and results.

Table T4.1
(Horowitz 83, et al)

In a variation on the general technique of solving Laplace's equation and the specific one of table look-up, M. Horowitz and R. Dutton of Stanford University have developed a compromise between the two. By detailed analysis of the actual current flow through various conductor geometries, a heuristic algorithm is used to divide complex polygons into simplier regions and then finds the resistance through each piece. The key difference between this method and that of table look-up is that the algorithm splits polygons into smaller regions such that normal current flow is not disturbed.

In the case of conductor changing widths from narrow to wide, the current is assumed to expand exponentially to fill the wider region. The algorithm linearizes this expantion and assumes that the current spreads out along a 45 degree angle to the direction of current flow (see Figure 4.2). Using these approximations, the resistance of various shapes as functions of their dimensions were found as seen in Figure 4.3. Table T4.1 shows the exact resistance as computed by solutions to Laplace's equation using relaxation methods as compared to those extracted by the algorithm.

This particular technique utilizes some basic facts from electromagnetics to provide a better solution to resistance extraction than using strict sheet resistivity approximations. The algorithm is an improvement on simple table look-up methods since general shapes with variable dimensions are used instead of shapes with specific dimensions. Thus instead of having several different entres in a table when the only difference is a matter of scale, one general shape of arbitrary dimensions is sufficient for this algorithm.

Capacitance Calculations

The effect of nodal and internodal capacitance is very important to the function and performance of modern high density circuits... it is also one of the hardest to extract or model. The resistance of interconnect lines is relatively straight forward since the currents are very much limited to the conducting tracks and the fringing effects of the E-fields can be ignored. For traditional cases of small scale integrated circuits or circuit boards, the parallel plate approximation for nodal capacitance was useful because of its' simplicity and ball-park accuracy. In todays VLSI circuits, most of the conducting paths lie in roughly the same plane and thus parallel plate approximations become very inaccurate and of little use. In response to this need for better accuracy for complex geometries, circuit extraction designers have turned to electromagnetics and microwave circuit solutions. As in resistance calculations, the tradeoff between accuracy and speed is still here and various extractors use combinations of methods to cope with these two conflicting requirements. The various methods range from integral-equation numerical solutions to three dimensional conductor problems (Rue 73), solutions to Gauss's law for various geometries (Smi 84), finite element solutions of Laplace's equation (Tay 84) and statistical modeling of parasitics (Michael 84).

One of the more interesting attempts at simplifying the magnitude of the required calculations is the use of statistical models for various interconnect geometries and shapes (Mic 84). This method uses conventional two-dimensional capacitance simulators to determine fringe and coupling capacitances under a variety of conditions and using these results along with statistical regression techniques to generate cooresponding capacitance equations.

For these initial simulations, the capaciance structure shown in Figure 4.4 were used where W, T and S are the width, thickness and separation respectively of the conductors and H is the height above the substrate. C_{pp} is the well worn parallel plate capacitance given by:

$$C_{pp} = \epsilon_0 \epsilon_r \frac{W}{H}$$

The conductor to ground capacitance is given by :

$$C_0 = C_{pp} + 2C_{f0}$$

$$C_1 = C_{pp} + C_{f0} + C_{f1}$$

$$C_2 = C_{pp} + 2C_{f1}$$

for the case of zero, one or two adjacent conductors respectively. The total capacitance for one or two adjacent conductors is:

$$C_{tot1} = C_{pp} + C_{f0} + C_{f1} + C_c$$

$$C_{tot2} = C_{pp} + 2C_{f1} + 2C_c$$

Numerous simulations were run for varying widths, separation distances and thicknesses and the results used as the data base for a regression analysis. Models were found for C_{f0}, C_{f1} and C_c using multiple linear regression. The predictor variables included the height, width, thickness and separation of the conductors.

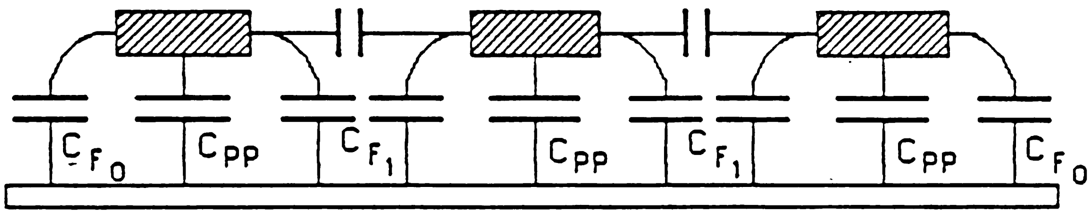


Figure 4.4

Details of the method and the reasoning behind it can be found in (Michael 84) but the results for several values of W, T, H and S can be found in Table T4.2. The final equations for C_{f0}, C_{f1} and C_c (actually the natural log of these variables) are shown below.

$$\begin{aligned} Ln(C_{f0}) = {} & 3.805 - 0.299Ln(H) + 0.0257(W) + 0.0524(T) - 0.00101W^2 \\ & - 0.0469(T \times Ln(H)) - 0.0181(Ln(H))^2 \end{aligned}$$

$$\begin{aligned} Ln(C_{f1}) = {} & 3.952 - 2.20(1/S) - 0.394Ln(H) - 0.533(Ln(H)/S) + 0.932(1/S^2) \\ & + 0.030W - 0.0187(W \, x \, Ln(H)/S) + 0.0846T - 0.00125W^2 \\ & - 0.00907(W/S) - 0.0776(T/S) \end{aligned}$$

$$\begin{aligned} Ln(Cc) = {} & 4.343 - 0.651Ln(S) + 0.193Ln(H) + 0.487Ln(T) - 0.879(1/W) \\ & - 0.212(Ln(S))^2 - 0.167(Ln(T)xLn(H)xLn(S)) \\ & + 0.104(Ln(H)xLn(S)/W) - 0.144(Ln(S)/W) + 0.0619(Ln(H)xLn(S)) \\ & + 0.470(1/W^2) - 0.144(Ln(T)xLn(S)/W) + 0.232(Ln(T)/W) \\ & + 0.111(Ln(T))^2 + 0.470(Ln(T) \, x \, Ln(H) \, x \, Ln(S)/W) \end{aligned}$$

Though lengthy, these equations provide a sort of empirical, closed form solution to some of the more common capacitance problems associated with complex circuits. As with any regression model, one should never try to extrapolate these results beyond the limits used for W, T, H and S in the initial 2D simulations.

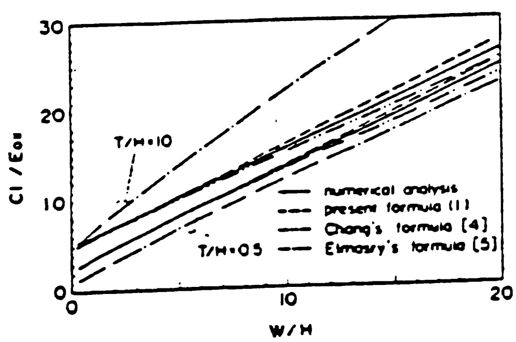
Another simple, closed form procedure for calculating two and three dimensional capacitances of interconnect lines has been proposed by T. Sakurai and K. Tamaru of Toshiba Corporation (Sak 83). The procedure is similar to the statistical modeling used by Michael in that detailed methods were used to calculate the "exact" capacitance by finite element methods and the correct capacitance was assumed to have been found when a doubling of the number of sub-areas produced less than a 1% change in the computed capacitance. From these calculations, the following empirical formulas were obtained for the cases of one, two and three conductors.

$$C_1 = \epsilon_{ox}[1.15(W/H) + 2.80(T/H)^{0.222}]$$

$$C_2 = C_1 + \epsilon_{ox}[.03(W/H) + .83(T/H) - .7(T/H)^{0.222}](S/H)^{-1.34}$$

$$C_3 = C_1 + 2\epsilon_{ox}[.03(W/H) + .83(T/H) - .7(T/H)^{0.222}](S/H)^{-1.34}$$

The wiring capacitance by various methods, including Sakurai's are shown in in the graph below (Sak 82).



Calculated wiring capacitance by various formulas.

Lumped Element Modeling of Signal Lines

The question of how much is enough for modeling signal lines on physical circuits is an important one. Obviously, the more elements used in a T or Ladder network the better the accuracy. This accuarcy comes at the price of increased CPU time if the circuit is to be run on a conventional SPICE type simulator. Antinone and Brown (Ant 83) used conventional transmission theory from electromagnetics to arrive at time and frequency domain performance of a line. They then used several modeling techniques including T networks, and 2, 5 and 10 element ladder networks see Figure 4.5. They specifically looked at a line 3000 microns long with an oxide thickness of 0.6 microns and a polysilicon thickness of 0.6 microns. The polysilicon is the conductor in this case and had a width of 6 microns and a sheet resistivity of 20 ohms per square.

Using the detailed modeling of Ruehli and Brennan they found that the line resistance was 10k ohms and had an effective line capacitance of 1.1 pF, including fringing capacitance. The results of these various simulation models are shown in Figure 4.6 where the magnitude and phase angle of the various structures is depicted. The main result of this study is that relatively few elements need be extracted for use in modeling parasitic effects on integrated circuit interconnect lines.

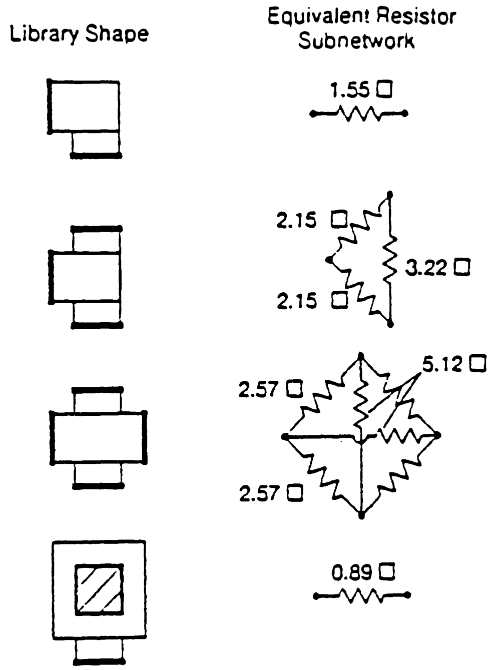


Figure 4.1

Conditions W	T	H	S	Regression C_C	Regression C_{F1}	Regression C_{TOT}	2-D Simulator C_C	2-D Simulator C_{F1}	2-D Simulator C_{TOT}
1.0	0.8	0.8	1.0	40.56	18.41	161.12	44.24	18.54	168.75
1.0	0.8	0.8	9.0	2.51	49.82	147.85	2.33	47.69	143.24
5.0	1.6	1.6	1.0	93.05	10.78	315.63	91.64	10.30	311.87
5.0	1.6	1.6	5.0	17.67	34.56	212.45	18.62	32.81	210.84
5.0	1.6	1.6	9.0	7.33	42.35	207.33	7.48	45.35	213.65
13.0	1.6	1.6	13.0	4.44	48.80	387.19	3.44	49.33	386.29

Conditions W	T	H	Regression C_{F0}	Regression C_{TOT}	2-D Simulator C_{F0}	2-D Simulator C_{TOT}
1.0	0.8	0.8	51.30	145.79	51.70	146.59
1.0	1.6	0.8	53.95	151.08	55.01	153.21
1.0	1.6	1.6	40.35	102.28	43.42	108.44
5.0	1.6	1.6	43.64	195.24	43.93	195.84
13.0	1.6	1.6	46.35	373.38	47.28	375.31

Table T4.2

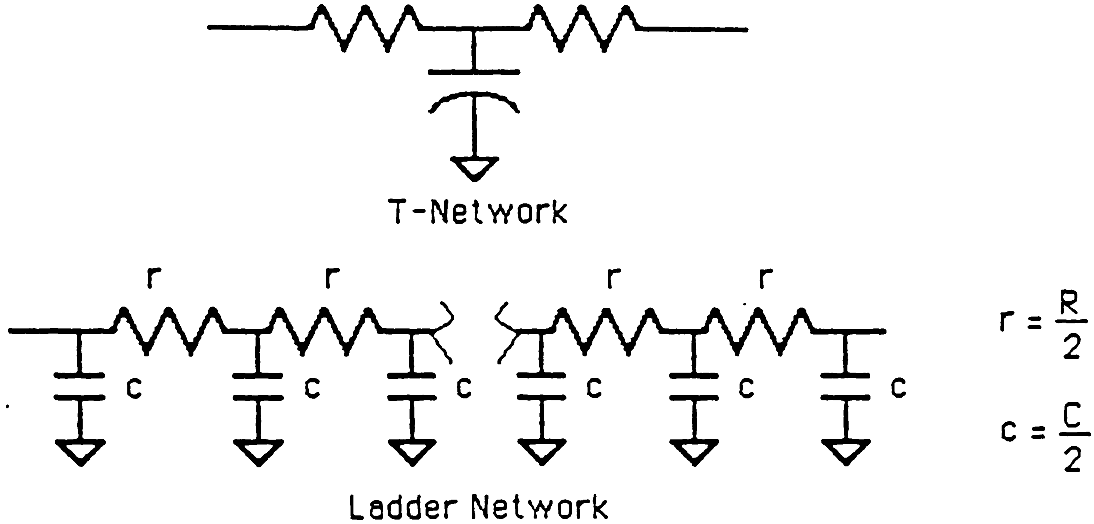
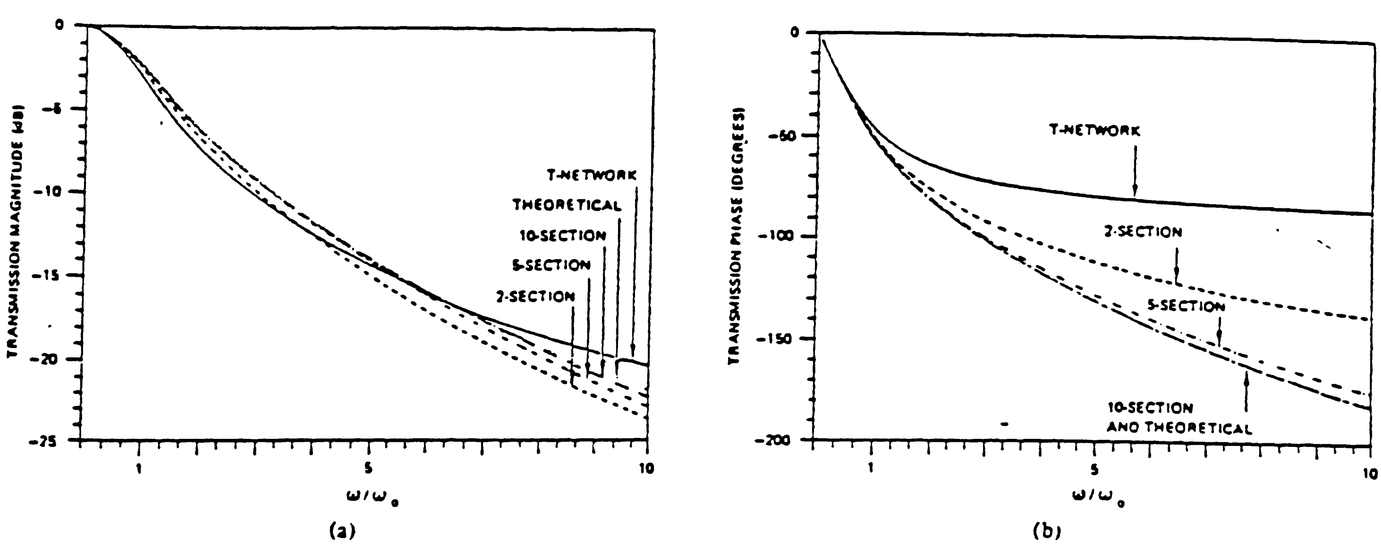


Figure 4.5



Simulated polysilicon line transmission characteristics in the frequency domain compared to theoretical results. (a) Magnitude. (b) Phase.

Figure 4.6

V. The MAGIC system

The MAGIC VLSI design and layout system incoporates a hierarchical circuit extraction system based on a "corner stitching" structured data base. Due to its' hierarchical nature, each cell in a design is given it's own file containing ;the circuit of that cell. This file contains the circuit contained in the cell, parasitics associated with it and any interconnects to other cells. The algorithm allows for almost arbitrary overlap between cells and parasitic capcitance is adjusted when overlaps occur. Since each cell is extracted independent of context, only the cell and its' ancestors need be reextracted.

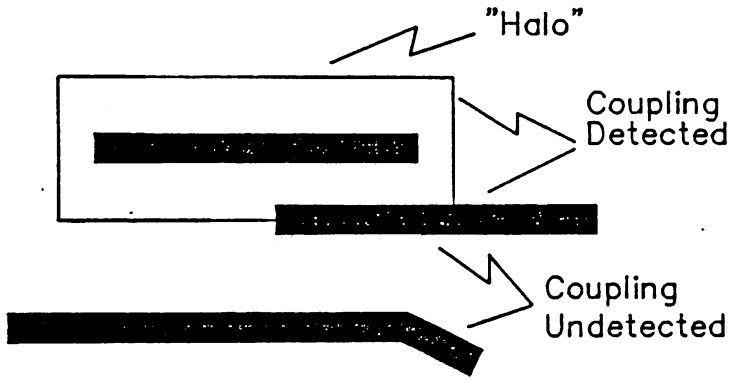
Parasitic element extraction under MAGIC

The interconnect extracted circuit nodes are modeled as shown in Figure 5.1 with R being the interconnect resistance and C the node to ground (substrate) capacitance. The internode coupling capacitance (not shown) is computed for cases when wiring on one mask layer overlaps wiring on a different layer and when parallel wires run near each other on the same layer. Since capacitance due to overlap can only occur between tiles on different layers, the capacitance calculation algorithm is fairly straight foreward as shown below.

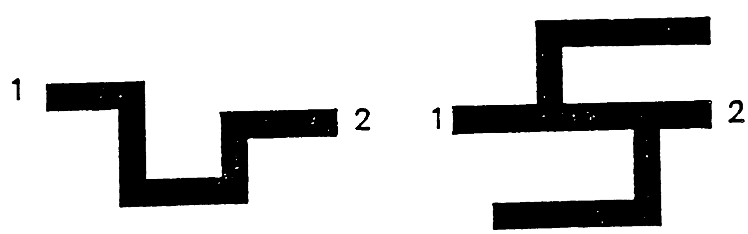
1. Search other planes for tiles that overlap tile t.

2. Compare the nodes of each tile found with t's node.
   If they are different, record a coupling capacitance
   between the two nodes. This capacitance depends on
   the tile type and the areas of overlap.

Since coupling capacitance due to parallel lines is inversely proportional to the separation between them, MAGIC ignores any parallel edges beyond a few lambda outside the edge in question. This distance is specified by the user and is usually technology dependent.



Interconnect resistance is computed somewhat simplistically in MAGIC, though this does allow faster program execution. The resistance extractor uses the perimeter and area of the interconnect region and solves a quadratic equation to find the effective resistance. This is a relatively good estimate for interconnects such as in (a) below but is poor for ones such as (b).



(a) (b)

Since MAGIC's extraction algorithm extracts cells independent of the way in which they are used by their parents, changing a parent cell doesn't require that its' childern cells be re-extracted. In order to guarantee that this method works, restrictions are required as to the way in which cells are allowed to overlap. Arbitrary cell overlap could cause a transistor to be split between cells or formed by accidental overlap of several cells. Instead of prohibiting cell overlaps as in some extractors, MAGIC allows cells to overlap or abute as long as this only connects portions of cells without changing their transistor structure (Sco 85).

VI. The SPHINX system

The SPHINX artwork editing and verification system was developed by the Hewlett Packard Corporation as a general purpose design and editing tool. The database is set up in a hierarchical structure that allows much of the stored data to be used efficiently by various tools without having to convert shape data, for example, to electrical net data.

The SPHINX system uses strictly manhatten shapes; horizontal or vertical rectangles. These rectangular shapes are stored in the database as shapes on mask layers and are referred to as horizontal edge pairs or H-pairs. By representing all shapes (including holes) as rectangles, the need for special hole figures and directed boundaries is avoided. There is detailed information available as to the actual database design for SPHINX (Wil 84) and should be referred to as needed. The main point is that SPHINX is a scan line based, hierarchical system capable of generating mask layers and performing circuit element extraction from the same databases.

Not a lot of information is written about SPHINX's resistance calculation process, but in general these are performed by either sheet resistance approximations or using shape look up tables.

Capacitance, on the other hand has been given more attention and provides for a pseudo-3D approximation to interlayer capacitance (Mor 84). Two forms of vertically related capacitance are modeled in SPHINX including overlapping areas and edge to area overlap. Thus for vertical capacitances, the nodal capacitance due to these two forms is given by:

$$\text{Capacitance} = C1 \times [overlapping\ area] + C2 \times [overlappingedge\ length]$$

where C1 is a capacitance per unit area and C2 is the capacitance per unit length. The edge part of this calculation is also used to determine the lateral capacitance for a diffusion junction (static case). Since this calculation does not handle non-overlapping mask patterns an additional capacitance term is used when the parallel edge length becomes significant. This additional term is:

$$\text{Capacitance} = C3 \times (facing\ edge\ length) \times e^{(-distance/D)}$$

where C3 is a capacitance per unit length and D is an effective interaction distance.

This simple handling of same layer coupling capacitance doesn't really follow the present trends in circuit extraction systems. As line widths decrease, their thickness is often increased to lower interconnect resistance. Thus the lines not only become more compact and in closer proximity to each other, their edge face areas are increasing. As the cross section of these lines becomes almost square, even the use of the simple parallel plate model becomes as important for lines in the same layer as for lines overlapping on different layers. Other extractors place more emphasis on this "same layer" problem that SPHINX unforunately neglects.

VII. The NEXT layout verification and circuit extraction system

The NEXT system, developed by researchers at Rensselaer Polytechnic Institute and General Electric, is a hierarchical verification and circuit extraction system. NEXT utilizes the fact that most complex VLSI designs are partitioned into smaller more manageable sub-systems and that the physical design usually follows this hierarchy directly.

In verification mode, NEXT extracts a netlist from the hierarchical layout and compares this to the reference list to uncover any inconsistencies between the two representations. After the layout is verified, the interconnect parasitic elements are generated and layout features are modeled. Various degrees of modeling are available from fairly detailed RC networks to capacitance only models, depending on the criticality of a particular portion of the circuit. A flowchart of proceedures is shown on the next page indicating required input data and generated output (extracted features, elements, etc.).

The feature recognition algorithms are based on a graphical abstraction of the layout information. Various relations between shapes, such as intersection, containment and proximity are used in "feature graphs" that are used to describe the physical layout of actual devices. All of the feature graphs required to describe the layout features is collectively called the "procss graph". The extracted features are used to generate a "layout graph" wherein devices and interconnect regions are determined. This information is then used by the "topological modeling" section and the specific circuit element information is extracted. Included in this is parasitic elements such as interconnect resistance and capacitance. Varying degress of modelling sophistication are available, depending on the criticality of the circuits involved.

For the most part, the parasitic element calculations used by NEXT are relatively primitive as far as transmission line modeling is concerned. Resistance calculations use sheet resistance models for the most part, though fracturing the interconnect into primitive polygons is possible allowing table look-up for more accurate resistance models. Capacitance calculations use impirical formulas for the most part that are functions of line width, depth, length and separation. Finite element methods are also available based on the work of Ruehli and Brennan (Bre 73) but should probably be avoided due to the large computation times involved. Detailed calculations become illrelevant in the long run since interconnect distributed transmission lines are modeled as simple lumped element π networks in the end anyway.

VIII. The EXCL system

EXCL is strictly a circuit extraction program and is designed to be run against the mask artwork data produced by some other means. Since EXCL works on manhattan shapes only, a preprocessor converts non-orthogonal shapes into stepped orthogonal ones (McC 85).

The program has general algorithms to find interconnection resistance, internodal capacitance, ground capacitance and transistor sizes. The extractor also incorporates other techniques for determining parasitic element values and whenever possible it uses these rather than the general algorithms to speed up operations. The software maintains a shape library containing the resistances for various frequently encountered geometries and uses these precalculated values rather than calculate them each time they're found.

Resistance extraction under EXCL

EXCL uses three different techniques to calculate interconnect resistance ranging from a very general technique that solves Laplace's equation to a very specific but fast technique using table look-ups. The first method, as mentioned, involves solving Laplace's equation over the resistive region and follows much the same pattern as discussed in the previous sections on general techniques.

The second method has also been discussed in the general techniques section and involves using sheet resistance calculations. This method is applied only to long, straigh interconnect regions of the design though EXCL does compensate somewhat for junction off of the straight section by removing "current spreading region" from each subregion.

The third technique is somewhat unique to EXCL though there certainly isn't any reason why this technique cannot be applied to other circuit extractors. This last techniuqe is simply a table look-up whereby commonly occuring shapes are contained in a "reistance shape library" and the equivalent nodal resistance is found and place into the extracted circuit based on the geometry of the found interconnect region. Several examples of this library are contained in Figure 8.1 along with the equivalent resistance model.

Capacitance extraction

EXCL extracts capacitance for two different cases; that of node to ground capacitance and that due to internodal coupling. The ground capacitance is arrived at in a straight forword manner by using the area and perimeter of the interconnection regions. The internodal capacitance is found by using three different techniques, depending on the situation. The program only extract coupling capacitance when the extimated inter-nodal capacitance, $C_c > \gamma C_{ground}$ where γ is an "accuracy" factor typically around 0.05 to 0.3. EXCL uses "windows" to view the area surrounding the interconnect region under examination. Much like MAGIC's "halo", EXCL ignores the effects of conductors that are outside of the window and assumes that their contribution is negligable. The intra-nodal capacitance determination is broken down into the following cases:

1. Coupling due to overlapping condutors.
2. Coupling due to two parallel conductors on the same or different layers.
3. General coupling solutions by solving Laplace's equation

The first technique uses the parallel plate approximation with the area being that of the overlap region between the two conductors. An additional "fringe correction" factor is applied to the perimeter of the overlapping region to correct for fringing fields effects. This correction factor is determined by computer simulations of the several geometetries and/or by actual experimental measurement. This fringe factor is therefore technology dependent and must be changed if the extractor is to be used on different device families.

The second technique works in similiar fashion to the first whereby a line capacitance constant is defined with dimensions of Farads/length along with an "end correction" factor akin to the fringing field correction factor used previously. The line capacitance constant is a function of the conductor width, spacing and layer combination. For most integrated circuit dimensions, the width facotr is ignored since it has small effect on this constant. For each layer combination, EXCL stores the line capacitance constant in a table and the required constant is pulled from the table depending on the situation.

The final, general technique is very similiar to that discussed in the general techniques section and need not be discussed here other than to mention that EXCL has this capability if required.

Bibliography

(Smi 84) William Smith, J. McDonald, C. Chang, R. Jerdonek, "NEXT: A Hierarchical Layout Verification System for VLSI", ICCAD-84, 1984.

(Wil 84) J. Wilmore, "The Design of the Databasea for the Sphinx IC Artwork Editing and Verification System",Digest of Technical Papers, ICCAD-84, 1984, p. 111-113.

(Wil 81) J. Wilmore, "Efficient Boolean Operations on IC Masks", 18th Design Automation Conference, 1981, p. 571-579.

(Mor 84) S. Mori, I. Suwa, J. Wilmore, "Hierarchical Capacitance Extraction of an IC Artwork Verification System", Digest of Technical Papers ICCAD-84, 1984, p. 266-268.

(Sco 85) W. Scott, J. Ousterhout, "MAGIC's Circuit Extractor", 22nd Design Automation Conference, 1985, p. 286-292.

(Ous 84) J. Ousterhout, "Corner Stitching: A Data Structuring Technique for VLSI Layout Tools", IEEE Transactions on Computer-Aided Design, Vol CAD-3, No. 3, Jan 1984, p. 87-100.

(Sak 83) T. Sakurai, K. Tamaru, "Simple Formulas for Two and Three Dimensional Capacitance", IEEE Transactions on Electron Devices, Vol. ED-30, No. 2, Feb. 1983.

(Ant 83) R. Antinone, G. Brown, " The Modelling of Resistive Interconnects for Integrated Circuits", IEEE Journal of Solid-State Circuits, Vol. SC-18, No. 2, Apr 1983.

(Tar 83) G. Tarolli, W. Herman, "Hierarchical Circuit Extraction with Detailed Parasitic Capacitance", Proc. 20th DA Conf., pp. 337-345, June 1983.

(Bas 83) J. Bastian, "Symbolic Parasitic Extractor for Circuit Simulation (SPECS)", Proc. of the 20th Design Automation Conference, 1983, p. 346.

(Mic 84) M. Michael, S. Weisbrod, "Reduction in Parasitic Capacance Computation Time Through Statistical Modeling", VLSI Multilevel Interconnect Conference, June 1984, pp. 237-243.

(Str 84) F. Straker, S. Selberherr, "Computation of Wire and Junction Capacitance in VLSI Structures", VLSI Multilevel Interconnect Conference, June 1984, pp. 209-217.

(Hor 83) M. Horowitz, R. Dutton, "Resistance Extraction from Mask Layout Data", IEEE Transactions on CAD, vol. CAD-2, No.3, 1983, pp. 145-150.

(McC 85) S. McCormick,"EXCL: A Circuit Extractor for IC Designs", Proceedings of the 21st Design Automation Conference, 1984, pp. 616-623.

Conclusion

Looking through the bibliography of this paper, one notices that many of the papers come from the integrated circuit world.... particularly the VLSI area. Both printed circuit boards and VLSI circuits have many of the same problems with transmission lines, including; crosstalk, loading, discontinuities, etc. Actually PCB have more of a problem since their interconnect lengths are much longer and their are generally more discontinuities due to vias and bends. On the other hand the dielectric thickness is also much larger which reduces the self capacitance of the lines.

The concepts developed for VLSI circuit extraction and parasitic element calculations should be valid for PCB use as well. One of the concepts that will most certainly be needed is the use of a "halo" similar to that used by MAGIC.... to calculate the effective crosstalk of each transmission line relative to every other transmission line would become ridiculously expensive computationally speaking. The main purpose of this paper has been, therefore, to keep from reinventing the wheel and to use those methodologies and ideas from VLSI circuit extraction that prove useful for PCB design analysis.