CAD at the Design-Manufacturing Interface

H. T. Heineken[*], J. Khare[*], W. Maly, P. K. Nag, C. Ouyang and W. A. Pleskacz[**]
ECE Dept., Carnegie Mellon University, Pittsburgh, PA 15213.

Abstract: *Owing to rapid changes of IC technologies, traditional design rule checking is becoming inadequate to assure satisfactory levels of IC manufacturability. This paper describes a new computer supported design analysis environment that improves the efficiency of manufacturability assessment of new products. This environment, called MAPEX 2, is described in the paper along with some of its key procedures and algorithms. Illustrations of MAPEX 2 applications and performance figures are provided as well.*

1.0 Introduction

With the continuous and rapid increase in complexity of VLSI designs and fabrication technologies, new problems are encountered at the design/manufacturing interface [1]. The essence of these problems is in the mismatches between designs and the manufacturing process, leading to substantial degradation of IC manufacturability [2]. Such degradation, resulting mainly from inadequate yield, must be addressed by introduction of a new design-manufacturing paradigm of information exchange [3]. This paradigm involves two components: *(a)* Manufacturability assessment - which evaluates the manufacturability of finished designs before they are sent to the foundry, and *(b)* Yield analysis - which produces a ranked list of both product-related and process-related yield loss mechanisms for newly fabricated products.

In this paper we focus on manufacturability assessment environment. In particular, the tools that constitute such an environment are described in detail, along with a set of algorithms used to extract yield relevant parameters.

2.0 Design Attributes Extraction Environment

Any extraction environment assessing the design/manufacturing interface must extract yield relevant IC design parameters. Since the yield relevance of some parameters is not always obvious, and in some cases unexpected [3], the number of extracted circuit attributes must be extensive.

Fig. 1 shows a block diagram of the necessary components of a complete design attributes extraction environment. As inputs the environment takes a layout and/or netlist of a design. In addition it records process characteristics (defect densities etc.) as well as relevant design environment information.

The key extraction routines are:

- Layout Reader - which scans in data from a mask database.
- Layout Connectivity Extractor - which identifies devices and connectivity between devices.
- Layout Attributes Extractor - which extracts physical design attributes such as bounding box (die area), critical areas, and antennae.
- Circuit Attributes Extractor - which extracts attributes such as transistor parameters (e.g., sizes) and interconnect parameters (e.g., lengths, number of vias).
- Netlist Connectivity Extractor - which identifies functional blocks in a netlist and extracts connectivity between blocks.
- Netlist Attributes Extractor - which extracts structural attributes such as number of nets and net density from a netlist.

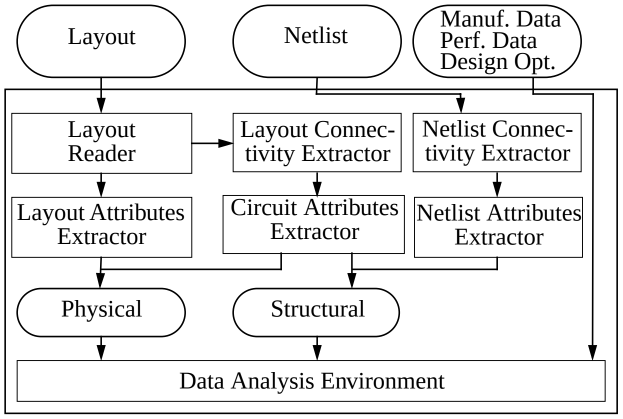

Layout
Netlist
Manuf. Data
Perf. Data
Design Opt.
Layout Reader
Layout Connectivity Extractor
Netlist Connectivity Extractor
Layout Attributes Extractor
Circuit Attributes Extractor
Netlist Attributes Extractor
Physical
Structural
Data Analysis Environment

Fig 1. Design Extraction Environment.

- Data Analysis Environment - which provides statistical and graphical routines to analyze relationships between extracted parameters and manufacturing/performance data of a IC.

The above framework has been implemented in a form of extraction environment called MAPEX 2 (MAnufacturability Parameter Extraction environment 2; the predecessor to MAPEX 2, MAPEX, was described in [4]).

MAPEX 2 takes as input a GDS layout and/or a Verilog netlist. It uses a commercial tool, Dracula [5], to extract physical attributes and device connectivity from a layout. Another tool, N-Stats, was developed to extract structural attributes from a netlist. The data analysis tools in MAPEX 2 are built using S-Plus [6].

In the remainder of this paper some of the algorithms implemented in MAPEX 2 are presented.

3.0 Select MAPEX 2 Capabilities

This section presents a set of algorithms to extract various IC design attributes that have been found to be relevant from a manufacturability perspective.

3.1 Physical Attribute Extraction

3.1.1 Extraction Tool Box

For a variety of extraction tasks, MAPEX 2 must perform a set of layout operations. These operations are made up of generic contour-based layout manipulation procedures [7] listed in Table 1. The more complex operations are describe below.

- *Overlap* $(A, K, N) \rightarrow C$*:* This operation places in set C those contours which enclose regions that are covered by N or more contours in set A, after they are expanded by K units. (See Figs. 2 and 3.)

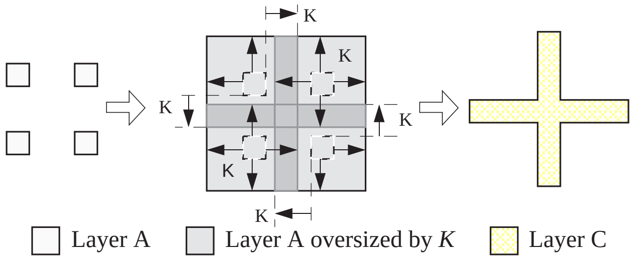


Fig 2. *Overlap(A, K, 2)* $\longrightarrow$ C operation.

[*] Currently with Level One Communications, Sacramento, CA.
[**] On leave from Institute of Microelectronics and Optoelectronics, Warsaw University of Technology, POLAND.

Table 1. Contour-based layout manipulation procedures.

Procedure	Explanation
$A \cap B$	Boolean AND of contours in *A* and *B*
$A \cup B$	Boolean OR of contours in *A* and *B*
Diff(A, B)	Selects contours of *A* not covered by contours in *B*
Enclose(A, B)	Selects contours of *A* enclosed by contours in *B*
Connect(A, B)	Labels contours in *B* with labels of overlapping contours in *A*
Select Node(A, "N")	Selects contours in *A* with label *"N"*
Select All(A, B)	Selects all contours in *A* and *B* that are connected
Total Area(A)	Calculates total area of contours in *A*
Area(A)	Calculates areas of contours in *A* with different labels
Length(A)	Calculates perimeter of contours in *A* with different labels

- *Over Size (A, K)* $\rightarrow$ *C*: This function increases or "grows" the contours from set *A* by *K* units. (See Fig. 4.)

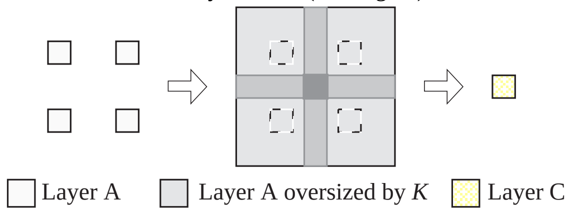


Fig 3. *Overlap(A, K,* 3*)* $\longrightarrow$ *C* operation.

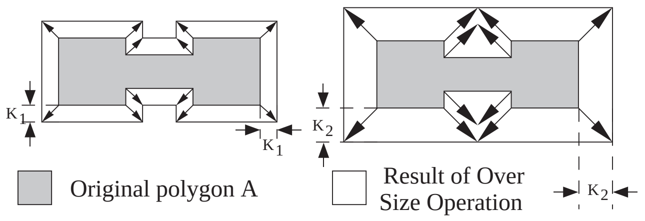


Fig 4. Two examples of the *Over Size(A, K)* $\longrightarrow$ *C* operation.

- *Under Size (A, K)* $\rightarrow$ *C:* Decreases or "shrinks" the contours of set *A* by *K* units. Observe that operation may lead to the "disappearance" of some segments (or the entire element) of the undersized set of contours (See Fig. 5).

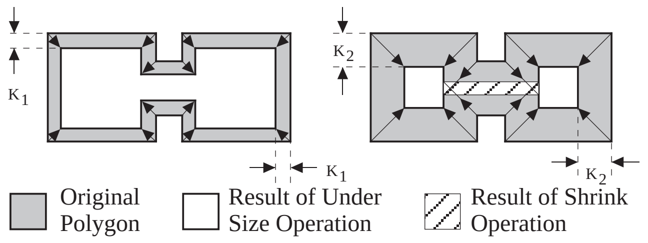


Fig 5. Examples of *Under Size* and *Shrink* operations

- *Shrink (A, K)* $\rightarrow$ *C:* Similar operation to *Under Size*. It also "shrinks" the contours of set *A* by *K* units. The difference between the *Shrink* operation and the *Under Size* operation is that *Shrink* places in set *C* only those contours that "disappear" after the *Under Size* operation. (See Fig. 5).

3.1.2 Extraction of Critical Area for Shorts

The defect sensitivity of a design is an important IC design attribute. One measure of a design's sensitivity is the critical area [8]. Critical area for defects of radius *r* is defined as a region on an IC layer where if a center of a defect of radius *r* is deposited, a fault (short or open) results. Fig. 6 illustrates the concept of critical area for extra material defects for three metal lines and two defect radii ($r_2 > r_1$).

This measure of layout sensitivity is especially effective when modeling the yield loss in high-volume mature fabrication lines [8,9,10,11,12].

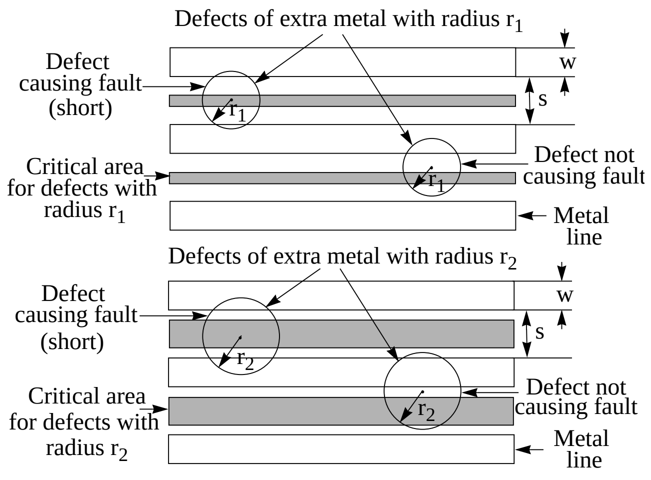

Defects of extra metal with radius r1
Defect causing fault (short)
w
s
r1
r1
Defect not causing fault
Critical area for defects with radius r1
Metal line
Defects of extra metal with radius r2
Defect causing fault (short)
w
s
r2
r2
Defect not causing fault
Critical area for defects with radius r2
Metal line

Fig 6. Critical area in an array of parallel metal lines.

Critical area for shorts can be computed in various ways [13,14]. Contour-based critical area extraction for shorts can be accomplished by using (for each conducting layer of the IC of interest) the following sequence of operations:

Pseudo Code 1:

```
For each defect radii $R_i$ {
    Overlap (M1, $R_i$, 2) $\rightarrow$ TMP1
    TMP1 $\cap$ BaseX $\rightarrow$ Crit_$R_i$
    Total Area (Crit_$R_i$) $\rightarrow$ Crit_Data
}
```

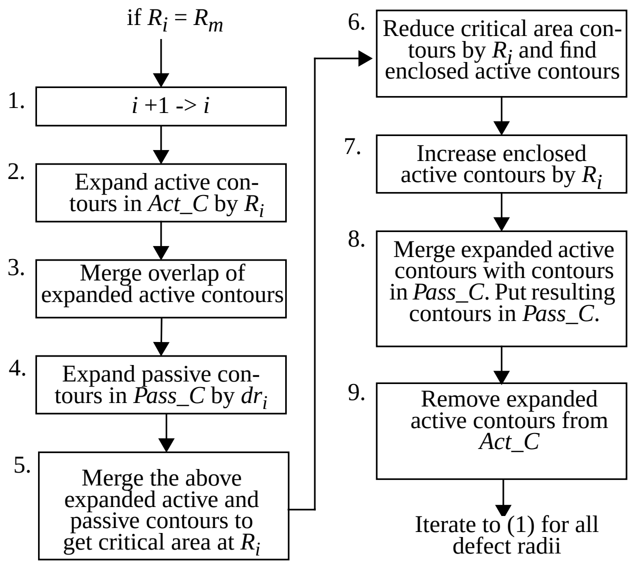
where *BaseX* is the region in which the critical area is being extracted, *Crit_Data* is the extracted critical area data.

The problem with Pseudo Code 1 lies in the complexity of the involved contour manipulation procedures taken from Table 1. Essentially, the worst case complexity of these procedures is $O(n^2)$, where *n* is the number of geometrical shapes in the layout that a single defect may intersect. Such a complexity is unacceptable for the critical area extraction of large defects for a circuit with millions of transistors. To overcome this obstacle, a new critical area algorithm has been proposed [7]. This algorithm extracts critical area incrementally, using for a subsequent value of defect radius information obtained from the extraction of critical areas previously computed at smaller defect radii. Such a strategy is especially effective at reducing the number of geometrical shapes that a large defect may intersect. Consequently, the run time is dramatically reduced.

Pseudo Code 2 describes the extraction algorithm which uses the above strategy. It essential breaks down the extraction process into two parts. The first part extracts critical area for small defects using the operations of Pseudo Code 1. (The latter is more effective for small defects.) In the Pseudo Code 2 the following terms are used:

- *Passive Contour* - a contour whose's expanded shape at a given defect radius is enclosed by the critical areas at the same defect radius.
- *Active Contour* - a non-passive contour in the layout.
- *Pass_C* - set of passive contours.
- *Act_C* - set of active contours.
- *Crit_R_i* - set of critical area contours at a defect radius of R_i.
- dr_i - incremental difference between defect radius R_i and R_{i+1}.

The second part extracts critical area for large defects. From experimentation it was found that the second part is more effective for defects with radii greater than $R_m = s + w/2$, where s and w are the minimum spacing and width design rules. A flow chart of the second part of the Pseudo Code 2 is shown in Fig. 7. Graphical illustration of Pseudo Code 2 is given in Fig. 8.



if $R_i = R_m$
1.
i +1 -> i
2.
Expand active contours in *Act_C* by R_i
3.
Merge overlap of expanded active contours
4.
Expand passive contours in *Pass_C* by dr_i
5.
Merge the above expanded active and passive contours to get critical area at R_i
6.
Reduce critical area contours by R_i and find enclosed active contours
7.
Increase enclosed active contours by R_i
8.
Merge expanded active contours with contours in *Pass_C*. Put resulting contours in *Pass_C*.
9.
Remove expanded active contours from *Act_C*
Iterate to (1) for all defect radii

Fig 7. Flow chart of second part of Pseudo Code 2.

Pseudo Code 2:

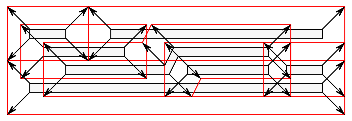
```
Ø ⟶ Pass_C
M1 ⟶ Act_C /* place metal 1 contours in active contour set */
/* Part 1 - extraction procedure for small defects */
For each defect radii $R_i \le R_m$ {
    Overlap (Act_C, $R_i$, 2) ⟶ TMP1
    TMP1 ∩ BaseX ⟶ Crit_$R_i$
    Total Area (Crit_$R_i$) ⟶ Crit_Data
}
/* Part 2 - extraction procedure for large defects */
For each defect radii $R_i > R_m$ {
    Overlap (Act_C, $R_i$, 2) ⟶ TMP2
    Over Size (Pass_C, $R_i - R_{i-1}$) ⟶ TMP3
    TMP2 ∪ TMP3 ⟶ TMP4
    TMP4 ∩ BaseX ⟶ Crit_$R_i$
    Total Area (Crit_$R_i$) ⟶ Crit_Data
    Under Size (Crit_$R_i$, $R_i$) ⟶ TMP5
    Enclose (Act_C, TMP5) ⟶ TMP6
    Over Size (TMP6, $R_i$) ⟶ TMP7
    TMP7 ∪ TMP3 ⟶ Pass_C
    Diff(Act_C, Pass_C) ⟶ Act_C
}
```

Examples of the results obtained with the above critical area extraction algorithm are shown in Fig. 9 for three IC designs. The relevant aspects of the designs are also given in the figure, as are the execution times. The algorithm described by Pseudo Code 2 speeds up extraction by polynomial factor as a function of defect radii [7]. Fig. 10 shows the critical area curves obtained for different layers for the layout of Design B.

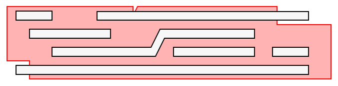
3.1.3 Extraction of Critical Area for Opens

There are two attributes of a "good" electrical connection in an IC chip - physical continuity of the conducting paths and good electrical contact between conducting layers. This section discusses extraction of critical area for opens – a measure of IC layout sensitivity to missing-material spot defects – by considering both interconnect continuity and inter-layer contacts [15,16].

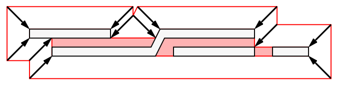
A typical interconnect is a chain of conducting segments on different layers and a set of contact plugs formed to provide connec-



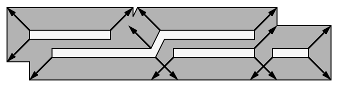
I. Expand contours by R_i (Step 2).



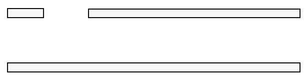
II. Extract critical area contours at R_i (step 5).



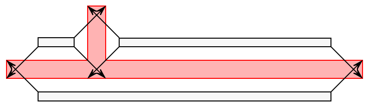
III. Reduce critical area contours by R_i and identify enclosed active contours (step 6).



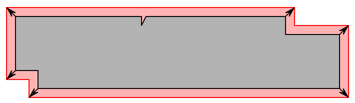
IV. Expand enclosed active contours in **III** by R_i and store in *Pass_C* (steps 7 and 8)



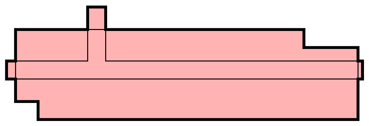
V. Remove enclosed active contours in **III** from *Act_C* (step 9).



VI. Expand remaining active contours and extract critical area contours for defect radii R_{i+1} (steps 2 and 3)



VII. Expand passive contours by dr_{i+1} (steps 4)



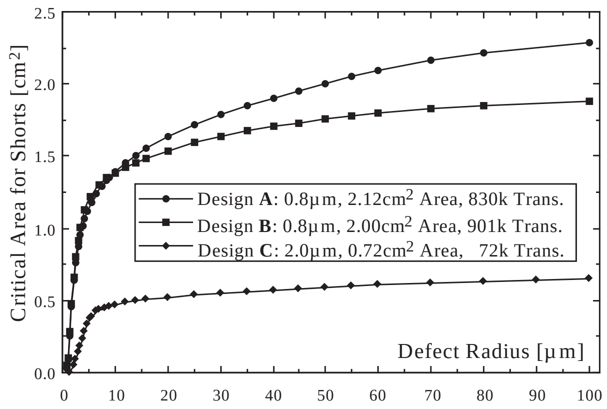
VIII. Merge contours in **VI** and **VII** to get critical area contours at R_{i+1} (steps 5)

Fig 8. Graphical examples of critical area extraction algorithm. First example **I** begins at $R_i = R_m + dr_i$, i.e., *Pass_C* = Ø (steps refer to flow chart in Fig. 7).

tion between these segments. Hence, to assess the probability of an open one needs to compute the critical area for each segment of an interconnect and for the contacts in the analyzed layout. Any segment of a VLSI interconnect can be viewed as a conducting path terminated with two or more contacting regions (see Fig. 11). Contacting regions are areas of the physical ohmic junction between different conducting layers.

A conducting path is broken (open) if a spot defect spans two opposite edges of the conducting path. Consequently, the critical area of a conducting path for a defect of radius R, can be determined by constructing a set of points located within a distance equal to R from both edges of this path. Observe that such a set can be obtained by appropriate shifts of the edges of the conducting path (Fig. 11).

An open in a contacting region is assumed to occur when a defect covers the entire contacting area. To extract the critical area for contacts the algorithm in Pseudo Code 3 is used [16]. A flow chart of the algorithm is given in Fig. 12. Illustration of the manipulation procedures for the extraction of critical areas of contact



Design	Total Run Time [hours]	
	Defects up to 10 µm	Defects up to 100 µm
A	45.69	237.60
B	41.35	164.29
C	1.49	1.85

Fig 9. Extracted m1 critical areas for 3 ICs and extraction times.

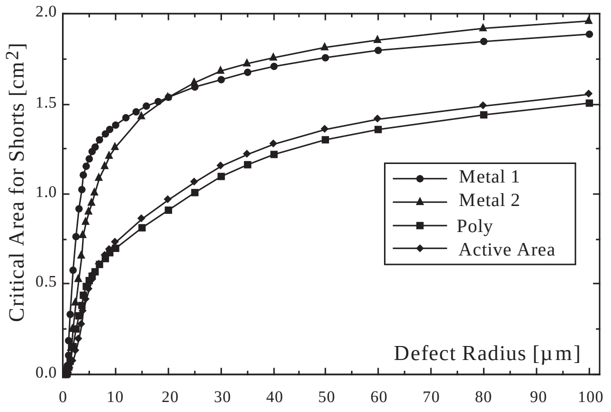
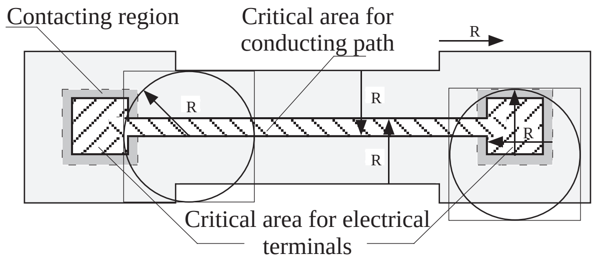


Fig 10. Critical are curves for different layers for an IC.



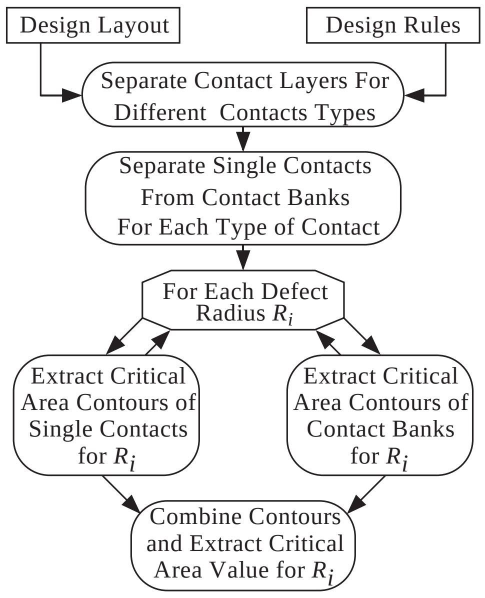
Contacting region
Critical area for conducting path
R
Critical area for electrical terminals

Fig 11. Critical area for an open caused by a defect of radius R

banks are given in Fig. 13. (Note, that this algorithm is applicable to the extraction of critical areas of single contacts and of contact banks).

In the Pseudo Code 3 the following definitions are used:

- L_j - set of contacts of type *j* for a given layer (e.g., for the metal1 layer, contact types are metal1/poly contact or metal1/metal1 via).
- M_j - the minimum spacing between non-equipotential contacts of type *j*. Non-equipotential contacts are those that are not part of a contact bank.
- *Nbank* - is the failure criteria for contact banks, i.e., the number of contacts that must fail in a contact bank for the entire bank to fail.
- *w* - minimum width design rule of contact cut.



Design Layout
Design Rules
Separate Contact Layers For Different Contacts Types
Separate Single Contacts From Contact Banks For Each Type of Contact
For Each Defect Radius R_i
Extract Critical Area Contours of Single Contacts for R_i
Extract Critical Area Contours of Contact Banks for R_i
Combine Contours and Extract Critical Area Value for R_i

Fig 12. Flow chart for Pseudo Code 3.

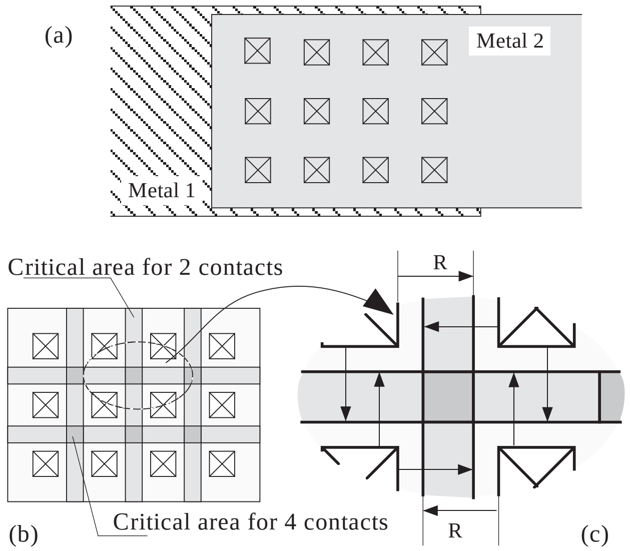


Fig 13. Critical area extraction example for contact banks.

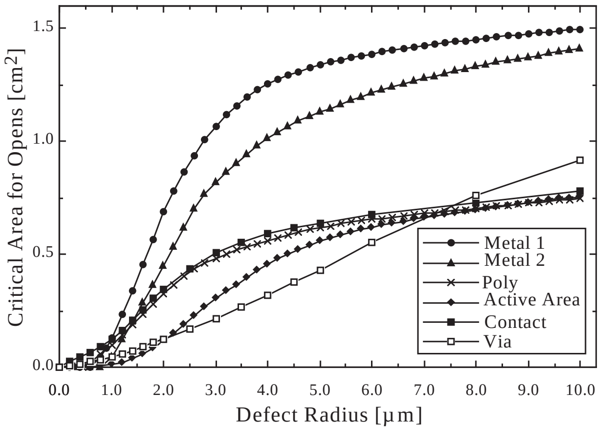
Note, to accommodate the "rounding" of the contacts due to manufacturing, the following scaling function is used [16]:

$$f(R_i) = \frac{\sqrt{\pi}}{2}\left(R_i + \frac{w}{2}\right) - \frac{w}{2}$$

Pseudo Code 3:

```
/* Step 1: Separate single contacts from contact banks */
Ø ⟶ SINGLE
For each $L_j$ {
    Under Size(Over Size ($L_j$, $Mj/2-\delta$), $Mj/2-\delta$) ⟶ TMP1
    Enclose(TMP1, $L_j$) ∪ SINGLEj ⟶ SINGLEj
    Diff($L_j$, SINGLE) ⟶ $BANK_j$
}
/* Step 2: Extract critical area of contacts */
For each $R_i$ {
    If ($f(R_i) < M/2$) then {
        Ø ⟶ TMP2
        For each $L_j$ {
            Over Size(SINGLEj, f($R_i$)) ∪ TMP2 ⟶ TMP2
            Overlap(BANK$_j$, f($R_i$), Nbank) ∪ TMP2 ⟶ TMP2
        }
    }
    else {
        Over Size(TMP2, f($R_i$) - f($R_{(i-1)}$)) ⟶ TMP2
    }
    Total Area(TMP2) ⟶ Carea.dat
}
```

The above Pseudo Code was used to extract critical areas for opens from the layout of an industrial design (900,000 transistors, 0.8 µm CMOS technology). The extracted critical area data is shown in Fig. 14. The table in Fig. 14 gives a summary of extraction times.



Layer	Metal 1	Metal 2	Poly	Active	Contact	Via	Total
Time [h]	72.14	16.41	17.43	20.50	7.43	2.45	136.36

Fig 14. Critical area extraction example for opens.

3.1.4 Extraction of Area of Minimum Spacing and Width.

The sum of the spacing between minimum spaced metal lines has been shown to be a useful attribute for estimating the yield of an IC [12,17]. In particular, this attribute can be used to derive the initial rise in slope of the critical area curve [12]. Since large defects occur much less frequently than small defects, this initial rise is the most relevant portion of the critical area curve. The sum of the spacing between minimum spaced metal lines, *Aminsp*, can be derived for the metal 1 layer (M1) using the following pseudo code:

Pseudo Code 4:

```
Overlap(M1,s/2+δ,2) → M1_ol
Total Area(M1_ol) → Acr_sp
Acr_sp * s / δ → Aminsp
```

Here *s* is the minimum spacing between metal layers and $\delta << s/2$ (for example, δ can be smaller than the grid resolution of the layout). This value of δ forces the above operations to extract the critical areas of only those lines that are a minimum spacing apart. This critical area, *Acr_sp*, turns out to be approximately proportional to the sum of the area between minimum spaced metal lines. The latter can be derived by multiplying *Acr_sp* by *s* and dividing by δ.

The total area of interconnects of minimum width is also a useful measure of the sensitivity of a design to defects. However, this sensitivity attribute represents missing material defects. The area, *Acr_wd* can be extracted as follows:

Pseudo Code 5:

```
Under Size(M1,w) ⟶ TMP1
Over Size(TMP1,w) ⟶ TMP2
Diff(M1,TMP2) ⟶ TMP3
Total Area(TMP3) ⟶ Acr_wd
```

In the above code *w* is the minimum width design rule.

3.1.5 Extraction of "Antenna" Condition

Small feature sizes in modern ICs are typically achieved using dry plasma-based processes. The problem with plasma processes is that they tend to charge floating conducting regions which are not connected to diffusion. (Until the last metal layer is laid down and etched, not all conductors will have paths to diffusion.) If such conductors are connected to the gate oxide, this charging can cause currents in the gate oxide, particularly in the proximity of oxide defects. Such currents can induce more trap states which in turn can amplify gate currents. In an extreme case, this can lead to gate oxide breakdown. This effect is called the "antenna effect" [18] due to the tendency of the floating conductors to act as antennae for gate oxide charging.

Floating conductors not connected to diffusion can occur during the following process steps: (a) polysilicon etching (b) etching of all metal layers (except the top metal layer), and (c) contact/via etching. To evaluate the susceptibility of a design to the antenna effect, therefore, it is necessary to extract the following attributes from the design (see [19]):

- For gate oxide: area, perimeter, source/drain length, bird's beak length.
- For contacts/vias: area and perimeter length
- For any conducting layer (except the top metal layer): area and perimeter length.

In addition, all these attributes have to be extracted for only the floating portions of the interconnect. Therefore the extraction procedure must also be able to find the connectivity status of each node at each manufacturing step. The complete algorithm for extraction of antenna parameters is given in [19]. In this paper, only two pseudo-codes are listed as examples.

Pseudo Code 6 (Extraction of gate oxide parameters):

```
Poly ∩ Active ⟶ GOX      /* Get the gate oxide region GOX */
Area(GOX) ⟶ File1        /* Area of gate oxide region */
Length(GOX) ⟶ File2      /* Perimeter of gate oxide region */
Over Size(GOX, δ) ⟶ TMP1 /* Oversize gate region by d */
(Diff((TMP1 ∩ Active), GOX))/δ ⟶ SDE /* Obtain s/d edge
length */
(Diff((TMP1 ∩ Poly), GOX))/δ ⟶ BBE /* Obtain the bird's
beak edge length */
```

Pseudo Code 7 (Extraction of poly antennae):

Since all poly nodes crossing active areas form antennae (because in standard CMOS there is no poly-active contact), the extraction pseudo code is:

```
Connect(Poly, GOX) /* Connect poly and gate oxide */
Select All(Poly, GOX) ⟶ CPGX /* Put poly nodes connected to
GOX in CPGX */
Total Area(CPGX) ⟶ File1 /* Poly antenna area */
Length(CPGX) ⟶ File2 /* Poly antenna perimeter */
```

3.2 Structural Design Attributes

In [20] it was shown that a good estimate of the layout density or critical area of an IC can be derived from a measure of "congestion of the netlist". It was also shown that a good measure of the congestion is the average neighborhood population of the circuit. The concept of neighborhood population has been used previously to model interconnect lengths [21] and is defined here in the following way. Let *Distance(cell1,cell2)* be the number of cells traversed in the shortest path between cells *1* and *2* (in Fig. 15, *Distance(a,g)*=3). The neighborhood population of a cell at level *i*, $Ngh(cell)_i$, is the number of cells residing within a distance *i* from the cell (in Fig. 15, $Ngh(d)_1=|\{c,e,f,g,h\}|=5$, and $Ngh(d)_2=|\{a,b\}|=2$). The total neighborhood population at level *i*, $Angh_i$, is the sum of neighborhood populations for all cells at level *i*. E.g., in Fig. 15, $Angh_1$=22.

Pseudo Code 8 can be used to extract the average neighborhood population of an IC design. It makes use of a routine *Get_Neighbors()* that recursively traces through nets to determine

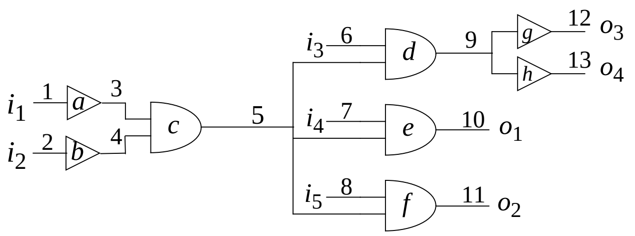


Fig 15. Example circuit used to define neighborhood population.

how many cells are within a given distance from a particular cell. Cells previously encountered by *Get_Neighbors()* are marked using the routine *Mark()*. This places a marker on the cell to indicate at which level (or distance from the original cell) the cell was encountered. If subsequently the cell is re-encountered at a lower level the cell is remarked and the neighborhood populations are adjusted. *Unmark_All_Cells()* sets the level of all cells to 0.

Pseudo Code 8:

```
1 $\longrightarrow$ level
Unmark_All_Cells()
For each $Cell_i$ in design {
  Mark($Cell_i$, level)
  For each $Net_i$ connected to $Cell_i${
    Get_Neighbors($Net_i$, level)
    Unmark_All_Cells()
  }
}
For each $level_i$
  Neighbors[$level_i$]/Ncell $\longrightarrow$ Avg_Neighbors[$level_i$]
/* recursive routine to extract neighborhood levels */
Get_Neighbors($Net_i$, level) {
  For each $Cell_k$ connected to $Net_i$ {
    Check_Marked($Cell_k$) $\longrightarrow$ level_cell
    If (level_cell $\neq$ 0 && level_cell $\leq$ level) then continue
    If (level_cell > level) then
      Neighbors[level_cell]-1 $\longrightarrow$ Neighbors[level_cell]
    Neighbors[level]+1 $\longrightarrow$ Neighbors[level]
    Mark($Cell_k$, level)
    If (level < level_max) then
      For each $Net_l$ connected to $Cell_k$
        Get_Neighbors($Net_l$, level+1)
  }
```

4.0 Conclusions

In this paper the key components of a design attributes extraction environment were described. In addition, representative algorithms to extract some of the more important design attributes relevant to assessing manufacturability, were presented. The chosen attributes, namely critical areas, minimum spacings and widths, and antennae, represent only some of the DFM IC design attributes. But they do illustrate the wide spectrum of extraction capabilities that one must implement.

The primary objective of any extraction environment is to provide the necessary data needed for cross-correlating design data with manufacturing data. Identifying key correlations provides manufacturing engineers with valuable information regarding deficiencies in the process [3]. It also provides designers and CAD tool developers with knowledge as to which design attributes should be minimized, and can point to sensitive areas in layouts that are susceptible to the dominant yield loss mechanisms [3].

The above capabilities are key elements needed to bridge the growing gap found in the design/manufacturing interface.

Acknowledgments

The authors would like to thank SRC, Sematech, AMD, National Semiconductor, Philips, and Siemens for financial and technical support.

References

[1] W. Maly, “Future of IC design, testing and manufacturing,” *IEEE Design and Test of Computers*, vol. 13, no. 4, pp. 8, 89-91, Winter 1996.

[2] W. Maly, H.T. Heineken, J. Khare, and P. K. Nag, “Design for manufacturability in submicron domain,” *Proc. of ICCAD-96*, pp. 690-697, Nov. 1996.

[3] W. Maly: Ed., “Design for manufacturability for next decade,” CMU-SRC Research Report No. CMUCAD-97-10, Pittsburgh, USA, March 1997.

[4] H.T. Heineken and W. Maly, “Manufacturability analysis environment - MAPEX,” *Custom Integrated Circuits Conference*, pp. 309-312, May 1994

[5] *Dracula 4.1*, Reference Manual Set, Cadence Design Systems Inc., San Jose, CA, 1993.

[6] *S-Plus 3.1*, Reference Manual Set, Statistical Science Inc., Seattle, WA, Oct. 1992.

[7] C. Ouyang and W. Maly, “Efficient extraction of critical area in large VLSI ICs,” *International Symposium on Semiconductor Manufacturing*, pp. 301-304, Oct. 1996.

[8] W. Maly and J. Deszczka, “Yield estimation model for VLSI artwork evaluations,” *Electron. Lett.*, vol. 19, no. 6, pp. 226-227, March 1983.

[9] C. H. Stapper, “Modeling of defects in integrated circuit photolithographic patterns,” *IBM J. Res. Develop.*, vol. 28, no. 4, pp. 461-474, July 1984.

[10] A.V. Ferris-Prabhu, “Modeling the critical area in yield forecasts,” *IEEE J. Solid-State Circuits*, vol. SC-20, no. 4, pp. 874-878, Aug. 1985.

[11] W. Maly, “Computer-aided design for VLSI circuit manufacturability,” *Proc. of the IEEE*, vol. 78, no. 25, pp. 356-392, Feb. 1990.

[12] H.T. Heineken, J. Khare, and W. Maly, “Yield loss forecasting in the early phases of the VLSI design process,” *Custom Integrated Circuits Conference*, pp. 27-30, May 1996.

[13] P. K. Nag and W. Maly, “Hierarchical extraction of critical area for shorts in very large ICs,” *IEEE Int. Workshop on Defect and Fault Tolerance in VLSI Systems*, pp. 10-18, Nov. 1995.

[14] Gerard A. Allan and Anthony J. Walton, “Yield prediction by sampling with the EYES tool,” *IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Systems*, pp. 39-47, Nov. 1996.

[15] W. A. Pleskacz and W. B. Kuzmicz, “SENSAT – a practical tool for estimation of the IC layout sensitivity to spot defects,” *Proc. of the European Design and Test Conference*, pp. 598, March 1995.

[16] C. Ouyang, W. A. Pleskacz, and W. Maly, “Extraction of critical areas for opens in large VLSI circuits,” *IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Systems*, pp. 21-29, Nov. 1996.

[17] D. Schmitt-Landsiedel, D. Keitel-Schulz, J. Khare, S. Griep, and W. Maly, “Critical area analysis for design-based yield improvement of VLSI circuits,” *Quality and Reliability Engineering International*, Vol. 11, pp. 227-232, 1995.

[18] S. Fang and J. McVittie, “Thin-oxide damage from gate charging during plasma processing,” *IEEE Electron Device Letters*, vol. 13, no. 5, p. 288, May 1992.

[19] W. Maly, C. Ouyang, S. Ghosh, and S. Maturi, “Detection of an antenna effect in VLSI designs,” *IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Systems*, pp. 86-94, Nov. 1996.

[20] H.T. Heineken and W. Maly, “Interconnect yield model for manufacturability prediction in synthesis of standard cell based designs,” *IEEE International Conference on Computer-Aided Design*, pp. 368-373, Nov. 1996.

[21] M. Pedram and B. Preas, “Interconnection length estimation for optimized standard cell layouts,” *Int. Conference on Computer-Aided Design*, pp. 390-393, Nov. 1989.