Magic - Multiple Analyses of the Geometry of Integrated Circuits

by **Martin Newell** and **Dan Fitzpatrick** and **Rich Pasco**

Version of April 26, 1982

[Indigo]<DA>Magic>Magic.bravo,.press

PREFACE

Magic is a system providing Multiple Analyses of the Geometry of Integrated Circuits. The system, which runs under the JaM system on Dolphins and Dorados, is configured as a framework providing commonly used facilities, on which separate modules implementing the various analyses have been implemented. As such the system is extensible as more input formats and analyses are required. Currently full CIF 2.0 and Icarus format files are supported. The following analyses are implemented: viewing on the screen, outlined plots on the Versatec plotter and Ramtek ColorGraphic Printer, color Press output, Mead/Conway geometric design rule checking, switch level circuit extraction, electrical rules checking, dynamic simulation (by running SimMOS under Magic), and conversion to various other formats including filtered CIF, Icarus, and Mann 3000.

XEROX
PALO ALTO RESEARCH CENTER
3333 Coyote Hill Road
Palo Alto California 94304

CONTENTS

1. INTRODUCTION

1.1 Acknowledgements

2. OPERATION

2.1 Input Formats

2.2 Usage

2.3 Where to find Magic

3. COMMANDS

3.1 Interactive usage

3.2 Initialization

3.3 Layout input

3.4 Viewing

3.5 Selection of Analysis Module

3.6 Commands common to all analysis modules

4.0 REFERENCES

5.0 INDEX OF COMMANDS

6.0 APPENDICES

A1. Color Press Module
A2. Circuit Extractor Module
A3. Electrical Rules Checking Module
A4. Design Rule Checking Module
A5. Versatec Plotting Module
A6. Mann Output Module
A7. Ramtek Output Module

1. INTRODUCTION

Magic is a system providing Multiple Analyses of the Geometry of Integrated Circuits. The system is configured as a framework providing commonly used facilities, on which separate modules implementing the various analyses have been implemented. As such the system is extensible as more input formats and analyses are required. Currently input from CIF 2.0 and Icarus format files is supported. The following analyses are implemented: viewing on the screen, outlined plots on the Versatec plotter, color Press output, Mead/Conway design rule checking, switch level circuit extraction (with subsequent electrical rules checking and simulation, see Magic), and conversion to various other formats including filtered CIF, Icarus, and Mann 3000.

1.1 Acknowledgements

To Bob Hon for provision and maintenance of the CIF Parser and Interpreter that provide the basis for the core of the system. To Forest Baskett for the electrical rules checking analysis. Also to the many people who have provided much valuable feedback from earlier versions of the system.

2. OPERATION

Magic is implemented as a set of programs in Mesa. Although the core of the system will run on an Alto, most analysis modules will run only on a D machine. This results principally from the difficulty of using more than 1 bank of memory for data structures on an Alto.

The basic Magic system is implemented in Mesa, and the user interface is provided by running Magic under the interactive programming system JaM [JaM]. JaM provides a simple interpreter that supports the JaM programming language, and from which Mesa programs can be called. For the purposes of this document the term Magic will be used to cover the combination of the basic implementation and the JaM user interface.

2.1 Input Formats

Layouts can be specified to Magic in one of two formats: Caltech Intermediate Form (CIF) or Icarus, or as a combination of both.

2.2 Usage

Magic can be used either interactively or via command files. The command files provide a very simple interface to the more commonly used functions. More involved analyses require using Magic interactively, which therefore requires a knowledge of JaM. This document does not include a JaM tutorial - the reader is refered to the JaM manual [JaM].

Typical usage involves retrieving a command file from Indigo for the analysis required, and executing it to retrieve the necessary programs and files. This step is necessary for only the first usage. One of the files will be a .do file whose execution requests information about input files and parameters pertinent to the analysis being performed.

Frequently the facilities provided by the command files are inadequate, in which case Magic should be used interactively. This takes the form of commands to Jam and use of the mouse. Commands applicable to all analyses are provided later in this document, and commands relevant to specific analyses are contained in the Appendices.

2.3 Where to find Magic

Magic can be obtained and initialized by executing one of the command files:

[Indigo]<DA>Magic>Magic*.cm

where the * stands for the name of the analysis required, e.g.

[Indigo]<DA>Magic>MagicDRC.cm

will get and initialize Magic and the Design Rule Checking module. Also on that directory is a file:

[Indigo]<DA>Magic>UpdateMagic.cm

that will update your disk for the entire system, including getting the system for the first time. Periodic execution of this command file is recommended. There is no need to update UpdateMagic.cm itself as it is expected to never change.

3. COMMANDS

Command file usage of Magic for any one analysis involves using one of two .do files - for CIF and Icarus input. For example, CIFtoDRC.do and ICtoDRC.do are used for design rule checking. The .do files for the various analyses require different parameters. They are documented in the Appendices.

3.1 Interactive usage

The interactive use of Magic usually proceeds as follows:

1. read one or more layouts into the Magic database;
2. use viewing commands to see what is there;
3. select an analysis module;
4. set up precise view and parameters for selected analysis;
5. invoke the analysis;
6. repeat from 3.

The reason for having selection of an analysis module separate from its invocation is that certain analysis modules affect the way in which the display is managed. For example, the Press module forces the viewing region to be the shape of the array of pages requested. Also, certain state can be set within a module only while it is selected.

The available commands are from three sources - commands directed to JaM, primitive Magic commands, and JaM procedures. While users need not be concerned with these distinctions, it may be helpful to note that online documentation of most commands that are implemented as JaM procedures can be obtained using the JaM utility command "?", e.g. (reset)?.

The commands are presented here in two categories: those common to all analyses, and those specific to particular analyses. The latter are documented in the Appendices. Screen output is implemented as an analysis, but it is treated specially in that the system always includes it.

3.2 Initialization

First get and initialize Magic as described under 2.3 "Where to find Magic". Magic is normally started by typing:

```
jam magic
```

When loaded Magic should prompt with the message:

```
Magic
##
```

at which time it is ready for commands. The initial getting and initializing of Magic will leave Magic in this same state.

reset

Usage: **reset**
Initialize the stored layout to null. The layout is automatically initialized on start-up, so reset need be called only to clean out an existing layout. e.g.

```
reset
```

To leave Magic in a controlled way use the normal JaM termination command **.quit**.

3.3 Layout input

readcif

Usage: <filename> **readcif**
Read the layout in CIF format from the indicated file, and add it to the layout currently stored. The extension .cif is assumed if no extension is given. The view is set up to show the whole of the currently stored layout fitted to the drawing area. e.g.

```
(mychip)readcif
```

will read a layout from the file mychip.cif.

readic

Usage: <filename><CIF units><Icarus units> **readic**
Read the layout in Icarus format from the indicated file, and add it to the layout currently stored. The extension .ic is assumed if no extension is given. The second and third parameters provide for scaling, where CIF units are in hundredths of a micron. The view is set up to show the whole of the currently stored layout fitted to the drawing area. e.g.

```
(myotherchip) 250 3 readic
```

will read a layout from the file myotherchip.ic, and scale it to 2.5 microns per 3 Icarus units. Icarus is normally used with 3 Icarus units representing 1 lambda, so the above specification would be appropriate for lambda = 2.5 microns.

parsecif

Usage: <filename> **parsecif**
As for readcif except that the current view is not altered. e.g.

```
(mychip)parsecif
```

parseic
Usage: <filename><CIF units><Icarus units> **parseic**
As for readic except that the current view is not altered. e.g.

```
(myotherchip) 250 3 parseic
```

3.4 Viewing

Viewing is accomplished by the Screen output module, which is always loaded. Viewing is controlled by a combination of the mouse and keyboard. The mouse is used for zooming, scrolling, and changing the window. Keyboard commands are used to change modes.

Zoom: Red mouse button. Define a rectangle by pressing down at one corner, moving the mouse to the other corner, then letting the button up. While the button is down the rectangle is indicated by video reversing its contents. Zooming up or down is determined by the relative positions of the two corners of the rectangle.

If the second corner is up and to the right of the first then the image is expanded so that the contents of the rectangle just fit into the current viewing area. This is a best fit - the image is never differentially scaled.

If the second corner is down and to the left of the first then the image is contracted so that the contents of the current viewing area just fit into the rectangle.

Scroll: Yellow mouse button. Press down at the source position, move the mouse to the destination position, then let the button up. The image is then scrolled so that whatever was at the source position now appears at the destination position.

Window: Blue mouse button. The viewing region can be redefined as a rectangle in much the same way as for zooming, except using the Blue button. The relative positions of the two corners has no effect.

draw
Usage: **draw**
Redraw the image. Drawing can be interrupted using the JaM break key (rightshift-swat). See also the command *outline* for dealing with complex files.

outline
Usage: **outline**
Set a mode such that only the outline of the current layout is drawn. This is to allow manipulations of the image without having to wait for all the detail to be drawn.

unoutline
Usage: **unoutline**
Unset outline mode.

fit
Usage: **fit**
The image is scaled and positioned so that the entire layout just fills the current viewing region.

clipfit
Usage: **clipfit**
The clipping region is adjusted to just surround the layout.

atorigin
Usage: **atorigin**

The image is positioned so that the bottom left corner of the layout is at the bottom left corner of the current viewing region.

scrollto

Usage: <x><y> **scrollto**
The image is scrolled to bring the point <x><y> to the center of the current viewing region. <x> and <y> are in CIF units (hundredths of a micron).

invis

Usage: <layer name> **invis**
The given layer name is made invisible for the purposes of both subsequent drawing on the screen, and subsequent analysis by any analysis module. The name is a string conforming to the CIF conventions. e.g.

```
(NM) invis
```

will suppress the metal layer.

vis

Usage: <layer name> **vis**
This is the inverse of invis. The given layer name is made visible for the purposes of both subsequent drawing on the screen, and subsequent analysis by any analysis module. The name is a string conforming to the CIF conventions. e.g.

```
(NM) vis
```

will restore the metal layer to visibility.

show

Usage: <log file name> **show**
This command is used to examine a log file, showing each text message in the typescript window while showing the corresponding geometry in the viewing region. e.g.

```
(analysis.log) show
```

For each entry in the file analysis.log the text portion will be output to the typescript window, and the corresponding geometry scrolled to the center of the viewing region, using the currently set up scale. It is therefore advisable to set up a suitable scale using the mouse before using show. After each line the prompt *More?* will appear, to which you should type <*CR*> to see the next entry, or *n*<*CR*> to suspend show. Note that no other command, nor the mouse, can be used while show is in control. See also the command *more*.

The command *show* is available for viewing any analysis log that conforms to the following syntax requirements. Each entry should be a line of text starting with two decimal numbers, each terminated with at least one space, which give the coordinates of the feature in CIF coordinates. The remainder of the line should contain the associated message.

more

Usage: **more**
Resume showing entries in a log after suspending a *show* command.

boundbox

Usage: **boundbox** => <left><bottom><right><top>
Return on the JaM stack the bounding box of the current layout.

3.5 Selection of Analysis Module

An analysis module is selected as current by a command of the form

<parameters> **<module>**

The <parameters> are dependent on the **<module>** being selected. The first time this command is issued for a given module will result in a delay while the code for the module is being loaded. Subsequent selections of the module are faster.

e.g.

```
2 4 press
```

selects the Press module which expects two parameters giving the number of pages to subsequently use for printing.

screen
Usage: **screen**
This is a particular example of a module selection command that deselects any currently selected analysis module. It also directs commands in the next section to the screen module, such as *setstipple*.

3.6 Commands common to all analysis modules

This section describes commands common to all analysis modules, but whose implementation may vary depending on the currently selected module.

output
Usage: <parameters> **output**
Output the layout to the currently selected analysis module. The <parameters> are dependent on the currently selected analysis module.

setscale
Usage: <scale factor> **setscale**
Set the scale in the currently selected analysis module. Interpretation of the scale factor is dependent on the currently selected analysis module.

setstipple
Usage: <layer><stipple> **setstipple**
Load the stipple for the given <layer> in the currently selected analysis module. The <stipple> is a single number. Interpretation of the stipple is dependent on the currently selected analysis module.

setcolor
Usage: <layer><color> **setcolor**
Load the color for the given <layer> in the currently selected analysis module. The <color> is a JaM array of three numbers. Interpretation of the color is dependent on the currently selected analysis module.

4.0 REFERENCES

[JaM] JaM, John Warnock and Martin Newell, [Indigo]<JaM>JaM.bravo.

[Mead and Conway] *Introduction to VLSI Systems*, Carver Mead and Lynn Conway, Addison Wesley, 1980.

5.0 INDEX OF COMMANDS

atorigin 3.4
boundbox => <left><bottom><right><top> 3.4
clipfit 3.4
draw 3.4
drc A4.3
<filename><lowlimit><highlimit> **erc** A3.2
extractor A2.3
fit 3.4
<layer name> **invis** 3.4
<units> **mann** A6.3
more 3.4
outline 3.4
<parameters> **output** 3.6
Press: <1st page row><1st page column>
<last page row><last page column><filename> **output** A1.3
Extractor: <filename> **output** A2.3
DRC: <filename> **output** A4.3
Versatec: <1st strip><last strip><filename> **output** A5.3
Mann: <filename> **output** A6.3
<filename> **parsecif** 3.3
<filename><CIF units><Icarus units> **parseic** 3.3
<pages down><pages across> **press** A1.3
<filename> **readcif** 3.3
<filename><CIF units><Icarus units> **readic** 3.3
reset 3.2
screen 3.5
Scroll 3.4
<x><y> **scrollto** 3.4
<layer><color> **setcolor** 3.6
<scale factor> **setscale** 3.6
<layer><stipple> **setstipple** 3.6
<log file name> **show** 3.4
unoutline 3.4
versatec A5.3
<layer name> **vis** 3.4
Window 3.4
Zoom 3.4

6.0 APPENDICES

A1. Color Press Module

The Color Press module produces color press files of the layout. The layout is split into separate pages that are generated on separate files. The image on each page overlaps its neighbor by 1/4", to allow for inaccuracies in trimming the paper. Each page is identified with page number, file name, and user name.

A1.1 Initialization

First get and initialize Magic for Press output as described under 2.3 "Where to find Magic", either by retrieving and executing

[Indigo]<DA>DStar>MagicPress.cm

or by getting everything from

[Indigo]<DA>DStar>UpdateMagic.cm

A1.2 Command file usage

Two .do files are provided for converting a single layout to Color Press output, one for CIF input and one for Icarus:

CIFtoPress.do
ICtoPress.do

These can be run using Do.run and answering the prompts for parameters, or by providing parameters on the command line in the form:

Do CIFtoPress <infile><outfile><pages down><pages across>

Four optional parameters are supported, to give the indices of the first and last pages to be generated. These can be given only from the command line which then takes the form:

Do CIFtoPress <infile><outfile><pages down><pages across> ^
<1st page row><1st page column><last page row><last page column>

This will fit the entire layout to the <pages down><pages across> requested, and will generate the pages in row order between <1st page row><1st page column> and <last page row><last page column>, inclusive, in row order. Any of the parameters can be out of range in which case the maximum or minimum permissible parameter value is used. Each page is output to a separate press file, the names being of the form: filename-i-j.press, where i and j are the row and column numbers of the page. If the filename starts with a fileserver name the pages are written to the file server as they are generated. This minimizes the need for local file space for the press files. Be warned that you must be logged in to the executive with a valid password before using this option.

A1.3 Interactive usage

Interactive use of Magic for generating Press output is useful for experimenting with the fit of the layout to a given set of pages, for extracting a piece of the layout for printing, and for superimposing several layouts for printing.

press
Usage: <pages down><pages across> **press**
Select the Press module for subsequent output. While the press module is selected the viewing area is constrained to be the shape of the array of pages requested. The outlines of

the pages are superimposed on the view. e.g.

```
2 3 press
```

will select press for output on an array of 2 pages down by 3 pages across.

Press can be reselected for adjusting the array of pages. Each time ***press*** is selected, a maximal-size array of pages of the specified aspect ratio is laid out within the existing clipping region, and then the clipping region is shrunk to conform to the aspect ratio of the new page array. Thus, reselecting press with a different aspect ratio will cause the clipping region to shrink. In order to scale the drawing to fit within the new, smaller clipping region, use the ***fit*** command. If repeated use of ***press*** leads to an awkwardly small clipping region, it's size may be reset using the blue mouse bug Window command, as described in Section 3.4 Viewing.

output

Usage: <1st page row><1st page column>
<last page row><last page column>
<filename> **output**

Output the current contents of the viewing area to the press module to generate the pages in row order between those indicated in the first four parameters, inclusive. Any of the parameters can be out of range in which case the maximum or minimum permissible parameter value is used. Each page is output to a separate press file, the names being of the form: filename-i-j.press, where i and j are the row and column numbers of the page. If the filename starts with a fileserver name the pages are written to the file server as they are generated. This minimizes the need for local file space for the press files. Be warned that you must be logged in to the executive with a valid password before using this option. e.g.

```
1 1 1000 1000 (chip) output
```

will output all pages (assuming the selected array is smaller than 1000x1000) onto files chip-1-1.press, chip-1-2.press, ..., chip-2-1.press, ... etc.

setcolor

Usage: <layername><color array> **setcolor**

Set the hue, saturation, and brightness of the given layer to the values in the color array, which must contain 3 numbers. e.g.

```
(NM) [ 160 255 255 ] setcolor
```

will set the color of layer NM to be bright blue.

The current setting of the color is retained in the JaM virtual memory, and is automatically restored on subsequent runs.

A2. Circuit Extractor Module

The Circuit Extractor Module generates a switch level circuit representation of the layout. It understands only NMOS technology, including buried contacts, but is not sensitive to design rules. The extractor generates two files - a .sim file for subsequent electrical rules checking [erc] and/or simulation [SimMOS], and a .nodes file that is a legitimate cif file containing all names with their positions, including names generated for unnamed nodes. This file can subsequently be read back in and combined with the original layout for viewing or plotting of the layout with node numbers.

Names in the input layout file are interpreted. Each name is taken to name the point that determines the position of the name. Each of these points should be inside a unique layer of geometry, and it is taken to name the node of which that piece of geometry is a part. Names over no geometry are reported. Names of more than one node are also reported, and the name is assigned to one of the nodes, generated names being used for the other nodes. A node can be named more than once. However, if the names differ a warning is output and one of the names is abandoned.

Note that the names Vdd and Gnd are required by subsequent circuit analysis tools. Beware of local names that appear within symbols that are used more than once. All names are global, and therefore the nodes within symbols that have the same name will effectively all be connected together.

A2.1 Initialization

First get and initialize Magic for Circuit Extraction as described under 2.3 "Where to find Magic", either by retrieving and executing

[Indigo]<DA>DStar>MagicExtractor.cm

or by getting everything from

[Indigo]<DA>DStar>UpdateMagic.cm

A2.2 Command file usage

Two .do files are provided for extracting the circuit from a single layout, one for CIF input and one for Icarus:

CIFtoExtractor.do
ICtoExtractor.do

These can be run using Do.run and answering the prompts for parameters. Two files are generated: a .sim file, which contains the extracted circuit; and a .nodes file that contains a legitimate CIF file giving the node names, including generated names.

A2.3 Interactive usage

Interactive use of Magic for Circuit Extraction is useful for extracting a piece of the layout for extraction, and for avoiding the need to restart Magic and read the layout again in order to examine the results of the extraction.

extractor

Usage: **extractor**
Select the Circuit Extractor module for subsequent output.. e.g.

```
extractor
```

output

Usage: <filename> **output**

Extract the circuit represented by that part of the layout currently in the viewing area. Two files are generated: filename.sim, which contains the extracted circuit; and filename.nodes that contains a legitimate CIF file giving the node names, included generated names. e.g.

```
(chip) output
```

will generate chip.sim and chip.nodes.

A3. Electrical Rules Checking Module

The Electrical Rules Checking Module, ERC, checks the .sim file (see A2 above) for conformance with Mead/Conway [Mead/Conway] electrical rules for NMOS. ERC generates one file -

.erclog file that lists all electrical rule violations, in *show* syntax;

The *show* and *more* commands described in Section 3.4 are useful for examining the .erclog file.

A3.1 Initialization

First get and initialize Magic for Circuit Extraction as described under 2.3 "Where to find Magic", either by retrieving and executing

[Indigo]<DA>DStar>MagicExtractor.cm

or by getting everything from

[Indigo]<DA>DStar>UpdateMagic.cm

A3.2 Interactive usage

Interactive use of Magic for ERC is useful for extracting a piece of the layout and checking it, and for avoiding the need to restart Magic and read the layout again in order to examine the results.

erc

Usage: <filename><lowlimit><highlimit> **erc**
ERC the circuit represented by the .sim file previously generated by the extractor. <lowlimit> and <highlimit> are two floating point numbers that specify the range of acceptable ratios for the ratio checks. One file is generated: filename.erclog, which contains a log of all electrical rule violations. Summary information on the electrical characteristics is output to the typescript window. e.g.

```
(chip) 3.5 4.5 erc
```

will generate chip.erclog and display a summary in the typescript window.

A convenient way of viewing the results is to use the *show* and *more* commands described in Section 3.4. Do this by *show*ing the .erclog file.

A3.4 Explanation

The summary messages that appear in the typescript window are

```
nodes, transistors, pullups : %d %d %d
pullups and non-inverting superbuffers : %d %d
inverting superbuffers and unknowns    : %d %d
Sum of 1/Z's for all pullups : %d
(syntax error : %s)
```

where *%d* is a decimal number. The decimal numbers are just counts of the designated types of components found in the circuit by the electrical rules checker. The sum of the 1/Z's can be used to estimate maximum power dissipation of the circuit by multiplying by the appropriate number for the process you are using (approximately .6 milliwatts per 1/Z for NSIL-II). The *syntax error* message shouldn't appear unless your files are confused.

The messages that appear in the .erclog file are listed below in alphabetical order with short explanations of what they might mean. *%n* indicates a node name in the message.

Assuming lightning arrested node is an Input : %n

A lightning arrestor is a transistor whose gate and drain are Gnd. The source of that transistor is assumed to be a TTL level input from off chip. Note that a TTL high level is approximately two thresholds below an NMOS restored high. ERC assumes this when checking thresholds and ratios.

depletion transitor not a pullup or a superbuffer : d %n %n %n

A depletion mode transistor that might be part of a function block?

Gate is GND : e %n %n %n

In the listed transistor the gate is Gnd but the transistor is not of the lightning arrestor type. You may have an instance of a standard cell in which one of the inputs is always 0.

gate is the same node as source or drain : e %n %n %n

An unusual thing to do.

Gate is VDD : e %n %n %n

In the listed transistor the gate is Vdd. You may have an instance of a standard cell in which one of the inputs is always 1.

Node can never be given a value : %n

There is no path to Gnd or to Vdd from this node.

Node can never be set to 1 : %n

There is no path to Vdd from this node.

Node can never be set to 0 : %n

There is no path to Gnd from this node.

node might be VDD : d %n %n %n

A depletion mode transistor with gate tied to source. The drain is usually Vdd, but not in this case.

Node name only occurs once : %n

A dangling node. Can't possibly be useful. Maybe it's an output that you are not using from a standard cell.

Node pulled up more than once : %n

Do you want to do this?

Non-restored node controlling input : %n %n

A negative going glitch on the input can destroy the input value you thought you had latched.

```
Non-restored node controlling pass transistor : %n %n
```

Normally not done.

```
pullup/pulldown ratio = %f node %n pulled up thru %d by %d
   pulled down (1/%d) by node %n thru %d by %d to node %n
```

Here is a pullup and a path to ground on which the pullup/pulldown ratio of lenths to widths is outside of the lowlimit, highlimit range. The (1/%d) will be (1/1) for a level restored pulldown, (1/2) for a stored charge pulldown, and (1/3) for an input pulldown. The L/W of a stored charge pulldown is multiplied by 2 before it is added to the L/W sum on the pulldown path to Gnd. The L/W of an input pulldown is multiplied by 3 before it is added to the L/W sum on the pulldown path to Gnd. This is to account for the ratio differences required by different thresholds. The actual lengths and widths of the pullup and each pulldown transistor are given.

```
Value not used : %n
```

A pulled up node is never used on a gate. Maybe it’s an output that you are not using from a standard cell.

A4. Design Rule Checking Module

The Design Rule Checking Module, MagicDRC, checks the layout for conformance with Mead/Conway [Mead/Conway] design rules for NMOS. MagicDRC generates two files -

.drclog file that lists all design rule violations, in *show* syntax;

.drv file that is a legitimate CIF file containing polygons on two layers to indicate the location and extent of design rule violations.

The *show* and *more* commands described in Section 3.4 are useful for examining these files.

A4.1 Initialization

First get and initialize Magic for DRC as described under 2.3 "Where to find Magic", either by retrieving and executing

[Indigo]<DA>DStar>MagicDRC.cm

or by getting everything from

[Indigo]<DA>DStar>UpdateMagic.cm

A4.2 Command file usage

Two .do files are provided for checking the design rules of a single layout, one for CIF input and one for Icarus:

CIFtoDRC.do
ICtoDRC.do

These can be run using Do.run and answering the prompts for parameters.

A4.3 Interactive usage

Interactive use of Magic for DRC is useful for extracting a piece of the layout, and for avoiding the need to restart Magic and read the layout again in order to examine the results.

drc
Usage: **drc**
Select the DRC module for subsequent output.. e.g.

```
drc
```

output
Usage: <filename> **output**
DRC the circuit represented by that part of the layout currently in the viewing area. Two files are generated: filename.drclog, which contains a log of all design rule violations; and filename.drv that is a legitimate CIF file containing polygons on two layers to indicate the location and extent of design rule violations. e.g.

```
(chip) output
```

will generate chip.drclog and chip.drv.

A convenient way of viewing the results is to use the *show* and *more* commands described in Section 3.4. Do this by adding the .drv file into the current data base, and then *show*ing the .drclog file.

A5. Versatec Plotting Module

The Versatec Plotting Module, MagicVersatec, generates a ‘‘.bits’’ format file for subsequent printing using the ‘‘Oliver’’ program at the Versatec plotter. The layout is split into separate strips that are generated on separate files.

A5.1 Initialization

First get and initialize Magic for Versatec output as described under 2.3 "Where to find Magic", either by retrieving and executing

[Indigo]<DA>DStar>MagicVersatec.cm

or by getting everything from

[Indigo]<DA>DStar>UpdateMagic.cm

A5.2 Command file usage

Two .do files are provided for converting a single layout to Versatec output, one for CIF input and one for Icarus:

CIFtoVersatec.do
ICtoVersatec.do

These can be run using Do.run and answering the prompts for parameters, or by providing parameters on the command line in the form:

Do CIFtoVersatec <infile><outfile><number of strips>

Two optional parameters are supported, to give the indices of the first and last stripts to be generated. These can be given only from the command line which then takes the form:

Do CIFtoVersatec <infile><outfile><number of strips> ^
<1st strip><last strip>

This will fit the entire layout to the <number of strips> requested, and will generate the strips between <1st strip> and <last strip>, inclusive. Any of the parameters can be out of range in which case the maximum or minimum permissible parameter value is used. Each strip is output to a separate bits file, the names being of the form: filename-i.bits, where i is the strip number.

Caution should be taken to insure enough disk space is present when generating a large ‘‘.bits’’ file; a typical plot 36" wide by 23" tall takes about 2000 Alto pages, depending on the complexity, which affects the effectiveness of the run-length coding in the file. If the filename starts with a fileserver name (e.g. [ivy]filename) each strip is written to the file server after it is generated. This minimizes the need for local file space for the bits files. Be warned that you must be logged in to the executive with a valid password before using this option.

A5.3 Interactive usage

Interactive use of Magic for generating Versatec output is useful for experimenting with the fit of the layout to a given set of strips, for extracting a piece of the layout for printing, and for superimposing several layouts for printing.

versatec
Usage: <number of strips> versatec
Select the Versatec module for subsequent output. While the versatec module is selected

the outlines of the strips are superimposed on the view. e.g.

```
2 versatec
```

will select versatec for output on 2 strips. Versatec can be reselected for adjusting the number of strips.

output
Usage: <1st strip><last strip><filename> **output**
Output the current contents of the viewing area to the versatec module to generate the strips between those indicated in the first two parameters, inclusive. Either of the parameters can be out of range in which case the maximum or minimum permissible parameter value is used. Each strip is output to a separate bits file, the names being of the form: filename-i.bits, where i is the number of the strip.

```
1 1000 (chip) output
```

will output all strips (assuming the selected number is smaller than 1000) onto files chip-1.bits, chip-2.bits, ... etc.

If the filename starts with a fileserver name each strip is written to the file server after it is generated. This minimizes the need for local file space for the bits files. Be warned that you must be logged in to the executive with a valid password before using this option. e.g.

```
1 1000 ([Ivy]chip) output
```

setstipple
Usage: <layername><stipple array> **setstipple**
Set the stipple vector of the given layer to the values in the stipple array, which must contain 4 numbers. e.g.

```
(NM) [ 12345 5432 12345 5432 ] setstipple
```

A5.4 Using Oliver to plot

The procedure for plotting a ‘‘.bits’’ file is: Spin up the ‘‘current press’’ Alto disk and the public Trident pack on the Versatec Alto, and place the Versatec plotter on-line. Boot and type ‘‘@OliverServer<CR>’’ On your Dolphin or Dorado workstation (on which you ran Magic-Versatec) use EFTP.run (available from [Maxc]<Alto>) to send the ‘‘.bits’’ file to the Versatec Alto (the VLSI System Design Area has chosen the name ‘‘Vice’’ as in ’’Vice-Versatec’’ for their Alto with net address 3#166#). Oliver should begin plotting the file as soon as it is transferred.

A6. Mann Output Module

The Mann Output Module generates a representation of the contents of the viewing area in in Mann 3000 format. Both English and Metric formats are supported. Currently only input files containing only rectangles aligned with the coordinate axes are supported.

A6.1 Initialization

First get and initialize Magic for Mann output as described under 2.3 "Where to find Magic", either by retrieving and executing

[Indigo]<DA>DStar>MagicMann.cm

or by getting everything from

[Indigo]<DA>DStar>UpdateMagic.cm

A6.2 Command file usage

Two .do files are provided for converting a single layout to Mann output, one for CIF input and one for Icarus:

CIFtoMann.do
ICtoMann.do

These can be run using Do.run and answering the prompts for parameters, or by providing parameters on the command line in the form:

Do CIFtoMann <infile><outfile><units>

<units> must be one of the keywords *english* or *metric*. This will generate a Mann directory file outfile.mann, and a set of files of the form outfile-i.mann, where i is the layer number. Correspondence between layer numbers and layer names is given on the typescript.

Dimensions in the input file are taken to be dimensions on the wafer, after reduction of the reticle by a factor of 10. For english units no attempt is made to scale from CIF units (hundredths of a micron) to english units. Rather, 1 CIF unit will come out as 0.01 mils on the wafer.

A6.3 Interactive usage

Interactive use of Magic for generating Mann output is useful for extracting a piece of the layout for conversion, and for superimposing several layouts for printing.

mann
Usage: <units> **mann**
Select the Mann module for subsequent output. The single parameter <units> must be one of the keywords *english* or *metric*. e.g.

```
english mann
```

will select the Mann output module in english units.

output
Usage: <filename> **output**
Convert the current contents of the viewing area to Mann format.

This will generate a Mann directory file filename.mann, and a set of files of the form filename-i.mann, where i is the layer number. Correspondence between layer numbers and layer names is given on the typescript. e.g.

```
(chip) output
```

will generate the directory file chip.mann, and the layer files chip-1.mann, chip-2.mann, ... etc.

A7. Ramtek Plotting Module

The Ramtek Plotting Module, MagicRamtek, generates a ‘‘.ram’’ format file for subsequent printing using the ‘‘Aries’’ program at the Ramtek plotter. The layout is split into separate strips that are generated on separate files.

A7.1 Initialization

First get and initialize Magic for Ramtek output as described under 2.3 "Where to find Magic", by retrieving and executing

[Indigo]<DA>DStar>UpdateMagic.cm

A7.2 Command file usage

Two .do files are provided for converting a single layout to Ramtek output, one for CIF input and one for Icarus:

CIFtoRamtek.do
ICtoRamtek.do

These can be run using Do.run and answering the prompts for parameters, or by providing parameters on the command line in the form:

Do CIFtoRamtek <infile><outfile><number of strips>

Two optional parameters are supported, to give the indices of the first and last stripts to be generated. These can be given only from the command line which then takes the form:

Do CIFtoRamtek <infile><outfile><number of strips> ^
<1st strip><last strip>

This will fit the entire layout to the <number of strips> requested, and will generate the strips between <1st strip> and <last strip>, inclusive. Any of the parameters can be out of range in which case the maximum or minimum permissible parameter value is used. Each strip is output to a separate ‘‘.ram’’ file, the names being of the form: filename-i.ram, where i is the strip number.

Caution should be taken to insure enough disk space is present when generating a large ‘‘.ram’’ file; a typical plot 15" wide by 20" tall takes about 400 Alto pages, depending on the complexity, which affects the effectiveness of the run-length coding in the file. If the filename starts with a fileserver name (e.g. [ivy]filename) each strip is written to the file server after it is generated. This minimizes the need for local file space for the ram files. Be warned that you must be logged in to the executive with a valid password before using this option.

A7.3 Interactive usage

Interactive use of Magic for generating Ramtek output is useful for experimenting with the fit of the layout to a given set of strips, for extracting a piece of the layout for printing, and for superimposing several layouts for printing.

ramtek
Usage: <number of strips> ramtek
Select the Ramtek module for subsequent output. While the ramtek module is selected the outlines of the strips are superimposed on the view. e.g.

```
2 ramtek
```

will select ramtek for output on 2 strips. Ramtek can be reselected for adjusting the number of strips.

output

Usage: <1st strip><last strip><filename> **output**
Output the current contents of the viewing area to the ramtek module to generate the strips between those indicated in the first two parameters, inclusive. Either of the parameters can be out of range in which case the maximum or minimum permissible parameter value is used. Each strip is output to a separate ram file, the names being of the form: filename-i.ram, where i is the number of the strip.

```
1 1000 (chip) output
```

will output all strips (assuming the selected number is smaller than 1000) onto files chip-1.ram, chip-2.ram, ... etc.

If the filename starts with a fileserver name each strip is written to the file server after it is generated. This minimizes the need for local file space for the ram files. Be warned that you must be logged in to the executive with a valid password before using this option. e.g.

```
1 1000 ([Ivy]chip) output
```

setstipple

Usage: <layername><stipple array> **setstipple**
Set the stipple vector of the given layer to the values in the stipple array, which must contain 4 numbers in [0..15]. e.g.

```
(NM) [ 0 6 9 15 ] setstipple
```

Each of the four numbers represents a 2x2 array, in Black, Cyan, Magenta, and Yellow. respectively, with bits weighted as shown:

```
2  1
8  4
```

The areas of the plot occupied by the given layer are tiled with repetitions of the pattern so defined. The example above thus yields a red and green checkerboard.

A7.4 Using Aries to plot

A simple interface program, Aries.run, is available from [Indigo]<DA>Magic> to drive the printer from ".ram" format color-bitmap files. (People wishing to write other software to generate ".ram" files will find the format documented in [Indigo]<DA>DStar>RamFileFormat.Press.) An Alto II, with network name ‘‘Color’’ is interfaced with the 4100 in the VLSI Purple Lab. A disk labelled ‘‘Pasco Ramtek’’ is maintained with a current copy of Aries.Run.

Color is now running in Server mode for remote, unattended operation. It accepts multiple ‘‘.ram’’ files via the Alto Pup FTP protocol, and prints them when the connection is closed. Files are deleted after they are printed. To use, you need only type (to your Alto executive),

```
FTP Color store File1.ram File2.ram ...
```

Color does not spool files and does not respond to a request for connection if it is busy printing. If your connection attempt fails, Color may be down or just busy, and you should

visit it or try later.

With Magic, the transmission and printing may be integrated with the file generation process. Simply tell Magic,

```
<number of strips> ramtek 1 99 ([Color]temp) output
```

and your design will be transmitted and printed, strip-by-strip, as it is generated.