Paper 210-27

Constructing and Publishing Wafer Maps on the World Wide Web Version 8 of the SAS® System

Scott Lacey, Advanced Micro Devices, Austin, TX
David J Meade, Advanced Micro Devices, Austin, TX

ABSTRACT

Wafer maps have become one of the most useful methods of data analysis in the semiconductor industry. By incorporating various plotting features into these maps, spatial patterns for passing and failing die become readily apparent. Publishing and communicating the results from these analyses to other interested parties is also of paramount importance. We illustrate how PROC GMAP, part of SAS/GRAPH®, can be used to create wafer maps. Various plotting features are incorporated. Advances in web based programming for SAS also make it possible to display the analysis results on the World Wide Web (WWW). In this paper we demonstrate how PROC GMAP, PROC UNIVARIATE and the SAS Output Delivery System can be used together to generate and publish interactive semiconductor wafer maps on the WWW. The use of SAS Macro Language is also discussed and illustrated.

INTRODUCTION

One of the most important analysis tools in the semiconductor industry is the wafer map. By way of review, a wafer map is a graph that shows each individual unit, or die, on the surface of a silicon wafer. Various properties associated with each die can be illustrated on the map by incorporating a variety of graphing features. With recent advances in SAS versions 7 and 8, publishing information on the WWW has never been easier. The SAS Output Delivery System (ODS) makes it possible to convert SAS reports into the HTML format, which can then be directly displayed on the Internet.

THE OUTPUT DELIVERY SYSTEM

The following excerpt from the SAS Version 7 online help provides a good introduction to the Output Delivery System.

Prior to Version 7, SAS procedures that produced printed output (that is, output destined for the procedure output file) generated output that was designed for a traditional line-printer. In Version 7, procedure output is much more flexible.

The Output Delivery System (ODS) enhances your ability to manage procedure output. Procedures that fully support ODS

- combine the raw data that they produce with one or more templates to produce one or more output objects that contain the formatted results
- store a link to each output object in the Results folder in the Results window
- can generate HTML files that contain the formatted results and links (such as a table of contents) to the results
- can generate output data sets from procedure output
- provide a way for you to customize the procedure output by creating templates that you can use whenever you run the procedure.

These new features combined with the analytical tools available in SAS, are a powerful combination. Throughout the remainder of this paper we will provide examples of how the ODS can be used to convert wafer maps generated using PROC GMAP into HTML readable files that can be displayed on the WWW.

Wafer Maps Using PROC GMAP

In our introduction to this paper we suggested that one of the most useful graphs in the semiconductor industry is the wafer map. We will now illustrate how PROC GMAP, the SAS Macro Language, PROC UNIVARIATE, and the Output Deliver System can be used to generate high quality wafer maps that also contain special features that aid in signal detection. It is important to note that several types of data can be displayed on a wafer map. Some wafer maps display the actual failure bins for each die. Other maps are designed to show how a numeric parameter varies from die to die. In the example shown below, we will build a wafer map that illustrates how a numeric parameter changes from die to die. The size of each die in the wafer map will be proportional to the value of the parameter we are plotting. This technique greatly increases the viewer's ability to see patterns in the data, as shown in Figure 1. In this example, the parameter of interest increases as you move from the bottom left toward the top right of the wafer. The varying die size makes this pattern clearly visible. We will now illustrate how SAS can be used to create wafer maps like the one shown in Figure 1.

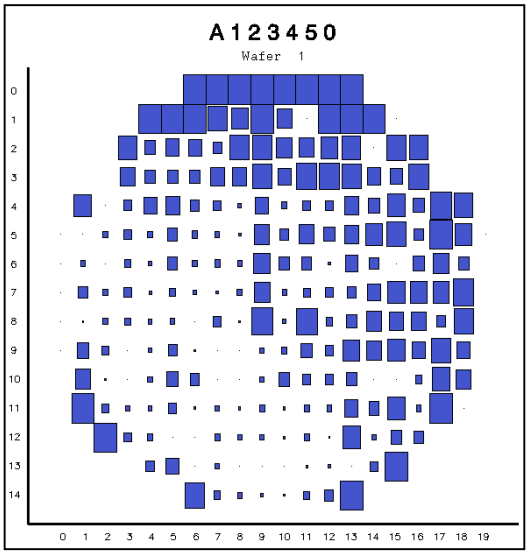

Figure 1: Example of wafer map with parametric pattern illustrated using varying die size.

Data Import: As with most other tasks, the first step in building a wafer map is to access and clean up the data. This is accomplished via the code shown in Figure 2.

```
%let idno  =%scan(&sysparm,1,' ');
%let testno=%scan(&sysparm,2,' ');
%let outdir=/sugi27/sasout;
%let urldir=/web/sugi27/sasout/;
data mapdata;
  infile &outdir/sugi27_&idno..mapdata"
    dlm='09'x;
  length lot device $8 wafer sortx sorty
    testno value testdate 8;
  informat testdate datetime18.;
  format testdate datetime16.;
  input lot device wafer sortx sorty
     testno value testdate ;
  if _n_=1 then do;
    call symput('LOT',trim(lot));
call symput('DEVICE',substr(device,1,4));
    end;
run;
proc sort data=mapdata;
  by lot wafer sortx sorty
      descending testdate;
run;
data mapdata(drop=testdate lot device);
  set mapdata; by lot wafer sortx sorty;
  if first.lot or first.wafer or
    first.sortx or first.sorty;
run;
proc sort data=mapdata(keep=wafer)
  nodupkey out=waflist; by wafer; run;
proc sort data=mapdata;
  by wafer sortx sorty; run;
proc sort data=mapdata(keep=sortx sorty)
  nodupkey out=sortxy; by sortx sorty;
run;
proc sql;
    create table fullwaf as
    select wafer, sortx, sorty
    from waflist, sortxy;quit;
proc sort data=fullwaf;
  by wafer sortx sorty; run;
data mapdata; merge mapdata fullwaf;
  by wafer sortx sorty;
run;
```

Figure 2: This code establishes access to the data and cleans it up.

Actual Die Size and Axis Orientation: The next step in the process is to determine the actual size of each die and orientation of the wafer axis. This step is basically a collection of DATA and SYMPUT statements. The code is found in Figure 3. This part of the code will need to be modified slightly to fit the needs of a given user.

```
* set die size and axis orientation;
data _null_;
  * axis: x_mult = -1 for reverse ;
  * xy_swap = 1 for swap x and y;
  x_mult=1;
  y_mult=1;
  xy_swap=0;
  * aspect = height / width ;
  aspect=1;
  select("&device");
    when('XX40') do;
      aspect=1.295;
      y_mult=-1;
    end;
    when('XXT0') aspect=1.125;
    otherwise;
  end;
call symput('ASPECT',put(aspect,best5.));
call symput('X_MULT',put(x_mult,3.));
call symput('Y_MULT',put(y_mult,3.));
call symput('XY_SWAP',put(xy_swap,2.));
run;
%put USERNOTE: aspect=&aspect
  x_mult=&x_mult y_mult=&y_mult
  xy_swap=&xy_swap;

* apply axis orientation changes ;
data mapdata(drop=tmp);
  set mapdata;
  length tmp x y 8;
  x = &x_mult * sortx;
  y = sorty * &y_mult;
  if &xy_swap then do;
    tmp=x;
    x=y;
    y=tmp;
  end;
run;
```

Figure 3: Die size and axis orientation code.

Calculation of Statistics: Our next task is to calculate the statistics, which will be used to determine the relative size of each die in the wafer map. Using the code shown in Figure 4, we determine the minimum, maximum, 10th percentile, and 90th percentile. If no outliers are present in the data, then we simply use the max and min values as reference points in determining the relative size for each plotted die. If outliers are present on either the high or low end of the data, then we use the 10th or 90th percentile respectively. Statistics are calculated using the UNIVARIATE Procedure.

Standardize Data For Proportional Die Size Calculations: Next, we standardize the numeric parameter data so that each data point is represented as a new value between 0 and 1. These new values are used to determine the relative size of each die in the wafer map. A die with a value of 1 will be represented as a rectangle that is 100% of the maximum size allowed in the wafer map. A die having a standardized value of 50% will be represented as a rectangle that is only 50% of the maximum size allowed. The code is shown in Figure 5.

```
* get the min, p10, p90, and max ;
proc univariate data=mapdata noprint;
  var value;
  output out=mapstat(keep=min p10 p90
     max) min=min p10=p10 p90=p90
     max=max;
run;

* reset min or max in case of outlier ;
data mapstat(drop=delta p10 p90);
  set mapstat;
  delta=p10-min;
  if delta>0 then do;
    delta=(p90-p10)/delta;
    if delta<1 then do;
      put 'USERNOTE: low outlier';
      min=p10; end;
  end;
  delta=max-p90;
  if delta>0 then do;
    delta=(p90-p10)/delta;
    if delta<1 then do;
      put 'USERNOTE: high outlier';
      max=p90; end;
  end;
run;
```

Figure 4: Code used to calculate statistics used for relative die size calculations.

```
* standardize data to 0-1 for die size;
data mapdata(drop=min max zmin zrange);
  merge mapdata mapstat;
  length zmin zrange zvalue chorvar 8;
  retain zmin zrange;
  if _n_=1 then do;
    zmin=min;
    zrange=max-min;
  end;
  if value>. then do;
    * do max(min()) in case of outliers ;
zvalue=max(min((value-zmin)/zrange,1),0);
    chorvar=1;
  end;
  else do;
    zvalue=0; chorvar=0; end;
run;
```

Figure 5: Code used to standardize actual data values. Standardized values are then used to determine the relative size of the die being plotted in the wafer map.

Generate Rectangles Which Define Each Die Location: The actual wafer map will be generated using PROC GMAP and the associated CORO statement. PROC GMAP provides the ability to display variations of a variable with respect to a specified area in the graph. In other words, each point on the graph is a defined area rather than a small discrete point on the graph. In the case of a wafer map, each of these defined areas will be a given die on the map. See Figure 6 for associated code.

```
* set regions used by gmap choro;
data mapdata; set mapdata;
  by wafer; retain dieno;
  if first.wafer then dieno=0;
  dieno+1;
run;
* make rectangles for the die ;
* scale y by aspect, and size by zvalue ;
* also recenter region;
data mapset(drop=zvalue);
  set mapdata(keep=wafer x y dieno
    zvalue);
  x=x+0.5-0.5*zvalue;
  y=(y+0.5-0.5*zvalue)*&aspect; output;
  x=x+zvalue; output;
  y=y+zvalue*&aspect; output;
  x=x-zvalue; output;
run;
```

Figure 6: Code that assigns data values to specific regions on the wafer map.

Generate Mouse Over Information: The next step in the process is to generate information that will be provided when the user points their mouse at any given die on the wafer map. This is an optional step. However, it provides an interactivity in the wafer map that most users appreciate. We accomplish this task by using the alternate name feature of anchor tags. Associating HTML with a region is a new feature available in PROC GMAP. Each area on the graph can be assigned an alternate name, which we then display when the user points the mouse at that region. In this example we assign the alternate name to be the die coordinates and the value of the numeric parameter. The code is shown in Figure 7. An example of the mouse-over functionality is illustrated in Figure 8. Drill down links can be done in a similar manner.

Generate X and Y Axis: We are now ready to begin drawing the wafer map. We begin by creating the X and Y Axes. The code shown in Figure 9 draws horizontal and vertical lines for the axes. It also places the appropriate labels on both axes.

Create Wafer Map in HTML Format: The final step in the process is to use PROC GMAP in connection with the ODS System to produce the wafer map in HTML format. As stated previously, PROC GMAP is a procedure that generates visual representations of data where a given data point is represented as a specified region in the graph. Within the GMAP procedure the CORO statement is used to create two-dimensional maps where values of the response variables are represented by varying patterns and colors. Finally, invocation of the ODS system enables SAS to generate the wafer maps in HTML format which can then be displayed on the World Wide Web. The code for this final step in the process is show in Figure 10. The resulting wafer map is shown in Figure 11.

```
* make mouse-over info for each die ;
data mapdata; set mapdata;
  length html $200; html='ALT="('
||compress(put(sortx,best5.))
||','||compress(put(sorty,best5.))||')';
  if value=. then html=trim(html)
||byte(13)||'Missing"';
  else html=trim(html)||byte(13)
||compress(put(value,best10.))||'"';
run;

pattern1 v=solid c=blue;
```

Figure 7: Code used generates the alternate names used in the mouse-over functionality of the wafer map.

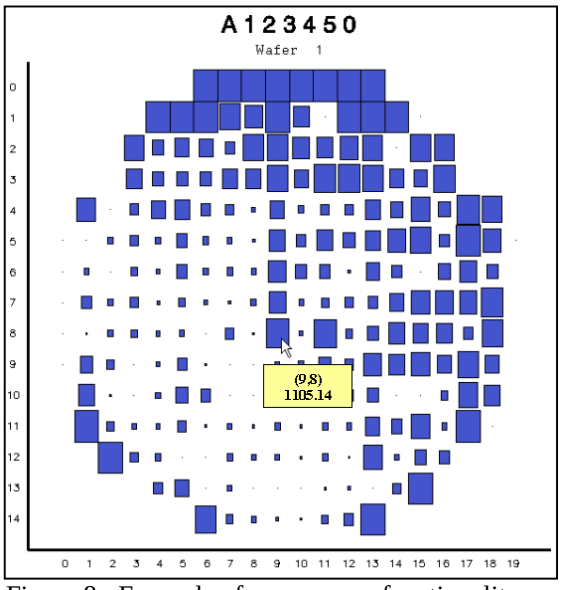

Figure 8: Example of mouse-over functionality.

```
* axis annotation
data annodata;
  length function color style text $8
    position hsys xsys ysys when $1;
  length x y size line 8;
  line=1; position='5'; xsys='3';
  ysys='3'; hsys='3'; size=0.5; when='B';
  color='BLACK'; style='';  text='';
  x=15; y=03; function='MOVE'; output;
  x=85; function='DRAW'; output;
  x=15; y=03; function='MOVE'; output;
  y=90; function='DRAW'; output;
run;
* put text labels on axes;
%macro axislab;
  %if &XY_SWAP=1 %then %do;
    %let xlab=y; %let ylab=x; %end;
  %else %do;
    %let xlab=x; %let ylab=y; %end;
  * get axis stats ;
  proc summary data=mapdata noprint;
    var x y; output out=statxy(keep=xmax
      ymax xmin ymin) max=xmax ymax
      min=xmin ymin;
  run;
  * x-axis annotation ;
  proc sort data=mapdata(keep=x
      sort&xlab) nodupkey out=xaxis;
    by x;
  run;
  data xaxis(drop=incr xmax xmin xxmin
      sort&xlab);
    merge xaxis statxy(keep=xmin xmax);
    length function color style text $8
      position hsys xsys ysys when $1;
    length x y size line incr xxmin 8;
    retain incr xxmin;
    line=1; position='5'; xsys='3';
    ysys='3'; hsys='3'; size=2.0;
    when='B'; color='BLACK';
    function='LABEL'; style='simplex';
    if _n_=1 then do; xxmin=xmin;
      incr=60/(xmax-xmin); end;
      x=20+(x-xxmin)*incr; y=01;
    text=compress(put(sort&xlab,best5.));
  run;
  proc append data=xaxis base=annodata;
  proc sort data=mapdata(keep=y
    sort&ylab) nodupkey out=yaxis;
    by y; run;
  data yaxis(drop=incr ymax ymin yymin
      sort&ylab);
    merge yaxis statxy(keep=ymin ymax);
    length function color style text $8
      position hsys xsys ysys when $1;
    length x y size line incr yymin 8;
    retain incr yymin;
    line=1; position='5'; xsys='3';
    ysys='3'; hsys='3'; size=2.0;
    when='B'; color='BLACK';
    function='LABEL'; style='simplex';
    if _n_=1 then do; yymin=ymin;
      incr=77/(ymax-ymin); end;
    y=09+(y-yymin)*incr; x=13;
    text=compress(put(sort&ylab,best5.));
  run;
  proc append data=yaxis base=annodata;
run;
%mend axislab;
%axislab;
```

Figure 9: Code used to generate X and Y axes.

```
goptions device=gif transparency
  gsfname=grafout graphrc;

title h=4 pct f=swissu "&lot";
* footnote saves for horizontal axis;
footnote h=0.2 pct ' ';

* make the graphs, changing the map ;
* dataset for each wafer and use ods ;
* to make html output;
%macro wafloop;
  * get count of wafers;
  data _null_;
    if 0=1 then set waflist nobs=nobs;
   call symput('wafct',put(nobs,best5.));
    stop;
  run;
  %do mi=1 %to &wafct;
    data _null_;
      set waflist(firstobs=&mi obs=&mi);
   call symput('currwaf',put(wafer,2.0));
    run;
    title2 "Wafer &currwaf";
    ods html style=styles.minimal
body="&outdir/wmap_&idno._&mi..html"
      gpath="&outdir"(url=none)
      base="&urldir" newfile=page;
    * draw the graph ;
    proc gmap data=mapdata(
where=(wafer=&currwaf)) map=mapset(
where=(wafer=&currwaf))
        anno=annodata;
    id dieno;
    choro chorvar / html=html missing
      coutline=black discrete nolegend;
  run;
  quit;
  ods html close;
  %end;
%mend wafloop;
%wafloop;
```

Figure 10: Code used to generate the actual wafer map using PROC GMAP and the ODS System.

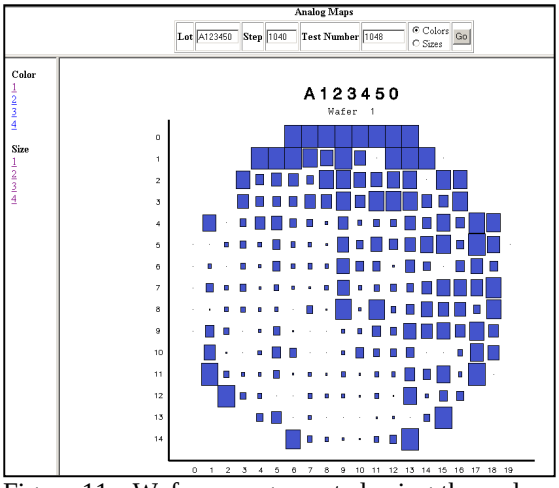

Figure 11: Wafer map generated using the code in this paper.

Summary

One of the most important graphical tools available in the semiconductor industry is the wafer map. PROC GMAP and PROC UNIVARIATE can be used in conjunction with SAS Macros and the SAS Output Delivery System to generate wafer maps. These wafer maps can then be displayed on the World Wide Web. These methods have been used at AMD to identify yield and parametric spatial patterns.

APPENDIX

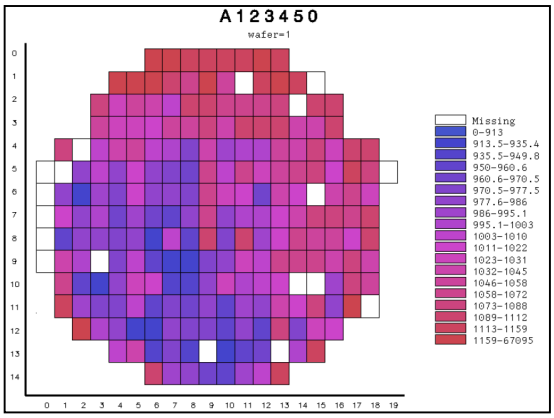

Figure 12: The methods illustrated in this paper can also be used to create wafer maps which display a numeric parameter in varying shades of color. In this example shades of blue represent low values of the parameter. Shades of red represent high values. A legend is provided on the right hand side of the wafer map.

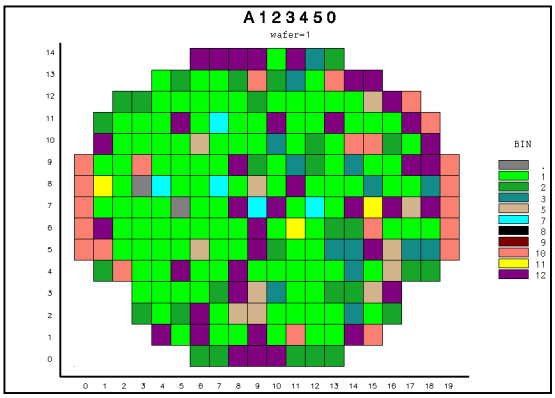

Figure 13: This is an example of a bin wafer map. In this type of map the failure or passing bin of each die is represented by a different color.

ACKNOWLEDGMENTS

CONTACT INFORMATION

Scott Lacey
5204 E. Ben White Blvd.
Mail Stop 601
Austin, TX 78741
USA
Email: scott.lacey@amd.com

David J. Meade
5204 E. Ben White Blvd.
Mail Stop 613
Austin, TX 78741
USA
Email: david.meade@amd.com