Targeted Layout Modifications for Semiconductor Yield/Reliability Enhancement

Gerard A. Allan

***Abstract*—A new layout modification tool for the automation of layout modifications to improve the yield and reliability of semiconductor IC layout is reported. The Peye tool combines a polygon library with the practical extraction and reporting language (Perl). This new tool permits complex layout modification operations to be defined using the powerful language features of Perl. The Peye tool has been interfaced with a sampling-based yield prediction system to enable the measurement of the layout modifications and yield predictions based on these modifications. This enables the usefulness of a modification to a particular design to be assessed by sampling before use. Both the sampled measurement and the final modifications to the whole chip database can be farmed out to a number of networked computers, enabling the system to assess and apply layout modifications to large industrial ICs in a reasonable time. The results of layout modifications are presented.**

***Index Terms*—Critical area, layout modification, redundant via, reliability, semiconductor yield, survey sampling, track displacement, wire spreading, yield modeling.**

I. Introduction

IT IS WELL known that IC layout can be made more robust to defects by making minor changes to the layout [1]–[4]. Such changes include adding extra vias where only a single via provides a connection between layers, moving wires so that they are further apart, and thickening thin wires. Other changes that can be made where there are manufacturability issues, include increasing the overlap around vias [5] and lengthening polysilicon transistor gate extensions. Such changes can have a measurable impact on the yield of a chip [5] and can help increase reliability [6].

Design rule checker (DRC) tools can be used to perform such functions, but these tools have not been primarily designed for this task and can lack some useful geometry operations or perform these operations inefficiently. Another important difference is that these tools do not have "nice" language features (control structures, subroutines, scoped variables, built-in data structures, symbolic references, objects, packages) that enable the design and reuse of maintainable code.

This paper presents a layout geometry manipulation tool Peye (practical extraction and reporting language (Perl) eye) that combines the polygon library originally developed for yield estimator, eye [7], with Perl. The eye library was primarily designed to be used in the prediction of IC yield and for IC layout manipulation and has a number of built-in operations designed specifically for these tasks. New algorithms have been added to address specific layout modification tasks. Perl is scripting language that includes a large range of language features designed to make the job of programming easy. As a result, Perl is widely used in a number of industries, including IC design, where it is often used as a glue between various design tools and in IC fabrication, where it is used to extract information from various automatically generated data files and in data base operations.

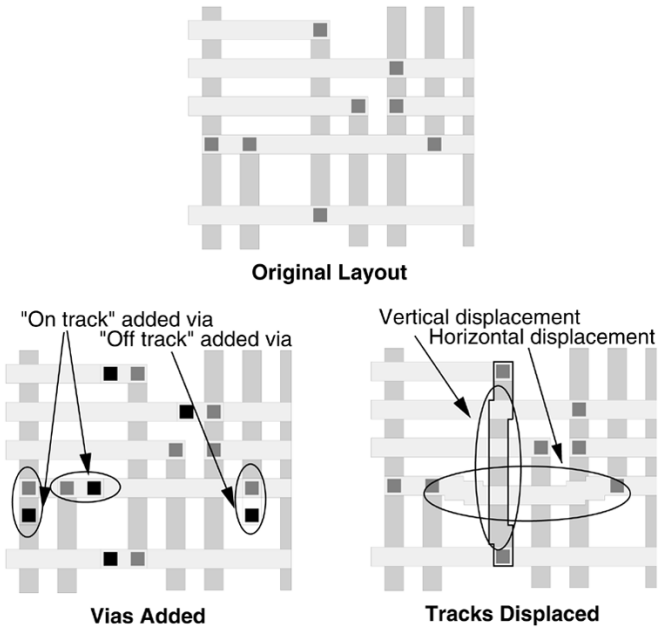

Fig. 1. Via additions and track displacement (wire spreading) of the original layout.

Perl provides the ability to link external C/C++ libraries and routines so that they can be used within a Perl program. The combination of Perl and the eye library creates a powerful new tool that enables geometrical operations to be combined and implemented as Perl subroutines. These, in turn, can be combined to implement complex layout modification functions. Within Peye, the layout modification functions themselves can be made largely design rule independent, for similar technologies, using data structures containing the design rules (in Perl hashes or arrays) that are passed to functions. This permits libraries of routines to be built that can be used with a range of technologies and design rule sets.

II. Peye Tool

The Peye tool enables the input of GDSII/CIF mask data to a Perl program as a set of layer objects. These layer objects can

Manuscript received November 1, 2002; revised March 10, 2004. This paper was presented in part at the 13th Annual IEEE/SEMIAdvanced Semiconductor Manufacturing Conference and Workshop (ASMC 2002).

The author is with Predictions Software Ltd., EH9 3JL Edinburgh, U.K. (e-mail Gerard.Allan@icyield.com).

Digital Object Identifier 10.1109/TSM.2004.835727

```
%layer_table= ("L1", active,# gds layer 1 to be called $active
               "L2", poly,  # gds layer 2 to be called $poly
               "L3", metal1); # etc.,

# load the defined layers from the file (compressed by gzip)
load_layers("myfile.gds.gz",\%layer_table);

$trangate=And($poly,$active); # generate a new layer

if(defined $metal2) { # sometimes we load a $metal2 as well
   $metal2->{color}="Blue"; # ps plot color
   push(@plotlist,$metal2); # add this to the plotlist
}

$trangate->{color}="Red";# ps plot color
push(@plotlist,$trangate);# add to list

plotps("file.ps",\@plotlist); # plot to file.ps
```

Fig. 2. Small Peye script showing use of Perl features.

be operated on using a range of Boolean, sizing, filter, intersection, and move operations to generate new layers. These new layers can be output as either a postscript file for viewing or as a GDSII/CIF format file.

A simple program is shown below, which reads in a number of layers from a GDSII file and generates a single new layer which is plotted along with the layer metal2, if the layer metal2 exists (not in this case). This small program shows how Perl hashes (%layertable), lists (@plotlist), and control structures are used to enable the construction of flexible programs.

Peye permits the generation of complex functions that can be used to create and modify IC layout layers. An example of these functions is given in Fig. 3. The displacement functions shown in Fig. 3 are part of the implementation of horizontal and vertical wire spreading or track displacement (shown in Fig. 1). The displace_layer function generates two new layers that define the cuts to the existing track and the new material to add, in order to generate the displaced tracks. The modifications are applied to the track layer within the Peye program to enable further operations on this layer. The two layers that define the differences are saved for later use (mask output/plotting). In general, differences to modified layers are saved rather than the changed layer itself. This permits the application of modifications to selected areas of the design rather than the whole design. These changes can be optionally applied when masks are generated simply using Boolean operations (or, andnot) or when additional layout processing is undertaken. Furthermore, having the changes immediately available makes it easier to determine if changes have been made and the amount and extent of these changes. Another reason is that the differences generally require significantly less file space than a new modified layer. This can be an important consideration as the modifications are normally applied to a flat layout (the majority of changes are applied to routing) which would result in large amounts of layout if the whole of each modified layer was stored. Thus, the increase in GDSII size is directly related to the number of modifications that are made.

III. Measuring the Impact of Changes

In principle, measuring the impact on yield can be achieved using the standard yield prediction methods of critical area analysis [8]–[11], via counts, etc. The yield calculated for the original layout can be simply compared to the calculated yield of the modified layout. However, generating the new modified layout can be time consuming and the extraction of metrics for yield analysis, where the whole layout is analyzed, can also be time consuming. The preferred method is to use a sampled measurement of the layout [12], [13]. The same sampling procedure is used on both the original and modified layout, with exactly the same regions of the chip used as samples in both measurements. Only the samples selected are used to generate the yield prediction, and hence only those regions need to modified and analyzed to compare the yield of a modified chip layout. This means that only a small fraction of a chip layout need be modified, not the whole layout. The sampling method greatly reduces the time required to analyze the impact on a layout of a set of applied modifications. This is particularly useful where the set of modifications to be applied is undecided. A number of different sets of modifications can be applied and the results compared without the need to modify the whole layout database, permitting a wider analysis and optimization than might otherwise be possible. Alternatively, where a design is already in production, a search can be conducted for modifications that maximize the yield improvement and minimize the cost of producing new masks.

The Edinburgh Yield Estimator Sampling (EYES) [14], [15] is a software tool for generating critical areas and other metrics of IC layout. It is used in industry to estimate chip yields preproduction [16] and to compare end of line yields [17]. The EYES system has been modified to permit samples of IC layout to be preprocessed before being passed to the measurement routines. This enhanced EYES system is shown in Fig. 4. The generated samples are first passed to a Peye program which applies the required modifications, or optionally samples are farmed out to a number of networked hosts and processed there by a Peye

```
sub displace_layer_vert {
  displace_layer(@_, 0);
}

sub displace_layer_horz {
  displace_layer(@_, 1);
}

sub displace_layer {
  my $self=shift;    # "object" calling this sub
  my $layer1=shift;  # symbolic reference to layer
  my $pins=shift;    # anchor points -- "ends" of wire that are fixed
  my $horz=shift;    # horizontal (1) else vertical diplacement

  my $numdisplace=2; # number of  iterations to try

  my $generated=new eye::Layer();# used to accumulated layer additions
  my $removed=new eye::Layer();# used to accumulated layer subtractions

  my $ruleref = $self->ruleref();# design rules for this operation

  for(my $i = 0 ; $i < $numdisplace; $i++) {
    if($horz==1) { # perform track moves where possible
      ($thecut,$extralayer)=displacewire_horz($$layer1,$pins,$ruleref);
    } else {
      ($thecut,$extralayer)=displacewire_vert($$layer1,$pins,$ruleref);
    }
    last if(empty($thecut));#if no displacements done-- don't try again

    #update layer for other operations
    $$layer1=Andnot($$layer1,$thecut); # subtract cut material
    $$layer1=Or($$layer1,$extralayer); # added new material

    # now create/update layers to be saved -- difference from original

    # add new extra and  subtract any cut to give layout generated
    $generated=Andnot(Or($generated,$extralayer),$thecut);
    # add new cut and subtract any new extra to give layout removed
    $removed=Or(Andnot($removed,$generated),$thecut);

    print "displace horz $layer1 $i\n" if($message &&  $horz);
    print "displace vert $layer1 $i\n" if($message && !$horz);
  }
  $generated->{cifname}=$ruleref->{generated_name};# assign cif name
  $removed->{cifname}=$ruleref->{removed_name};

  $self->returnlayers([$generated,$removed]); # used later
}

sub displacewire_horz {
...
...
```

Fig. 3. Peye subroutines used in displacement of layout tracks (wire spreading). Generated layers are the differences required to implement the displacements using a layer to be added to, a layer to be subtracted from, and the original layer.

program. These modified samples are then measured and compared with the original layout sample results.

A similar process is used when generating mask data. The layout is divided into regions, rather than samples, with each region processed either on the same host or farmed out to another host. The Peye program/s generate mask data. The CIF[1] files generated on multiple hosts are combined to form a single GDSII file by a simple Perl program that concatenates the

[1]CIF is a simple text format that is easily manipulated.

TABLE I
VIA 1–4 STATISTICS FOR VIA ADDITIONS TO REDUCE THE NUMBER OF NONREDUNDANT VIAS

Redundant Via Additions				
Chip 1 Via Statistics				
	via 1	via 2	via 3	via 4
original total vias	3,278K+/- 88K	2,332K+/- 65K	2,747K +/- 129K	2,718K +/- 126K
original non redundant vias	1,135K +/- 30K	866K +/- 16K	328K +/- 9K	93K +/- 3K
peye added redundant vias	490K +/- 14K	550K +/- 10K	208K +/- 5K	53K +/- 2K
(%)	43%	64%	63%	57%
Chip 2 Via Statistics				
	via 1	via 2	via 3	via 4
original total vias	25,130K+/- 406K	16,912K+/- 257K	15,114K +/- 275K	3,960K +/- 267K
original non redundant vias	19,545K +/- 247K	13,091K +/- 180K	9,696K +/- 137K	174K +/- 70K
peye added redundant vias	167K +/- 14K	12,794K +/- 161K	6,390K +/- 82K	0K +/- 0K
(%)	1%	98%	66%	0%
Chip 3 Via Statistics				
	via 1	via 2	via 3	via 4
original total vias	3,228K+/- 70K	2,467K+/- 52K	1,499K +/- 57K	1,368K +/- 73K
original non redundant vias	1,264K +/- 19K	1,180K +/- 15K	432K +/- 10K	123K +/- 2K
peye added redundant vias	765K +/- 10K	820K +/- 11K	235K +/- 6K	80K +/- 2K
(%)	61%	69%	54%	65%
Chip 4 Via Statistics				
	via 1	via 2	via 3	via 4
original total vias	3,682K+/- 67K	3,164K+/- 65K	1,688K +/- 63K	1045K +/- 43K
original non redundant vias	1,820K +/- 26K	1,630K +/- 20K	615K +/- 9K	134K +/- 3K
peye added redundant vias	703K +/- 10K	893K +/- 13K	367K +/- 7K	56K +/- 2K
(%)	39%	55%	60%	42%

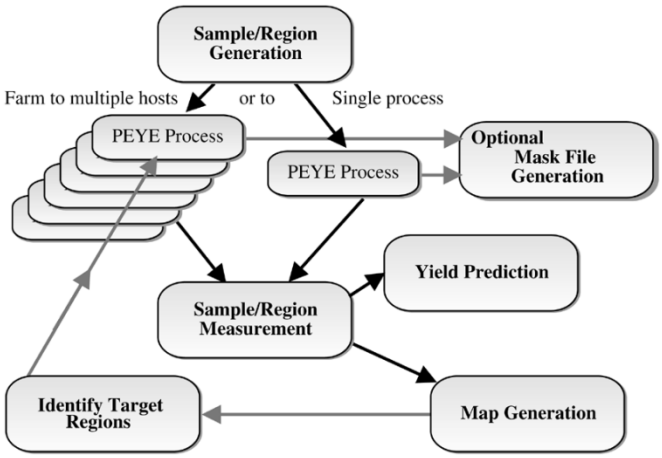

Fig. 4. EYES system diagram for farmed or single-process layout modification measurement and mask generation.

generated data onto the existing GDSII creating a new "root" cell that includes an instance of the original root cell. This ensures that the original design data can be recovered from the modified file.

A. Cost Benefit Analyses

The process of modifying a large design with the aim of improving its yield and/or reliability should only be undertaken where there is a benefit. One of the advantages of the sampling method using Peye is that it permits an estimate of the change in yield and reliability [18] before the process of modifying the complete database. After estimates of yield improvements have been obtained, it is possible calculate the impact of the yield improvement process taking into account the cost of processing the changes using Peye, the cost associated with LVS and DRC on the modified database, and possibly the cost on new masks (where a chip is already in production). A further consideration might also be any delay that the modification and checking process adds to manufacture of first silicon.

It is clear that for some designs that are only produced in low volume the gains will not be outweighted by the costs and inconvenience. However, for chips that are destined for large volume or are already in production in volume, the benefits of even a small increase in yield can represent considerable cost savings.

IV. LAYOUT MODIFICATIONS

There is a large range of layout modifications that can be used to increase chip yield and reliability. Many such modifications are minor. The yield/reliability improvement associated with such changes is often process and chip dependant. These changes can be associated with processing issues or random defects. For example, via overlap can be a problem where a design has been scaled and insufficient overlap is left around vias [5]. Presented below are layout changes primarily intended to reduce the susceptibility of layout to random defects, addition of vias, and track displacements.

A. Addition of Redundant Vias

Within an IC layout there are often many instances where an extra via or contact could be introduced to "partner" a single nonredundant via. This is particularly true of routing where many tracks have sufficient space available for an extra via. This extra via enables a single via failure to be tolerated. This assumes that via failure is an isolated random event that does not impact the partnering via. This is not always the case, for example, if the via is in a sparsely routed area other techniques

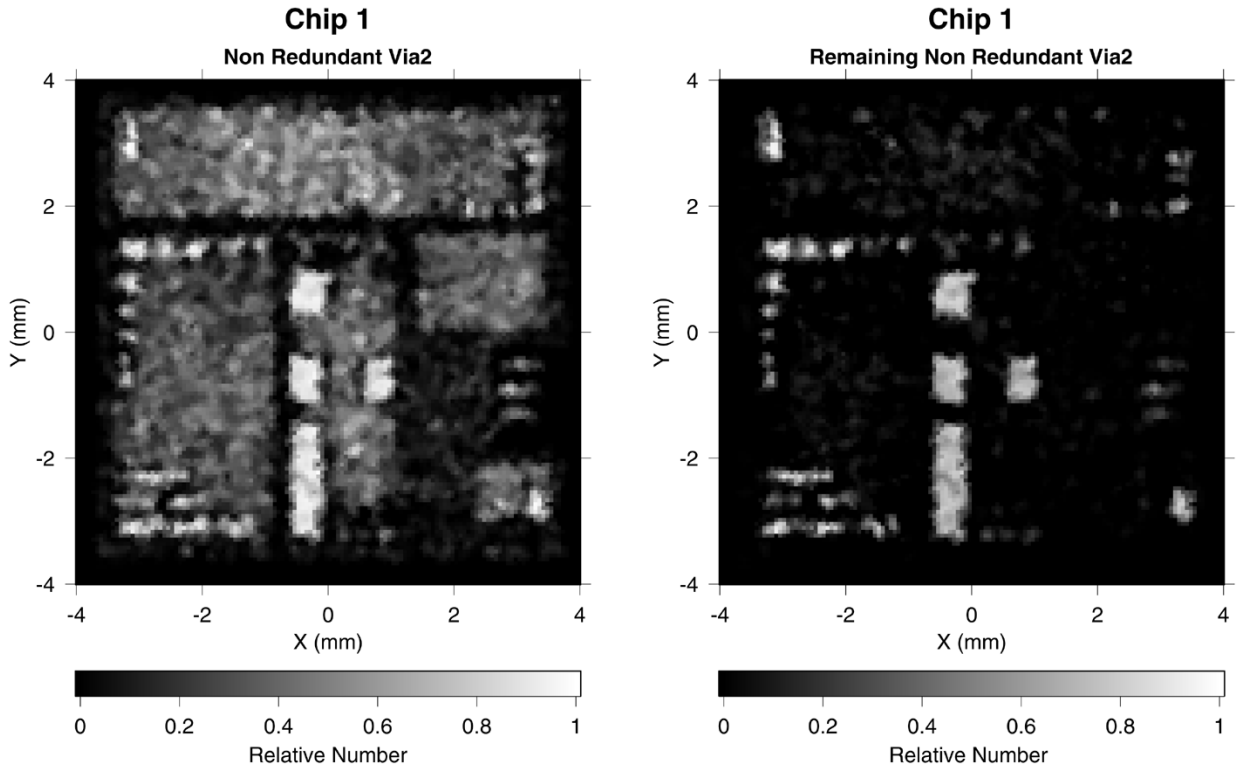

Fig. 5. Maps (generated by sampling) of original on redundant via2 and remaining nonredundant vias after via additions.

to increase via density may also be required. The process of adding extra vias by hand is very time consuming and for large complex designs an automated solution is desirable.

The process of partnering nonredundant vias uses the built-in Peye nonredundant function

$$\$\text{nrvia2} = \text{nonredundant}(\$\text{metal2}, \$\text{metal3}, \$\text{via2}).$$

This function takes three layers as arguments. The first two are the conductor layers and the third is the via layer. The nonredundant operation defines nonredundant vias as those that connect two polygons, one from each of the two conductor layers that are not connected by any other via. That is, it identifies those vias that join a unique pair of polygons, one from each layer. Once these nonredundant vias are identified a series of sizing, intersection, Boolean, and move operations are used to generate a new via and associated conductor layers [3].

The via addition procedure is structured so that vias can be restricted to be placed on only one of the conductor layers (e.g., metal2) so that only the other conductor layer (e.g., metal3) and the via layer are modified (assuming no extra overlap of the via is required). This restricted modification might be used where the cost of modifying more than two masks was not cost effective. The via placement can also be restricted to placement on either of the existing conductor layers (on track placement). On-track placement minimizes the addition of extra conductor material, which may be important if intralevel shorts are predicted to cause a high level of faults. The placement strategy which obtains the highest number of additional vias attempts to place vias on track first, and if this cannot be achieved it attempts to place the via off the existing metal tracks beside the existing via (off track). These via placement types are shown in Fig. 1.

1) Results: Four industrial designs have been analyzed. These use a 0.4-μm CMOS technology with chip sizes ranging from 0.6 to 3.4 cm^2. All contain varying amounts of standard cells, routing, and RAM. Results are presented for via additions in Table I generated by sampling.

The via addition results show that commonly more than 50% of nonredundant vias can be partnered with a new via. Chip 2 is different than the other three chips in that via1 and via4 are resistant to via additions. Chip2 is mostly a very dense regular design in which via1 is very dense with few opportunities for additional via1. Interestingly, the via2 level of the same design can have nearly all (98%) of the original nonredundant vias converted to redundant vias.

The sampling method permits the visualization of the amount and position of original nonredundant vias and results of adding vias. Fig. 5 shows two maps of Chip 1 vias, before and after via additions. These show that areas of dense via2 of Chip 1 are not impacted as much by additional vias as there is insufficient space to add many more vias.

B. Metal Displacements

Wire displacements can be used to reduce the critical area of a layout. That is, they can reduce the probability that a defect falling on the chip will cause a fault. This can be shown graphically using a fault probability map. Fault maps indicate regions of a layout which are most sensitive to defects. A map is generated by plotting critical area regions, colored with an intensity proportional to the number of defects, found in the fabrication process, of the size associated with the critical area. This size distribution is normally assumed to be of the form $(1/\text{Defect Size}^3)$ [19].

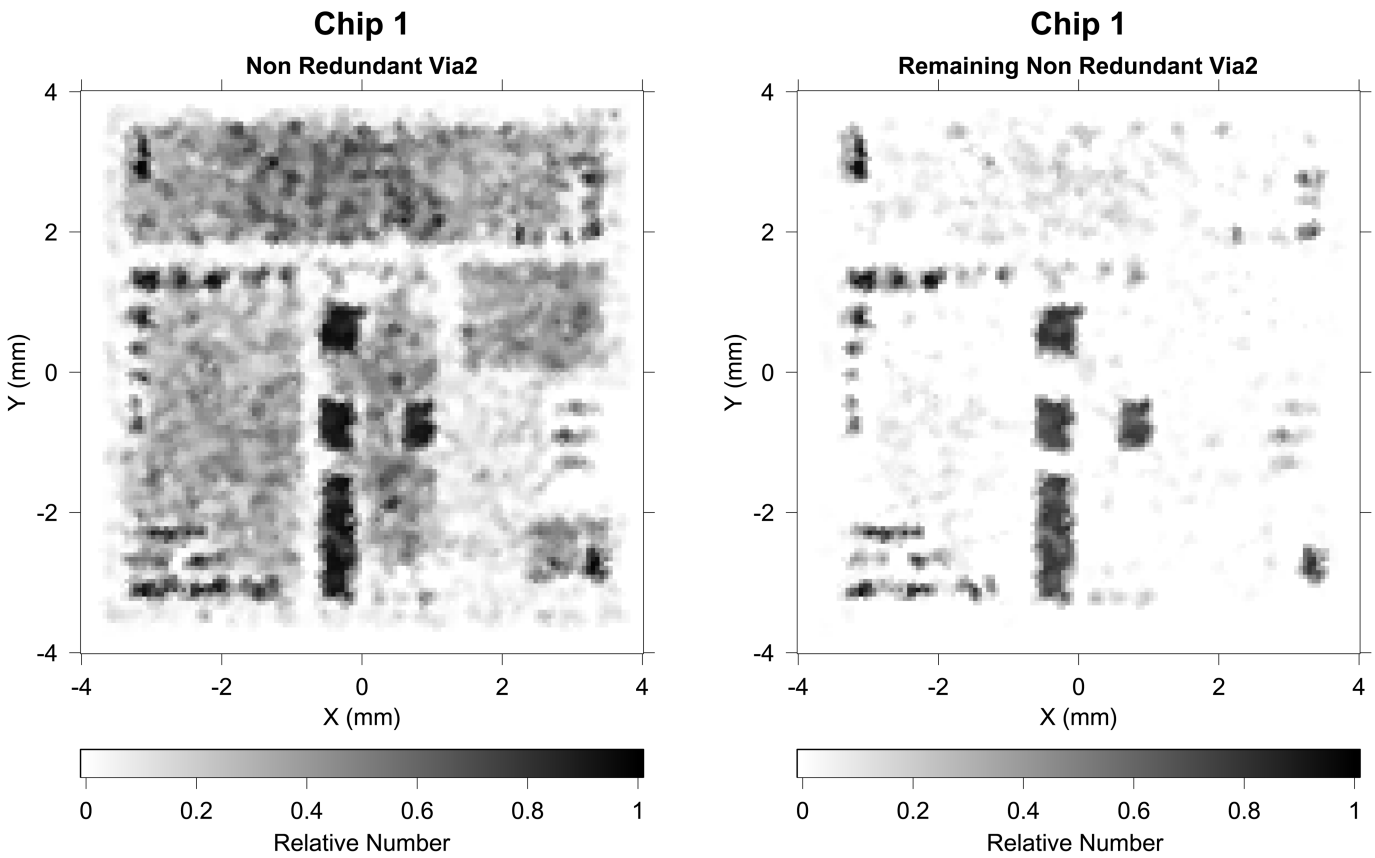

Fig. 6. Fault maps for an original layout and the same layout with a displaced track.

The fault maps shown in Fig. 6 show the change in critical area and hence the change in fault probability as a result of displacing metal tracks. The amount of "critical area" is reduced at the displacement site. This is due to the fact that smaller defects (darkest critical area) can no longer cause a fault at the displacement site and larger defects are less likely to cause a fault.

The displacement operation implemented in Peye is more complex than the via addition procedure. Displacements are applied by cutting sections from existing tracks and adding new metal geometry, slightly extended and displaced, to replace the removed track. The algorithm identifies suitable track lengths and implements the changes using a combination of intersection, Boolean, filtering, and move operations. The powerful language features of Perl make this implementation significantly easier than might otherwise be the case. The top level displacement functions are shown in Fig. 3.

1) Results: Again, the same four industrial designs have been analyzed for metal displacements using sampling. Fig. 7 shows results from Chip 1 (500 MB GDSII 0.4-μm CMOS technology industrial design). Displacements have been applied to the metal2–metal5. Metal1 was not modified because the design rules are more complicated. The layout was modified by displacing metal tracks away from neighboring tracks. Two iterations of this displacement technique were used, that is displaced tracks can themselves be displaced, as shown by the horizontal displacement in Fig. 1. The result for this chip shows a significant reduction in the probability of shorts (4%–12%) for the metal2–metal5 layers. These figures were obtained using a sampled extraction, using the same sample regions for both before and after the modifications.

TABLE II
METAL2–METAL5 REDUCTIONS IN FAULTS AS A RESULT OF MINOR METAL DISPLACEMENTS

Metal Displacement Fault Reduction				
Chip 1				
	Metal 2	Metal 3	Metal 4	Metal 5
Change	8.5%	12.0%	12.2%	4.2%
Reduction Weight	0.30	0.44	0.24	0.01
Chip 2				
	Metal 2	Metal 3	Metal 4	Metal 5
Change	0.0%	5.9%	0.1%	0.0%
Weighted Change	0.01	0.93	0.06	0.00
Chip 3				
	Metal 2	Metal 3	Metal 4	Metal 5
Change	7.2%	11.2%	13.3%	5.7%
Weighted Change	0.20	0.37	0.41	0.02
Chip 4				
	Metal 2	Metal 3	Metal 4	Metal 5
Change	4.9%	8.0%	6.7%	3.1%
Weighted Change	0.26	0.37	0.34	0.03

The results for all the chips analyzed are shown below. The percentage change in each of the fault mechanisms is given along with the reduction weight. This is the relative importance of the change with respect to the other metal layer faults. For example, in Chip 1 it is clear that metal3 has a higher overall number of faults. A 12% reduction in metal3 faults has a greater impact on the total number of faults than a 12.2% reduction in metal4 faults. In fact, the metal3 change represents 44% of the total fault reduction obtainable by displacing metal2–metal5.

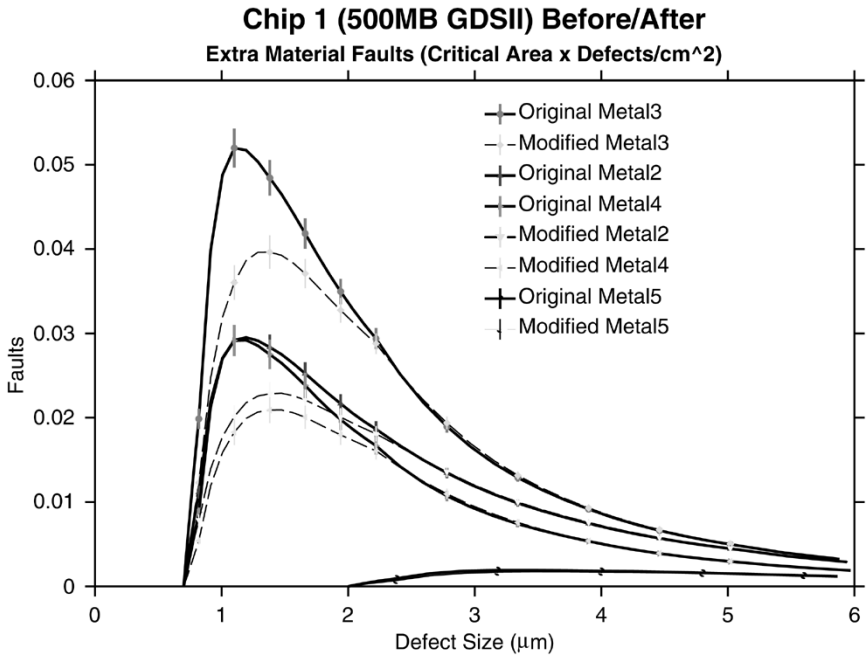

Fig. 7. Metal2–metal5 fault curves showing reductions in faults as a result of layout modifications—metal2 8.5%, metal3 12%, metal4 12.2%, metal5 4.2% (assuming a defect size distribution of the form $(1/\text{Defect Size}^3)$).

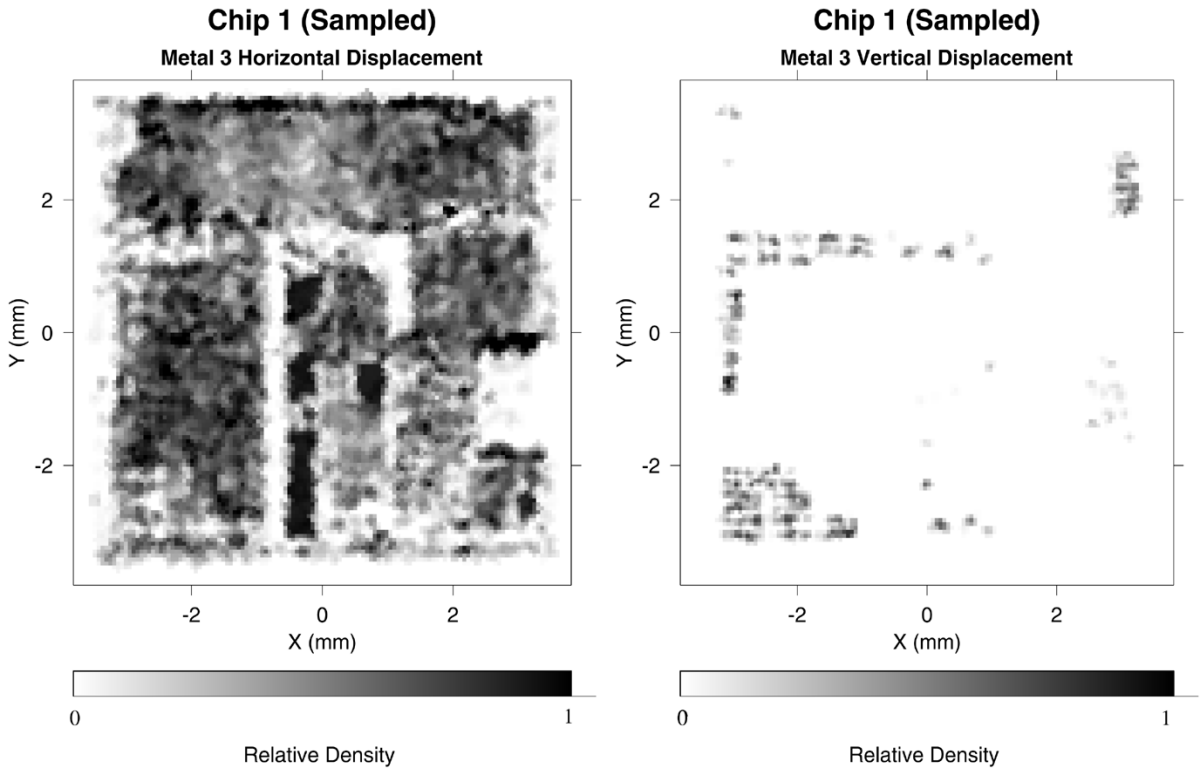

Fig. 8. Maps of sampled metal3 horizontal and vertical displacement modification.

The results given in Table II indicate that displacements are dependant on the chip layout. Chip 2 (which is large and dense) shows only a relatively small change to faults overall, with only metal3 impacted in a significant way; whereas, the other chips indicate "reasonable" changes in all metal layers. The weighted change shows that a good majority (75%) of the change effect can usually be obtained from displacement of one or two layers.

2) Displacement Maps: The sampling procedure generates a number of data points over the chip that can be plotted to visualize where changes have been made. Fig. 8 shows the metal3 results generated by sampling the layout of chip 1. These maps show where metal3 displacements horizontal and vertical have occurred. It can be seen that there are significantly fewer vertical displacements. This implies that most metal3 displacements are

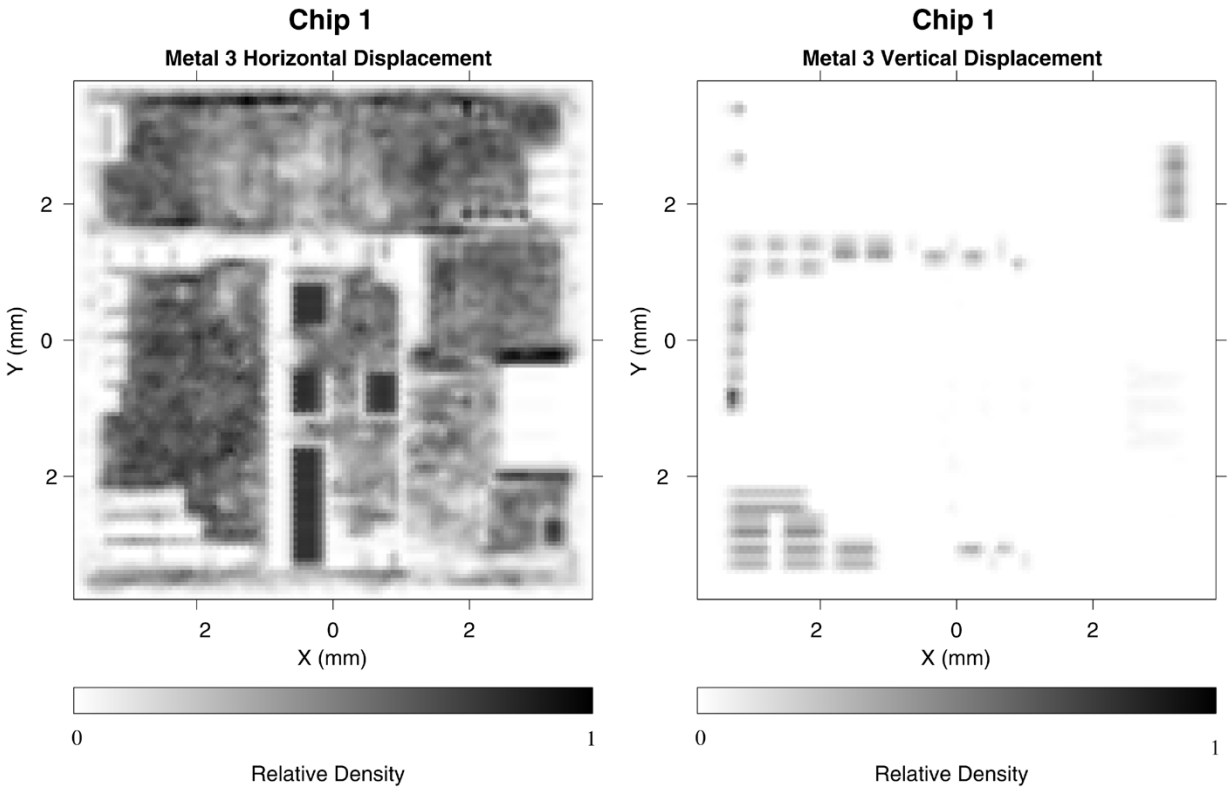

Fig. 9. Maps of metal3 horizontal and vertical displacement modification.

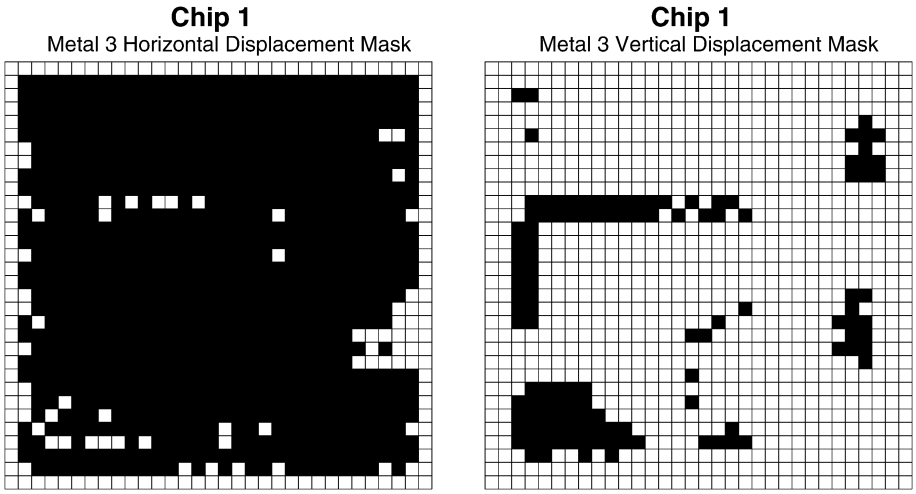

Fig. 10. Sections of chip 1 that are recommended for the application of metal3 horizontal and vertical displacement modifications.

horizontal and that that much of the computational effort to generate vertical displacements will be wasted.

A comparison of the sampled horizontal displacements measurements with the results for the whole chip (divided into sections), shown in Fig. 9, indicates good agreement for both the horizontal and vertical displacements.

V. Required Resources

The use of the sampling procedure makes the determination of which of the modifications can be usefully applied to a particular chip layout significantly less resource intensive than measuring the modifications applied to the whole layout. For the examples given here, all the metal displacements and via additions and associated measurements were carried out together in the same run. The time taken for Chips 1–4 was 2.3, 4.1, 3.2, and 2.9 CPU hours using a 750-MHz PIII Linux Laptop. The memory requirement is not large (say 128 MB); sampling does not require the whole layout to be present in memory.

The time and memory resources to apply the selected modifications to the whole layout of a large industrial design can be large. If layout modifications were to be applied to a whole chip in a single run, rather than sections, the memory requirement would be rather large (in the order of several gigabytes). The memory resource requirement can be reduced by processing the design in sections. Generation of mask data for the whole

of Chip 1 to implement metal2–metal5 displacements and the via2-via5 addition was performed on a network of Sun Ultra 5s (450 Mz) taking 8 h in real time (92 CPU h) to complete. The layout was automatically farmed out in sections (4096) to 15 of the lightest loaded in a farm of some 50 Sun Ultra 5s. The same job was completed in 32 CPU h on a 750-MHz PIII Laptop. As the layout is divided into sections, the total memory requirement is low (256 MB would normally be sufficient).

The efficiency of the process can be improved further by using the sampled information to direct determine which sections of layout should be processed. Fig. 10 shows the sections of layout that are considered, on the basis of a sampled measurement, to be suitable to for horizontal and vertical displacement modifications. For metal3 displacements on Chip 1 a reduction of 42% in the execution time was obtained by using the sample data to direct the modifications. This resulted in a 4% reduction in the amount of track displaced. Modifications on other chips have shown even larger reductions in computation time (up to 80%) with only minor reductions in the number of modifications.

VI. Conclusion

A new layout modification tool Peye has been reported. This tool combines IC layout functions, from the eye tool library, with Perl. This tool enables complex layout modification algorithms to be built with the aid of Perls wide range of language features. These algorithms can be implemented in a largely design rule independent way, enabling a library of layout modification procedures to be developed that are reusable and easily maintained.

The "best" layout modifications to apply to a chip layout are dependant on the existing chip layout and the fabrication process. The impact of a set of layout modifications can be determined by applying the modifications to samples of the chip layout, rather than the whole layout. The results from these samples can be used to define the modifications to be applied to the whole chip. Peye can be used with the EYES tool to generate sampled predictions of layout modifications. The sampled results have also been used to determine where within a chip modifications should be applied, reducing the amount of "wasted" computation.

The Peye tool has been used to implement complex layout modifications on large industrial designs in a timely manner by splitting the task of modification over a number of hosts and combining the results. Results presented suggest that the modifications implemented using Peye are able to significantly reduce the number of faults caused by shorting defects, by as much as 12% in some cases, and can often reduce the number of nonredundant vias by more than 50%.

References

[1] G. A. Allan, A. J. Walton, and R. J. Holwill, "A yield improvement technique for IC layout using local design rules," *IEEE Trans. Comput.-Aided Design*, vol. CAD-11, pp. 1355–1362, Nov. 1992.

[2] V. K. R. Chiluvuri and I. Koren, "Layout-synthesis techniques for yield enhancement," *IEEE Trans. Semiconduct. Manufact.*, vol. 8, pp. 178–187, May 1995.

[3] G. A. Allan and A. J. Walton, "Automated redundant via placement for increased yield and reliability," in *Proc. SPIE Symp. Microelectronic Manufacture*, Austin, TX, Oct. 1997, pp. 114–125.

[4] T. C. Brennan and Assigned to International Business Machines Corporation, "Method for adding redundant vias on VlSI chips," U.S. Patent 6 066 179, Apr. 2003.

[5] A. Brand, A. Haranahalli, N. Hsieh, Y. C. Lin, G. Sery, N. Stenton, B. J. Woo, S. Ahmed, M. Bohr, S. Thompson, and S. Yang, "Intel's 0.25 micron, 2.0 volts logic process technology," *Intel Technol. J.*, 1998.

[6] E. Bruls, "Quality and reliability impact of defect data analysis," *IEEE Trans. Semiconduct. Manufact.*, vol. 8, pp. 121–129, May 1995.

[7] G. A. Allan and A. J. Walton, "Hierarchical critical area extraction with the EYE tool," in *Proc. IEEE Workshop Defect Fault Tolerance in VLSI Systems*, Lafayette, LA, Nov. 1995, pp. 28–36.

[8] C. H. Stapper and R. J. Rosner, "Integrated circuit yield management and yield analysis: Development and implementation," *IEEE Trans. Semiconduct. Manufact.*, vol. 8, no. 2, pp. 95–102, May 1995.

[9] W. Maly and J. Deszczka, "Yield estimation model for VLSI artwork evaluation," *Electron. Lett.*, vol. 19, no. 6, pp. 226–227, Mar. 1983.

[10] D. M. H. Walker, *Yield Simulation for Integrated Circuits*, ser. International series in engineering and computer science. Boston, MA: Kluwer, 1987.

[11] G. A. Allan and A. J. Walton, "Efficient extra material critical area algorithms," *IEEE Trans. Comput.-Aided Design*, vol. 18, pp. 1480–1486, Oct. 1999.

[12] G. A. Allan, "Yield prediction by sampling IC layout," *IEEE Trans. Comput.-Aided Design*, vol. 19, pp. 359–371, Mar. 2000.

[13] ——, "Property estimation of an integrated circuit," U.S. Patent 6 066 179, May 2000.

[14] G. A. Allan and A. J. Walton, "Yield prediction by sampling with the EYES tool," in *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems*, Boston, MA, Nov. 1996, pp. 39–47.

[15] *EYES: User Manual*, Predictions Software Ltd., Edinburgh, U.K., 2004.

[16] S. Levasseur and F. Duvivier, "Application of a yield model merging critical areas and defectivity data to industrial products," in *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems*, Paris, France, Oct. 1997, pp. 11–19.

[17] S. Barberan and F. Duvivier, "Management of critical areas and defective data for yield trend modeling," in *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems*, Austin, TX, Nov. 1998, pp. 17–25.

[18] G. A. Allan and A. J. Walton, "Critical area extraction for soft fault estimation," *IEEE Trans. Semiconduct. Manufact.*, vol. 11, no. 1, pp. 146–154, Feb. 1998.

[19] C. H. Stapper, "Modeling of integrated circuit defect sensitivities ," *IBM J. Res. Develop.*, vol. 27, no. 6, pp. 549–557, Nov. 1983.

Gerard A. Allan received the B.Sc. degree in mechanical engineering and the M.Sc. degree in design and fabrication of microelectronics systems, and the Ph.D. degree from the University of Edinburgh, Edinburgh, U.K., in 1985, 1986, and 1992, respectively.

After receiving his doctorate, he was employed as a Research Associate in the Department of Electronic and Electrical Engineering, Edinburgh University, from 1992 to 1998, and later as a Lecturer until 2003. He is now the Director of Predictions Software Ltd. His current research interests include layout optimization, yield prediction, computer-aided design tools, and IC design.