Design-Aware Mask Inspection

Abde Ali Kagalwalla*, Puneet Gupta*, Chris Progler† and Steve McDonald†
(abdeali, puneet)@ee.ucla.edu
*Department of Electrical Engineering, University of California, Los Angeles
†Photronics Inc.

***Abstract*—Mask inspection has become a major bottleneck in the manufacturing flow taking up as much as 30% of the total manufacturing time [15]. In this work we explore techniques to improve the reticle inspection flow by increasing its design awareness. We develop an algorithm to locate non-functional features in a post-OPC layout with 100% accuracy without using any design information. Using this, and timing information of the design (if available), we assign a minimum size defect to each reticle feature that could cause the design to fail. The criticality of various reticle features is then used to partition the reticle such that each partition is inspected at a different pixel size and sensitivity so that the false+nuisance defect count is reduced without missing any critical defect. Up to 4X improvement in false+nuisance defect count is observed with our technique resulting in up to 55% improvement in first pass yield coming from reduction in nuisance defects and substantial reduction in defect review load.**

I. Introduction

A reticle (mask) is basically a stencil that determines what patterns eventually print onto the wafer. The increasing aggressiveness of various RET techniques like OPC, PSM, and scattering bar along with decreasing feature sizes has increased the complexity of reticles considerably [12]. Their increasing complexity has also increased the burden of reticle inspection tools. In fact, mask inspection is more challenging than mask writing itself [15]. High resolution reticle inspection tools are required to detect every potential printable defect in order to prevent yield loss. But this also produces a large number of "nuisance" defects, i.e. defects that do not affect yield. As a result post-inspection review of defects has become very time consuming. The overall inspection flow has a considerable impact on mask cost and turnaround time (TAT). Keeping mask cost in control is extremely critical, especially for low volume SoCs. The problem is likely to get worse for future patterning technologies (e.g. multi-layer EUV lithography and nanoimprint templates). Hence there is a strong need to improve the inspection flow.

A. Mask Inspection Primer

A comprehensive inspection of reticles must be done by the mask shops before sending it to the fabs. The basic steps of inspection are shown in Figure 1.

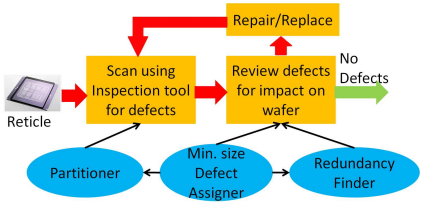

Fig. 1. Key steps of reticle inspection along with our contributions in circular blocks

Initially the reticle is passed through an inspection tool (e.g. KLA-Tencor's Terascan [8] or NEC's LM series [18]) which takes an image of a die and compares it to a reference database or another die (die-die or die-database modes). The difference between the two images is found and if the intensity of the difference exceeds a predefined threshold, the difference pattern is labeled a defect. The inverse of this threshold is referred to as sensitivity. These tools can have a pixel size as low as 55nm and can detect critical dimention(CD) defects as small as 20nm on the mask at maximum sensitivity (minimum threshold) [8]. Inspection tools generate a very large number of defects (100+) most of which do not impact the final design. Defects can be classified as shown in Figure 2. A *false defect* is an incorrect detection reported by the inspection tool due to vibration, misalignment, optical distortion, error in database rendering (die-database mode), etc. Real defects are caused either due to misalignment or vibration of the mask writer (CD defects) or contamination of the mask (contamination defects). Inspection tools typically have different algorithms to detect these two categories of defects and hence have different sensitivities for these defects. Many real defects do not print on the wafer. Among printable defects, some lie on non-critical regions of the design such as dummy fill or redundant vias. Only a small fraction of the defects reported by the inspection tool really matter. All the non-printable and non-critical defects are also called *nuisance defects*. Reducing the number of false+nuisance defects reported by the inspection tool is essential to reduce inspection cost. Reducing nuisance defects is particularly important to maskshops as it impacts *first pass yield*, which is the fraction of total masks manufactured that can be shipped without repair or detailed review.

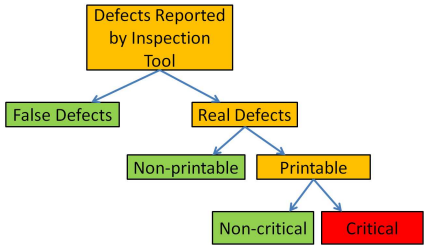

Fig. 2. Various categories of defects reported by Inspection Tool

The next step in mask inspection is defect review where each defect reported by the inspection tool is checked to find out if it really matters. False, non-printable and non-critical defects are filtered out during this step. Images of defects reported by the inspection tool are analyzed using software tools [17], [19] or manually. Often defect images need to be recaptured at a better resolution. For this the inspection tool could be reused (Online Review) or an e-beam is employed [16]. After pruning out a significant fraction of false/non-printable/non-critical defects the mask is passed through an aerial imaging tool (e.g. Carl Zeiss AIMS system [10]). AIMS is essentially a hardware emulator of the wafer stepper that operates at the same optical settings as the stepper and gives a very accurate estimate of the printability of defects. Although extremely accurate, AIMS is slow and cumbersome. Hence, minimizing the number of defects that have to pass through AIMS tools is important in order to ensure reasonable turnaround time. Defects which are found to be printable by the AIMS tool are then either repaired or if they are unrepairable the reticle must be replaced. The repaired or replaced reticle must

again go through this inspection cycle. Because of the manual steps and use of AIMS tool, defect review is typically the slowest part of reticle inspection.

B. Related Work

There has been considerable work to improve mask manufacturing by using design intent. For instance, [6], [13] use electrical and design metrics to reduce OPC runtime and mask write cost. These approaches indicate that considerable benefit can be derived by using design intent to reduce the inherent pessimism of various mask manufacturing steps, including inspection.

The traditional approach to mask inspection discussed above does not use any design information to assess the criticality of defects. It assumes that all printable defects larger than a threshold size (say 10% of mask critical dimension) are critical. If design information is available to maskshops and fabs, they might be able to avoid the expensive process of repair/replacement of the mask due to non-critical defects. Design information can also be used to reduce false and nuisance defects reported by the Inspection tool. Communicating design intent to the inspection tool in the form of additional control layers has been suggested before [14], [22], [23]. Mask shops can use design information to lower the inspection sensitivity of non-critical regions in order to reduce the number of false+nuisance defects. Hedges et al. [14] have shown that up to 100X reduction in nuisance defect count is possible just by using variable sensitivity during reticle inspection. Current inspection tools allow the user to define inspection sensitivity on a per pixel basis. But memory requirements to store this sensitivity information are impractical since a reticle can have up to 10^{12} pixels. These approaches assume that maskshops know the design criticality of the layout which is rarely the case. Driessen et al [9] analyze a post-OPC layout to extract some non-critical features in the absence of any design data. Stoler et al [21] extract some criticality information as part of Manufacturing rule check(MRC). Both these approaches focus on extracting assist features from the layout which are a major source of nuisance defects.

C. Our Work

Key contributions of this work are as follows:

- We develop an graph based algorithm to locate non-functional features (redundant and dummy features) in a post-OPC layout (flat and 10X more complex than pre-OPC layout) in the absence of any design information. Location of non-functional regions can be used during defect review to prune out non-critical defects and also for assigning minimum size defects as discussed below and shown in Figure 1.
- We assign minimum defect size that impacts the design to each feature of the reticle for both CD and contamination defects. This is inferred using the timing slack of critical paths and the location of non-functional features found using the method mentioned above.
- Using the minimum defect size of each feature of a reticle, we partition the layout using a scan-line based heuristic, where each partition is assigned a different pixel size and CD and contamination sensitivity to minimize false+nuisance defects. Apart from sensitivity, pixel size can also be varied during inspection since most tools have multiple pixel sizes for scanning the reticle. Although inspecting different regions of reticle at different pixel size can reduce false+nuisance defects, it does require multiple scans of the reticle in current generation tools, increasing inspection runtime.

Remaining paper is organised as follows. Section II discusses the non-functional feature finding problem. Section III and IV describe the criticality assignment and partitioning problems, respectively. Results are covered in Section V. Section VI concludes the paper.

II. Non-functional Feature Finding

A. Problem Formulation

Given a post-OPC layout identify non-functional features of the layout.

Although we focus only on redundant vias and dummy fill in this work other non-functional features such as spare cells, non-tree routes and assist features can also be found using our graph based methodology . We assume that the layout has only rectilinear shapes, and that floating dummy fill in different metal layers are not connected through a via. This is consistent with most commercial fill synthesis tools [1].

B. Solution

We fracture the layout into rectangles, use a scan-line based algorithm to construct a neighborhood graph for these rectangles which is then reduced. This reduced neighborhood graph (RNG) can then help identify dummy fill and redundant vias. The various steps are detailed below.

1) Algorithm Steps:

- *Fracturing Polygons:* The rectilinear polygons are fractured into rectangles using a simple horizontal slicing method [11]. The rectangles are then stored in different sets based on their layer. For example, a rectangle corresponding to a Metal 2 shape is stored in two sets, M_2V_1 and M_2V_2. A set M_iV_j corresponds to all rectangles belonging to same/adjacent metal or via layers.
- *Neighborhood Graph Construction:* The new layout with fractured polygons is used to construct an undirected Neighborhood Graph, $G(V, E)$ in which every rectangle of the fractured layout corresponds to a vertex and edge $(u, v) \in E$ if the two corresponding rectangles are physically in contact with each other in the layout.
  A scan-line based one-pass, optimal algorithm is used to solve the rectangle intersection problem as described in [20]. The problem is reduced to two subproblems, an interval query and a point query. Interval tree and range tree are two "semi-dynamic" tree datastructures that are used to solve this problem. We shall refer to these two sets of trees as scan-line trees. A separate scan-line is used for each set M_iV_j but there is a single graph for the entire layout. Both these trees can perform INTERSECTSEARCH, INSERT and DELETE operations in $O(log(m))$, where m is the number of nodes in the tree, which depends linearly on the number of rectangles [2].
- *Edge Contraction:* All neighboring vertices of the Neighborhood Graph that correspond to rectangles of the same layer are merged. At the end of this operation each vertex has an edge only to vertices belonging to an adjacent layer. Hence, a vertex corresponding to Metal 2 in RNG will have edges only to vertices of Via 1 or Via 2 and so on.
- *Graph Analysis:* Floating fill is identified by looking for isolated vertices. Cycles in the RNG correspond to redundant vias which can be identified using DFS. Double and even multi-cut vias can be identified by scanning the reported cycles and identifying the set of vias connected to the same pair of metal layer vertices in RNG.

[1] Future work can extend this to grounded fill and via fill as well.

[2] INTERSECTSEARCH returns all rectangles stored in the scan-line tree that intersect the input rectangle and constructs edges in the neighborhood graph between the input and all returned rectangles

2) Runtime Optimization Techniques:

- *Routing-aware scan-line:* The routing direction of each set of rectangles, M_iV_j can be found by taking the larger of the average length and width of all rectangles in the set. If the routing direction is X(Y) we define y(x) coordinates of the rectangles as scan-line events so that the average duration for which a rectangle needs to be stored in the tree reduces, thus improving INTERSECTSEARCH time.
- *Shape Simplification:* Before fracturing the polygons into rectangles, we perform shape approximation on the post-OPC polygons to reduce the number of rectangles created after fracturing. We create two sets of buckets for the coordinates of each polygon. Each point is included in two buckets, one in x-direction and another in y such that x(or y) coordinate of each point in a bucket is within a certain threshold distance of others. All the x(or y) coordinates of a bucket are then changed to the average x (or y) coordinate of the corresponding bucket. This approach reduces small deviations along a straight line as shown in Figure 3 and hence reduces rectangle count.

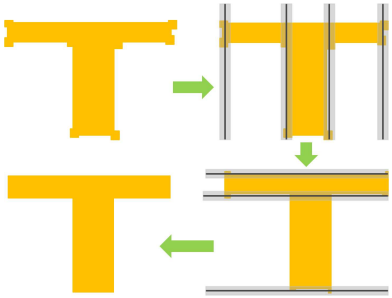

Fig. 3. Shape simplification for a distored T-shape

Algorithm 1 summarizes the entire algorithm. Figure 4 illustrates the complete algorithm for a sample double via. The scan-line based graph construction is the slowest operation in this algorithm. The total number of events are $2N$, where N is the number of rectangles in the layout. Each scan-line tree operation can be completed in $O(log(m))$, where m is number of events in the tree which can be $O(N)$ at worst. Hence the algorithm runtime is $O(Nlog(N))$.

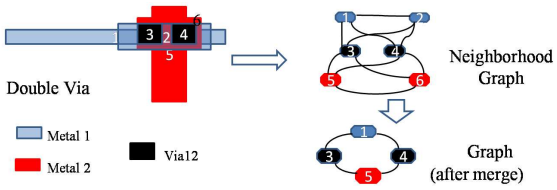

Fig. 4. Illustration of various steps of Redundancy-finding algorithm

III. Criticality Assignment

A. Problem Formulation

Given the timing of critical paths and non-functional features identified in Section II, find the minimum size defect at each location in layout which can cause failure.

For poly layer, we use timing slack of various paths to assign minimum size defect for each polysilicon shape corresponding to a transistor pair (PMOS+NMOS) on the critical path. Location of redundant vias is used to assign minimum size defect for via layer and location of dummy fill for metal layer.

Algorithm 1 Non-functional feature finding

Require: Shapes of all metal and via layers, S.
1: **for all** Shape $s \in S$ **do**
2: SHAPE-SIMPLIFICATION(s)
3: Set of rectangles, B_s = FRACTURE(s)
4: Store B_s in set M_iV_j corresponding to shape layer
5: **end for**
//EVENT DEFINITION
6: Find routing direction R of each rectangle set, M_iV_j
7: **if** Routing direction R is X(Y) **then**
8: Store $bottom(left)$ and $top(right)$ of each rectangle in set as separate events in E_{ij}.
9: **end if**
//SCAN-LINE
10: **for all** Events $e \in E_{ij}$ for each set E_{ij} **do**
11: **if** e is $bottom(left)$ **then**
12: INTERSECTSEARCH(Scan-line Tree, $e.rect$)
13: INSERT(Scan-line Tree, $e.rect$)
14: **else**
15: DELETE(Scan-line Tree, $e.rect$)
16: **end if**
17: **end for**
//EDGE CONTRACTION
18: Edge Contract $G(V, E)$ to obtain RNG $G(V', E')$
//GRAPH ANALYSIS
19: Mark all isolated vertices as dummy fill
20: Find cycles in $G(V', E')$ using DFS to detect redundant vias

TABLE I
Impact of different defect types in Poly layer

Type	Gate Length	Design Impact
Intrusion	Decrease	Open/Delay Decrease
Extrusion	Increase	Short/Delay Increase
Pinhole	Decrease	Open
Pindot	No change	None

B. Solution

On the basis of geometry, reticle defects are classified as pindots, pinholes, intrusion and extrusion. Intrusion and extrusion are considered CD defects. Pindots and pinholes are usually classified as contamination defects. Apart from size, type and location of defect, CD impact on the wafer also depends on the type of reticle (bright-field or dark-field), type of resist (positive or negative) and mask error enhancement factor (MEEF) at the defect location. In this section, we will develop a method of estimating CD impact of reticle defects for poly, metal and via layers only. Phase defects are not considered since defect data from a commercial maskshop suggests they are rare. We use a square approximation for defects in our analysis (as do most critical analysis methods).

In this section, we denote the size of a square defect by a and the minimum detectable defect size by a_{min}. W_{min} and S_{min} are the width and spacing design rules (DR) of the corresponding layer, respectively [3]. Df_{min}^{CD} and Df_{min}^{Con} are the minimum size CD and contamination defects that really matters for design functionality.

1) Poly: Poly layer printing typically uses bright-field masks with positive photoresist. Their impact is illustrated in Figure 5 and described in Table I.

Our evaluation of minimum defect size must take timing criticality

[3]For simplicity and pessimism we use minimum DR rules instead of using exact design values.

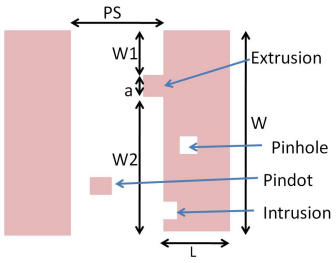

Fig. 5. Impact of various defect types on poly reticle

TABLE II
IMPACT OF DIFFERENT DEFECT TYPES IN METAL LAYER

Type	Wire Width	Design Impact
Intrusion	Decrease	Open
Extrusion	Increase	Short
Pinhole	Decrease	Resistance Change
Pindot	No change	None

of different cells into account apart from the possibliliy of open or shorts. For each transistor, we find the timing slack of the corresponding cell from the design timing report. Assuming that at most K defects lies on a critical path [4], we can evaluate the minimum defect size that changes the delay of the transistor and hence the path delay by less than T_{slack}/K. The formula for deriving the maximum gate length deviation, $\delta L_{critical}$ is derived in Equation (1). using basic transistor model and first order approximations where the various parameters are shown in Figure 5.

$$\frac{W}{L}_{new} = \frac{W1}{L} + \frac{W2}{L} + \frac{a}{L-a}$$
$$\frac{\triangle Delay}{Delay_{nom}} = -\frac{\triangle \frac{W}{L}}{\frac{W}{L}} = \frac{a^2}{WL}$$
$$\delta L_{critical} = a_{critical} = \sqrt{\frac{(T_{slack} - \alpha_{cycle})/K}{Delay_{nom}} WL}$$
$$= a_{min} \quad if \quad T_{slack} < \alpha_{cycle}$$

where α_{cycle} is taken as 1% of the design cycle time. The delay margin for each transistor is chosen to guardband against later variations. Hence, transistors on extremely critical paths which have slack less than 1% cycle time are assigned the minimum detectable defect size

To guardband against process variations downstream we set the minimum defect size as 20% the width (opens) and spacing (shorts) dimensions. We assume that pinholes do not have any parametric impact and can only cause an open if they are bigger than the gate length.

$$Df^{CD}_{min} = \frac{min(0.2W_{min}, 0.2S_{min}, \delta L_{critical})}{MEEF}$$
$$Df^{Con}_{min} = \frac{0.2W_{min}}{MEEF} \quad (1)$$

2) Metal Layer: Bright-field masks with negative resist are typically used to make trenches for depositing copper (dual damascene process). The impact of various types of defects is shown in Table II.

Dummy fill do not have any design impact and can be assigned a large Df^{CD}_{min} and Df^{Con}_{min}. Delay impact of metal layer mask defects

[4]A critical path typically consists of only 20-30 transistors and hence the area occupied by a critical path is very small compared to the area of the chip.Therefore $K > 1$

TABLE III
IMPACT OF DIFFERENT DEFECT TYPES IN VIA LAYER

Type	Via Width	Design Impact
Intrusion	Increase	Short
Extrusion	Decrease	Open/Resistance Increase
Pinhole	None	Metal Short
Pindot	Decrease	Resistance Increase

is negligible. Hence for a metal layer shape, Df^{CD}_{min} and Df^{Con}_{min} can be evaluated as follows:

$$Df^{CD}_{min} = \frac{0.2min(W_{min}, S_{min})}{MEEF}$$
$$Df^{Con}_{min} = \frac{0.2S_{min}}{MEEF} \quad (2)$$

3) Via Layer: Dark-field masks with positive resist are typically used to print via layer. Impact of various defect types on via layer is summarised in Table III and shown in Figure 6.

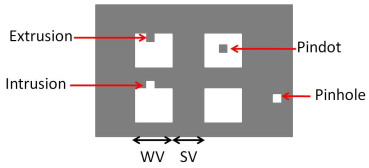

Fig. 6. Impact of different defect types on Via Reticle

Note that regions where the non-fill shapes on adjacent metal layers overlap must be assigned minimum detectable defect size of the inspection tool for contamination defects since even the smallest pinhole defect could cause a short. 20% change in via area is taken as the constraint to assign defect size for CD and contamination (pindot) defects. Redundant vias will have a larger Df^{CD}_{min} and Df^{Cont}_{min}. We can write the minimum size defects Df_{min} for a set of mXn redundant vias ($m = 1, n = 1$ for single via) as follows:

$$Df^{CD}_{min} = \frac{0.2max(m,n)min(W_{min}, S_{min})}{MEEF}$$
$$Df^{Cont}_{min} = \frac{0.2max(m,n)W_{min}}{MEEF}$$
$$Df^{Cont}_{min} = a_{min} \quad for \quad metal \quad intersect \quad regions \quad (3)$$

Note that since the value of MEEF varies considerably with optical process paramaters, we must take the largest value across the focus-exposure process window in these calculations. *Current inspection tools support adaptive thresholding where the threshold value is dynamically changed by the tool depending on online MEEF estimation [8]. Hence, we take MEEF=1 in our experiments instead of relying on a lithography simulator to compute it.*

IV. PARTITIONING

A. Problem Formulation

Given the minimum size defect for each feature on a reticle, partition the reticle such that each partition has length and breadth greater than a predefined value and is assigned a pixel size and sensitivity (CD and contamination) such that no critical defects are missed and the number of false+nuisance defects reported by the inspection tool are minimized.

In order to solve this problem we need to find the minimum detectable defect size as a function of sensitivity and pixel size. The resolution of any digital imaging system scales linearly with pixel size. Also, increasing the sensitivity helps in detecting smaller

features. Hence, for an inspection with pixel size, p and sensitivity s, we shall model this as shown in Equation (5) for both CD and contamination defects. Current inspection tools are capable of inspecting a $20nm$ defect (on the mask) which corresponds to $5nm$ on the wafer (MEEF=1) at a pixel size of $55nm$ and sensitivity of 100 [8]. Hence $K_c \approx 9$ [5].

$$D_{min} = K_c \frac{p}{s} \tag{4}$$

The total number of false defects (noise) reported by the inspection tool is a function of pixel size and sensitivity. Largest component of image noise, photon noise depends linearly on the density of pixels on the sensor and is hence, inversely proportional to the square of pixel size. To model other noise sources as well, we assume that noise is proportional to $p^{-\alpha}$, where p is the pixel size. Since, false defects are essentially noise, we can model light intensity falling on a pixel as a gaussian function, and hence the number of false defects as a function of threshold (inverse of sensitivity) is a complementary error function [2]. The model for false defects as a funcion of pixel size and two sensitivity levels (S^{CD} and S^{Con}) is shown in Equation (6). K^{CD}, K^{Con}, σ^{CD}, σ^{Con} and α depend on the inspection tool. We used commercial maskshop's inspection data from over 800 reticles with inspection area ranging from $8000 - 15000mm^2$, pixel size ranging from $72-250nm$, and sensitivities ranging from $75-100$ to fit these parameters and got $\alpha = 2.35$, $K^{CD} = 489.38$, $K^{Con} = 489.38$, $\sigma^{CD} = 0.01$ and $sigma^{Con} = 0.01$ if the area is taken in mm^2 and pixel size in nm.

$$\begin{aligned} FD = \frac{A}{p^{\alpha}} (K^{CD} erfc\left(\frac{1}{\sigma^{CD} S^{CD}}\right) \\ + K^{Con} erfc\left(\frac{1}{\sigma^{Con} S^{Con}}\right)) \end{aligned} \tag{5}$$

Number of nuisance defects depends on the design and the total number of real defects, which are the non-nuisance defects . Assuming that the defect distribution for a reticle follows the same negative binomial distribution as wafer defects, we can derive a model for the total number of real defects for a reticle of area A, inspected with pixel size p and a single sensitivity s using the following equation.

$$\begin{aligned} RD &= \sum_{DefectTypes} \int_{D_{min}}^{\infty} \frac{K_2}{D^{\alpha}} dD \\ &= \sum_{DefectTypes} \frac{K_2}{\alpha - 1} \left(K_c \frac{p}{s}\right)^{\alpha - 1} \\ &= T^{CD} \left(\frac{p}{S^{CD}}\right)^{\beta^{CD} - 1} \\ &+ T^{Con} \left(\frac{p}{S^{Con}}\right)^{\beta^{Con} - 1} \end{aligned} \tag{6}$$

where the constants were fitted using the same maskshop data described above to obtain $T^{CD} = 0.0002555$, $\beta^{CD} = 1.3$, $T^{Con} = 0.00008208$ and $\beta^{Con} = 0.88$.

Current inspection tools can take DNIR (Do Not Inspect Regions) as inputs. DNIR rules specify that a DNIR region can be as small as one pixel but there is forty pixel band in each direction that is not inspected. For our partitioning problem this essentially means that a partition must have dimentions of at least 80 pixels (recall that multiple pixel sizes are implemented as multiple scans with DNIRs). For simplicity we assume the same partition for both pixel size and sensitivity and use the largest pixel size in our experiments to define minimum dimension of a partition. Based on the above discussion, our problem can be stated more precisely as:

[5] K_c is slightly different for CD and contamination types of defects but we assume a constant value for simplicity.

Given a reticle with minimum size defect for each feature, create a partition with rectangular blocks each of width W_j and height H_j assigning a pixel size p_j, and sensitivities S^{CD}, S^{Con} such that the following function is minimized:

$$F = FalseDefects + \gamma TotalDefects \tag{7}$$

and the following constraints are obeyed:

- *Minimum dimension constraint:*

$$\min(W_j, H_j) > L_{min} \tag{8}$$

- *For any feature with min. size defect D_{CD} and D_{Con} lying in the j^{th} block of the partition:*

$$D^{CD} > K_c \frac{p_j}{S^{CD}}, \quad D^{Con} > K_c \frac{p_j}{S^{Con}} \tag{9}$$

where γ is a weighting factor that we can choose and L_{min} is the minimum dimension constraint.

B. Solution

We use a scan-line based approach which consists of alternatively moving horizontal and vertical lines across the reticle and placing them at the location which minimizes the cost function [6]. This procedure is done for a finite number of iterations. The number of iterations is chosen such that increasing them does not have a significant improvement in the cost function. The minimum unit by which the scan-lines are moved is equal to the minimum dimension constraint for the partitions ensuring that Equation (8) is obeyed by construction. The change in cost if a line is introduced at a particular position is calculated incrementally to improve runtime.For a particular position of a new scan-line, we only need to look at the neighboring lines in each direction and calculate the cost function for that region. Figure 7 gives an illustration of this idea. The current partitioning has blocks P1-9. When a new vertical scan-line is introduced, P2, P5 and P8 are split into two blocks and the improvement in cost function can be evaluated as shown below.

$$\begin{aligned} \delta Cost = Cost(P2A) + Cost(P2B) - Cost(P2) \\ + Cost(P5A) + Cost(P5B) - Cost(P5) \\ + Cost(P8A) + Cost(P8B) - Cost(P8) \end{aligned} \tag{10}$$

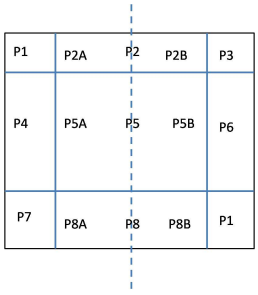

Fig. 7. Incremental Cost Evaluation for Partitioning

The cost of any single partition block can be computed by finding the minimum size defect inside the block, finding sensitivity for each pixel size (only a small number of discrete pixel sizes are available in the inspection tool) to minimize Equation (10).

The runtime for this method is $O(W_{chip} N_{iter} / L_{min})$, where W_{chip} is chip size, L_{min} is the moving distance of the scan-line and N_{iter} is the number of iterations.

[6] This method bears some resemblance to the DNIR placement algorithm proposed in [7]

TABLE IV
SHAPE SIMPLIFICATION RESULTS

Design Name	# Gates	Area (um^2)	# Rectangles (before)	# Rectangles (after)
AesCipher (8-Metal)	15467	102494	2512023	1012226
Mips (6-Metal)	11577	59461	2876871	1721828
Nova (6-Metal)	43156	268594	13243201	8041773

TABLE V
EXPERIMENTAL RESULTS FOR REDUNDANCY FINDING

Design Name	# Double Vias	# Dummy Fill	Runtime (min.)	Memory Usage (MB)
AesCipher (8-Metal)	131464	97772	8	910
Mips (6-Metal)	44004	67341	5	1190
Nova (6-Metal)	209623	303792	79	4814

TABLE VI
LAYER BY LAYER NON-CRITICAL REGIONS

Design	Via Layer	# Vias	% Redundant	Metal Layer	% Dummy Area
Mips	Via1	71724	23	Metal1	3.6
	Via2	72467	78	Metal2	6.4
	Via3	29970	65	Metal3	8.7
	Via4	12642	36	Metal4	10.9
	Via5	4850	46	Metal5	12.9
				Metal6	20.5
Nova	Via1	266215	22	Metal1	1.8
	Via2	324409	80	Metal2	4.5
	Via3	125926	75	Metal3	5.5
	Via4	37474	49	Metal4	9.9
	Via5	10992	64	Metal5	15.9
				Metal6	25.4

V. EXPERIMENTAL RESULTS

A. Non-functional Feature Finding

We implement our neighborhood graph based algorithm to identify redundant vias and dummy fill in C++ using OpenAccess (OA) API [1]. Layouts of some benchmark circuits implemented in $45nm$ Nangate OpenCell library along with insertion of double vias and dummy fill was done in Cadence Encounter [3]. OPC was performed on the generated GDSII files using Mentor Calibre [4].

The size of the post-OPC benchmark circuits that we considered along with improvement in rectangle count due to shape simplification are shown in Table IV. The threshold for bucketing was taken as $20nm$, which is less than the minimum metal width for $45nm$ Nangate Design. Around 50% reduction in number of shapes is observed. Table V summarizes the results of redundancy finding. Runtime ranges from 5 minutes to 1 hour 19 minutes for the designs considered. The number of redundant vias and dummy fill reported by our approach are verified with the number obtained from DEF file of the corresponding design. Double vias are reported with almost 100% accuracy by our approach and there is less than 1% error in dummy fill due to some outliers. The runtime of this algorithm can be improved by partitioning layout into smaller blocks and using a separate graph for each region. The algorithm can also be parallelized easily by running the critical graph construction step for each set M_iV_j in parallel. These techniques are left for future work.

Table VI shows the percentage non-critical regions for two benchmarks which indicates the potential benefits that can be derived from design-aware inspection of metal and via layers. For metal layers, dummy area is reported as a percentage of the total die area. Higher metal layers typically have less congestion after routing and hence have a greater percentage of dummy area.

B. Criticality Assignment & Reticle Partitioning

For assigning minimum size defect to each layout feature, we use the design rules from Free PDK [5]. Timing analysis was done on the post-routed design using Cadence Encounter [3]. Using this criticality assignment, reticle partitioning was implemented in C++. From the fitting results of false and nuisance defects, it is clear that false defects are typically 10-20X the nuisance defects. But nuisance defects are more important to maskshops as they help improve first pass yield. Hence, we took $\gamma = 10$ for our cost function in these experiments. Only two pixel sizes, $72nm$ and $90nm$, were used in our experiments. The minimum dimension constraint was taken as $2um$, which is slightly larger than dimension of 80 pixels at $90nm$ pixel size. The number of iterations for the scan-line was taken as 500.

Table VII shows the false and total defect count values evaluated on the basis of our fitted equations after partitioning compared to the case where inspection is done at a single value of pixel size and CD and contamination sensitivities for the entire reticle. Experiments were done for all metal, via and poly layers on two designs, for which the non-functional features have also been reported. Note that the reduction in real defects (non-false defects reported by inspection tool) is due to the decrease in nuisance defects since the partitioning problem is constrained to not miss any critical defect. Since the designs we consider are very small compared to real reticle sizes, we scaled up the values by 1000X to obtain defect count comparable to realistic size reticles. Note that up to 4X improvement in both false and nuisance defects is observed for higher via layers. The initial value of false and real defect count for Via1-3 and Via4-5 are the same due to similar minimum width design rule. Via layer show the most reduction in defect count due to a large number of redundant vias, some of which are arrays with more than 10 vias. The improvement is smaller in lower via layers due to metal intersect regions that need to be inspected at high resolution for pinhole defects as discussed in Section III(B). Since lower metal layers are dense, such regions occupy a large fraction of reticle area. The improvement in poly layers is not very substantial because in the designs we used a large fraction of cells had very low slack and hence inspection resolution could not be lowered in most regions of the reticle. Metal layers, due to their larger sizes, have the smallest number of false and nuisance defects initially and the improvement is due to dummy regions. *Note that the metal/via layer processing does not require any explicit timing information while poly layer leverages it heavily.*

For evaluating the first pass yield improvement due to design-aware inspection, we implemented a Monte Carlo simulation for both Mips and Nova where real defects were randomly placed on all the reticle corresponding to metal, via and poly layers. Minimum defect size placed was $7nm$ which is the smallest detectable feature size at $72nm$ pixel size inspection. Defect size distribution was taken as K/r^3 for a defect of size r, where K is found by taking the maximum defect size of $150nm$. Spatial distribution of the defects was uniform. If a reticle were inspected in a design-unaware fashion, *no* reticle will pass without repair/replacement. With this setup, we iterated over 10000 reticles and we find number of reticles R which do not report any defect with the design-aware inspection. This gives the first pass yield as $R/10000$. Results for the average first pass yield of the two designs for each reticle is shown in Figure 8.

TABLE VII
EXPERIMENTAL RESULTS FOR PARTITIONING

Design Name	Layer	Before		After	
		# False	# Real	# False	# Real
Mips	Poly	14.35	1.63	4.90	1.11
	Via1	15.23	1.43	10.46	1.10
	Via2	14.97	1.35	9.79	0.93
	Via3	14.97	1.35	8.38	0.827
	Via4	14.70	0.88	6.22	0.42
	Via5	14.70	0.88	2.91	0.21
	Metal1	1.06	1.20	0.98	1.11
	Metal2	0.52	1.10	0.44	0.92
	Metal3	0.52	1.10	0.39	0.82
	Metal4	0	0.49	0	0.36
	Metal5	0	0.49	0	0.33
	Metal6	0	0.49	0	0.23
Nova	Poly	69.32	9.04	32.44	6.63
	Via1	137.5	11.3	54.5	5.3
	Via2	67.51	6.10	51.14	4.97
	Via3	67.51	6.10	37.93	4.00
	Via4	66.34	3.99	16.1	1.32
	Via5	66.34	3.99	4.09	0.43
	Metal1	4.79	5.42	4.56	5.17
	Metal2	2.36	4.97	2.07	4.36
	Metal3	2.36	4.97	2.01	4.23
	Metal4	0	2.21	0	1.77
	Metal5	0	2.21	0	1.37
	Metal6	0	2.21	0	0.65

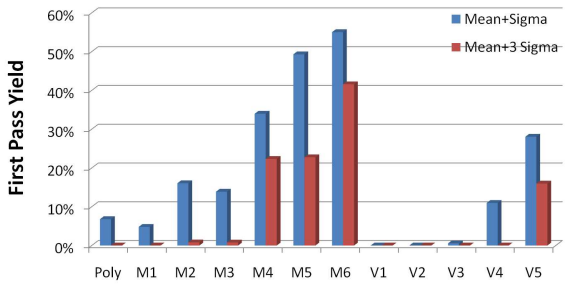

Fig. 8. First Pass Yield with design-aware inspection. The number of defects sprinkled are $\mu + \sigma$ and $\mu + 3\sigma$ derived from the reticle inspection statistics we have available from a commercial maskshop. Note that the yield is zero for the conventional design-unaware inspection strategy in all these cases.

VI. CONCLUSION

In this work we developed a comprehensive design-aware mask inspection flow:

- Proposed and implemented a graph based algorithm that finds non-functional features (dummy fill and redundant vias) in a post-OPC layout with almost 100% accuracy.
- We formulated a method to assign a minimum size defect to each feature of a reticle.
- We developed a scan-line partitioning algorithm to inspect different regions of the layout with different pixel size and sensitivity and up to 4X reduction in nuisance and false defects was observed along with up to 55% improvement in first pass yield coming from reduction in nuisance defects.

One key assumption in this work is that maskshops have complete mask set data for a design which is unrealistic for merchant maskshops. In the future, we shall explore methods to derive non-functional features and assign criticality when only a few layers are available. We shall also study the tradeoffs of tuning only sensitivity versus tuning both pixel size and sensitivity during inspection. Use of larger pixel sizes will also be explored. We will also explore a multi-level scan-line approach to speed up partitioning. Finally, we plan to test our approach in an actual commercial maskshop.

ACKNOWLEDGEMENT

The authors would like to acknowledge the generous support of IMPACT UC Discovery Grant and NSF CAREER award number 0846196. The authors would also like to thank Prof. Kahng (UCSD) and Dr. Stan Stokowski (KLA-Tencor) for their valuable inputs and contributions.

REFERENCES

[1] Openaccess API. http://www.si2.org/.
[2] Personal Communication with Dr. Stan Stokowski, KLA-Tencor.
[3] Cadence SOC Encounter. http://www.cadence.com/, 2008.
[4] Mentor Calibre. http://www.mentor.com/, 2008.
[5] FreePDK 45nm Design Rules. http://www.eda.ncsu.edu/wiki/FreePDK45:Contents, 2009.
[6] S. Banerjee, P. Elakkumanan, L. Liebmann, and M. Orshansky. Electrically driven optical proximity correction based on linear programming. In *ICCAD 2008*, pages 473–479. IEEE Press, 2008.
[7] K. Chakraborty, A. Lvov, and M. Mukherjee. Novel algorithms for placement of rectangular covers for mask inspection in advanced lithography and other VLSI design applications. *IEEE TCAD*, 25(1):79 – 91, Jan. 2006.
[8] A. Dayal, B. Mu, V. Iyer, P. Lim, A. Goonesekera, and B. Broadbent. Results from the KLA-Tencor TerascanXR reticle inspection tool. volume 7122, page 71223G. SPIE, 2008.
[9] F. Driessen, J. Gunawerdana, Y. Saito, H. Tsuchiya, and Y. Tsuji. Flexible sensitivity inspection with TK-CMI software for criticality-awareness. volume 7122, page 712222. SPIE, 2008.
[10] A. Dürr, A. Zibold, and K. Böhm. An advanced study for defect disposition through 193-nm aerial imaging. volume 6152, page 61522M. SPIE, 2006.
[11] K.D. Gourley and D.M. Green. A Polygon-to-Rectangle Conversion Algorithm. *Computer Graphics and Applications, IEEE*, 3(1):31 –36, jan. 1983.
[12] P. Gupta, A. B. Kahng, D. Sylvester, and J. Yang. A cost-driven lithographic correction methodology based on off-the-shelf sizing tools. In *Proc. DAC*, pages 16–21, 2003.
[13] P. Gupta, A. B. Kahng, D. Sylvester, and J. Yang. Performance driven OPC for mask cost reduction. In *ISQED 2005*, pages 270–275, 2005.
[14] S. Hedges, C. Le, M. Eickhoff, M. Wylie, T. Simmons, V. Vellanki, and J. McMurran. Novel mask inspection flow using sensitivity control layers (SCL) on the Terascanhr-587 platform. volume 7122, page 71221G. SPIE, 2008.
[15] K. Hosono and K. Kato. PMJ panel discussion overview on mask complexities, cost, and cycle time in 32-nm system LSI generation: conflict or concurrent? volume 7122, page 712207. SPIE, 2008.
[16] T.-Y. Kang, H.-C. Lee, H. Zhang, K. Yamada, Y. Kitayama, K. Kobayashi, and Peter Fiekowsky. Auto-classification and simulation of mask defects using SEM and CAD images. volume 7122, page 71221F. SPIE, 2008.
[17] L. Karklin, M. Altamirano, L. Cai, K. Phan, and C. Spence. Automatic defect severity scoring for 193-nm reticle defect inspection. volume 4346, pages 898–906. SPIE, 2001.
[18] H. Moribe, T. Bashomatsu, K. Matsumura, A. Uehara, and H. Takahashi. Improvement of image quality and inspection speed in LM7500 reticle inspection system. volume 7028, page 70282K. SPIE, 2008.
[19] L. Pang, A. Lu, J. Chen, E. Guo, L. Cai, and J.-H. Chen. Enhanced dispositioning of reticle defects for advanced masks using virtual stepper with automated defect severity scoring. In *SPIE*, volume 5256, pages 461–473, December 2003.
[20] H.W. Six and D. Wood. The rectangle intersection problem revisited. *BIT Numerical Mathematics*, 20(4):426–433, 1980.
[21] D. Stoler, W. Ruch, W. Ma, S. Chakravarty, S. Liu, R. Morgan, J. Valadez, B. Moore, and J. Burns. Optimizing defect inspection strategy through the use of design-aware database control layers. In *SPIE*, volume 6730, October 2007.
[22] H. Tsuchiya, M. Tokita, T. Nomura, and T. Inoue. Die-to-database mask inspection with variable sensitivity. volume 7028, page 70282I. SPIE, 2008.
[23] W. W. Volk, C. Hess, W. Ruch, Z. Yu, W. Ma, L. Fisher, C. Vickery, and Z. M. Ma. Investigation of smart inspection of critical layer reticles using additional designer data to determine defect significance. In *SPIE*, volume 5256, pages 489–499, December 2003.