TABLE OF CONTENTS

LIST OF FIGURES ......................................................................................... 3
PREFACE .......................................................................................................... 5

1. Ubuntu ...................................................................................................... 6
   1.1 About Ubuntu ....................................................................................... 6
   1.2 Download Ubuntu ............................................................................... 6
   1.3 Install Ubuntu (Different methods to install Ubuntu) ......................... 8

2. MAGIC ....................................................................................................... 13
   2.1 What Is Magic? ................................................................................... 13
   2.2 About Magic? ..................................................................................... 13
   2.3 Why Magic? ....................................................................................... 14
   2.4 How Is Magic Different? ................................................................. 15
   2.5 What Is IRSIM? ................................................................................ 15

3. INSTALLATION OF MAGIC ................................................................... 17
   3.1 Installation of Magic is of two ways ............................................... 17
      3.1.1 Download Magic from the Official website and then install it ....... 17
      3.1.2 Install Magic through sudo commands ...................................... 18
   3.2 System Requirements ..................................................................... 21
   3.3 Installation ....................................................................................... 21

4. IRSIM ....................................................................................................... 23
   4.1 What Is IRSIM? ................................................................................ 23
   4.2 How To Install IRSIM? .................................................................... 24
# COURSEWARE DEVELOPMENT FOR OPEN SOURCE MAGIC VLSI TOOL

## 5. GETTING STARTED

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Running Magic</td>
<td>25</td>
</tr>
<tr>
<td>5.2 The Box and the Cursor</td>
<td>25</td>
</tr>
<tr>
<td>5.3 Invoking Commands</td>
<td>27</td>
</tr>
</tbody>
</table>

## 6. BASIC PAINTING AND SELECTION

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 Cells and Paint</td>
<td>29</td>
</tr>
<tr>
<td>6.2 Painting and Erasing</td>
<td>29</td>
</tr>
<tr>
<td>6.3 Undo</td>
<td>30</td>
</tr>
<tr>
<td>6.4 The Selection</td>
<td>30</td>
</tr>
<tr>
<td>6.5 Operations on the Selection</td>
<td>31</td>
</tr>
<tr>
<td>6.6 Labels</td>
<td>31</td>
</tr>
</tbody>
</table>

## 7. DESIGN-RULE CHECKING

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1 Continuous Design-Rule Checking</td>
<td>33</td>
</tr>
<tr>
<td>7.2 Getting Information about Errors</td>
<td>33</td>
</tr>
<tr>
<td>7.3 Errors in Hierarchical Layouts</td>
<td>35</td>
</tr>
<tr>
<td>7.4 Turning the Checker Off (Optional)</td>
<td>35</td>
</tr>
<tr>
<td>7.5 Porting Layouts from Other Systems</td>
<td>37</td>
</tr>
</tbody>
</table>

## 8. CIRCUIT EXTRACTION

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1 Introduction</td>
<td>38</td>
</tr>
<tr>
<td>8.2 Basic Extraction</td>
<td>38</td>
</tr>
<tr>
<td>8.3 Feedback: Errors and Warnings</td>
<td>40</td>
</tr>
</tbody>
</table>

## 9. AN INVERTER DESIGN-EXAMPLE

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
</table>

## 10. DESIGN OF BASIC LAYOUTS

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
</table>

## 11. CONCLUSION

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
</table>

REFERENCES

53
LIST OF FIGURES

Figure 1: Direct Link................................................................. 6
Figure 2: Downloading using Bit Torrent .................................... 7
Figure 3: Software Package in CD.............................................. 7
Figure 4: Ubuntu Start up......................................................... 8
Figure 5: Welcome Window..................................................... 8
Figure 6: Setup Preparation ....................................................... 9
Figure 7: Installation Type ....................................................... 9
Figure 8: Erase Disk and Installation ....................................... 10
Figure 9: Keyboard Layout..................................................... 10
Figure 10: User Information................................................... 11
Figure 11: WMWare OS ........................................................... 11
Figure 12: New Virtual Machine Wizard ................................ 12
Figure 13: Virtual Machine Wizard Setup ............................... 12
Figure 14: MAGIC Home Page .............................................. 17
Figure 15: MAGIC VLSI Field Option .................................... 17
Figure 16: Download Source.................................................. 18
Figure 17: Install TK ............................................................... 18
Figure 18: TK Version and Update ...................................... 19
Figure 19: Install MAGIC....................................................... 19
Figure 20: Extraction during Installation .............................. 20
Figure 21: Install m4 Package ............................................... 20
Figure 22: m4 Extraction ....................................................... 20
Figure 23: IRSIM Installation ............................................... 24
Figure 24: Layout Window .................................................... 26
Figure 25: Command Window ............................................. 26
Figure 26: Box .................................................................. 27
Figure 27: NWell and Pdiff .................................................. 43
Figure 28: Pdc and ndiff ....................................................... 43
Figure 29: Poly ................................................................ 44
COURSEWARE DEVELOPMENT FOR OPEN SOURCE MAGIC VLSI TOOL

Figure 30: Final Layout of Inverter ..................................................... 44
Figure 31: Open Design ................................................................. 45
Figure 32: DRCs ............................................................................ 45
Figure 33: Extraction ................................................................. 45
Figure 34: Extract to Spice ....................................................... 46
Figure 35: Open Spice ............................................................. 46
Figure 36: Mismatch Result ....................................................... 46
Figure 37: Extract to Sim .......................................................... 47
Figure 38: IRSIM Report ......................................................... 47
Figure 39: In Out Signals .......................................................... 47
Figure 40: Output Report .......................................................... 48
Figure 41: Output Analyzer ...................................................... 48
Figure 42: Output Waveform ..................................................... 48
Figure 43: Inverter Layout ......................................................... 49
Figure 44: NAND Layout .......................................................... 50
Figure 45: NOR Layout ............................................................. 50
Figure 46: Transmission Gate .................................................... 51
PREFACE

This manual is a part of our project funded by NMEICT (Nationa Mission on Education Through Information Communication Technology), Government of India. The fund was to develop the course material for open source VLSI and Embedded System tools. It is a laboratory course. VLSI tools are costly and have to renew in every three/four years thus you need recurring money to maintain the lab. It is not possible to many private engineering colleges of the country. However, learning and using VLSI laboratory is the need of the day. Few VLSI tools are available open source. Most of these tools run in linux platform. Using this tool without a proper manual containing installation process, tool download, and its use is not easy task. Keeping this in mind, we have developed the 35 Hours Video lecture (attached with this manual in a DVD). It contains how to install Ubuntu, VLSI tool (mainly MAGIC and IRSIM) and how to design simple circuit etc. This manual also helps to learn the tool without the help of any instructors having simple electronics devices and circuits knowledge. All the Video Lectures and soft copy of this manual is available at http://www.nist.edu/Projects/NMEICT/index.html.

For this, we would first like to thank NMEICT funding without which we could not have brought this manual and Video. We would like to thank members of the Steering Committee of NMEICT who has reviewed the project. We particularly would like to thank Prof Kannan of IIT, Bombay and his group for their valuable suggestions, kind words and encouragement to work on this project. Without Prof Kannan initiation, we would have not received this fund and would have been nowhere of developing this Video Course. We want to thank Director, NIST and its administration for providing us with an excellent environment to make us work efficiently and developing the Videos lecture.

Dr. Ajit Kumar Panda
Abhyarthena Bisoyi
Goutam Padhy
Saswat Mohanty
Mukesh Kumar Sukla
National Institute of Science and Technology, Palur Hills, Berhampur - 761008
1. Introduction to Ubuntu

1.1 About Ubuntu

Ubuntu is a Debian-based Linux operating system, with Unity (a graphical user interface) as its default desktop environment. It is based on free software. Ubuntu is an ancient African word meaning 'humanity to others'. It also means 'I am what I am because of who we all are'. The Ubuntu operating system brings the spirit of Ubuntu to the world of computers.

1.2 Download Ubuntu

There are three ways to download Ubuntu:

1. Direct link
   http://www.ubuntu.com/download/ubuntu/download (Press Ctrl and then Click)
2. **Bit torrent**

http://www.ubuntu.com/download/ubuntu/alternative-download (Press Ctrl and then Click)

![Figure 2: Downloading using Bit Torrent](image)

3. **Buy CD(s)**

http://www.ubuntu.com/download/ubuntu/cds (Press Ctrl and then Click)

![Figure 3: Software Package in CD](image)
1.3 Install Ubuntu (Different methods to install Ubuntu)

There are two ways to install Ubuntu:

1. **Direct Install.**
   
   ✓ Turn on the computer and insert the Ubuntu installation CD into the CD-ROM. Ensure that the BIOS is set to boot from the CD. Wait for the BIOS to read the CD and recognize the operating system. If everything is done properly you'll be able to see the Ubuntu start up screen as shown in figure.

   ![Figure 4: Ubuntu Start up](image)

   ✓ Wait for Ubuntu to start up. Once the loading is completed, you are presented with the dialogue box as shown in figure, where you need choose to install Ubuntu. You can choose the installation language from the options shown on the left side in the figure.

   ![Figure 5: Welcome Window](image)
Ubuntu checks if your system has access to the internet, connected to the power supply (laptop) and has at least 4.4 GB hard disk space. You will also be required to choose whether you want to install third party software and updates during the installation.

![Figure 6: Setup Preparation](image)

You are now presented with options to choose the type of installation you would like to perform. You can (a) install Ubuntu alongside other operating systems (if you have any personal data, they will not be erased), (b) erase disk and install Ubuntu as the only operating system on your system or (c) something else.

![Figure 7: Installation Type](image)
If you have no other operating system installed on your system, then you will only have the last two options. In this section which deals with Ubuntu as the only operating system, you need to choose the second option which removes any other operating system such as Windows XP, Vista or 7 and automatically does all the hard disk partitioning for you.

The previous step was the hardest. Now it is basically entering some personal information. Next you need to choose your keyboard layout. Ubuntu will automatically detect your keyboard layout. However, you can change the keyboard layout to match the local language settings.
You need to enter the place where you live. This is required to automatically get the time zone and set the correct time.

In the last step you are required to set the username, password and other options like requiring a password to login in.

2. Creating Virtual machine and then installing Ubuntu inside it.
For that you need to install VMW in your system first, which is a Virtual Machine.
VMW is a software, which creates a virtual environment for some other operating system which in this case is Ubuntu.

VMWare Operating system.
When you click on “Create a New Virtual Machine”, the username and password are to be set for the Virtual operating system installed in VMWare.

![New Virtual Machine Wizard](image1)

Figure 12: New Virtual Machine Wizard

The following figure shows snapshot of creating a virtual disk inside VMWare and installing Ubuntu inside it.

![Virtual Machine Wizard Setup](image2)

Figure 13: Virtual Machine Wizard Setup

And then the same process continues like in direct installation.
Thus, Ubuntu should be installed first. If your machine has already Ubuntu, skip this section and go to the next section. Next section deals with installation of MAGIC and IRSIM.
2. MAGIC

2.1 What Is Magic?

Magic is a Very-large-scale integration layout tool originally written by John Ousterhout and his graduate students at UC Berkeley during the 1980’s. The current version is 8.0, but 6.x is still widely used.

2.2 About Magic?

Since its origination in 1980’s, Magic continues to be popular because it is free (Berkeley open-source license), easy to use, and easy to expand for specialized tasks. The current version is 8.0, but 6.x is still widely used. Due largely in part to its liberal Berkeley open-source license, magic has remained popular with universities and small companies. The open-source license has allowed VLSI engineers with an inclination toward programming to implement clever ideas and help magic staying abreast of fabrication technology. However, it is the well thought-out core algorithms which lend to magic the greatest part of its popularity. Magic is widely cited as being the easiest tool to use for circuit layout, even for people who ultimately rely on commercial tools for their product design flow.

Magic is different from other layout editors. The most important difference is that Magic is more than just a color painting tool: it understands quite a bit about the nature of circuits and uses this information to provide you with additional operations. For example, Magic has built-in knowledge of layout rules; as editing is done, it continuously checks for rule violations. Magic also knows about connectivity and transistors, and contains a built-in hierarchical circuit extractor. Magic has a plow operation as well that can be used to stretch or compact cells. Lastly, Magic has routing tools that can be used to make the global interconnections in your circuits.

Magic is based on the Mead-Conway style of design. This means that it uses simplified design rules and circuit structures. The simplifications make it easier for you to design circuits and permit Magic to provide powerful assistance that would not
be possible otherwise. Magic features real-time design rule checking, something that some costly commercial VLSI design software packages don't feature. Magic implements this by counting distance using Manhattan distance rather than Euclidean distance, which is much faster to compute.

Magic currently runs under Linux, although versions exist for DOS, OS/2, and other operating systems. Magic is frequently used in conjunction with IRSIM and other simulation programs.

Magic includes several facilities traditionally contained in separate batch-processing programs. Magic incorporates expertise about design rules, connectivity, and routing directly into the layout editor and uses this information to provide several unusual features. They include a continuous design-rule checker that operates in background and maintains an up-to-date picture of violations; a hierarchical circuit extractor that only re-extracts portions of the circuit that have changed; an operation called plowing that permits interactive stretching and compaction; and a suite of routing tools that can work under and around existing connections in the channels.[1]

MAGIC uses what is called a "lambda-based" design system. Lambda is a scale factor used to define the minimum technology geometry increment on the die, which we see represented on the CRT as a small "square". In the VLSI world, layout items are aligned to a grid which represents a basic unit of spacing determined by the contents of the particular <technology file> invoked when the MAGIC software is started.

The features that magic offers, in spite of being free of cost makes it an ideal tool for using in educational institutions. Keeping in mind the need for a open source tool, this project involves study of Magic tool and preparing a manual that would be helpful in induction of Magic at educational level.

2.3 Why Magic?

Magic offers the following advantages to its users:

- It is free (Berkeley open-source license),
- Easy to use
- Easy to expand for specialized tasks.
2.4 How Is Magic Different?

- Magic is more than just a color painting tool.
- It understands quite a bit about the nature of circuits and uses this information to provide us with additional operations.
- It knows about connectivity and transistors, and contains a built-in hierarchical circuit extractor.
- It has a plow operation that can be used to stretch or compact cells.
- It has routing tools that can be used to make the global interconnections in your circuits.
- Magic is based on the Mead-Conway style of design (uses simplified design rules and circuit structures)
- Magic features real-time design rule checking (Magic implements this by counting distance using Manhattan distance rather than Euclidean distance, which is much faster to compute.)
- Magic currently runs under Linux, although versions exist for DOS, OS/2, and other operating systems.
- Magic is frequently used in conjunction with IRSIM and other simulation programs.

2.5 What Is IRSIM?

IRSIM is a tool for simulating digital circuits. It is a "switch-level" simulator; that is, it treats transistors as ideal switches. Extracted capacitance and lumped resistance values are used to make the switch a little bit more realistic than the ideal, using the $RC$ time constants to predict the relative timing of events.

IRSIM shares a history with magic, although it is an independent program. Magic was designed to produce and IRSIM to read, the ".sim" file format, which is largely unused outside these two programs. IRSIM was developed at Stanford, while Magic was developed at Berkeley. Parts of Magic were developed especially for use with
COURSEWARE DEVELOPMENT FOR OPEN SOURCE MAGIC VLSI TOOL

IRSIM, allowing IRSIM to run a simulation in the "background" (i.e., a forked process communicating through a pipe), while displaying information about the values of signals directly on the VLSI layout.

For "quick" simulations of digital circuits, IRSIM is still quite useful for confirming basic operation of digital circuit layouts. The addition of scheduling commands ("at", "every", "when", and "whenever") put IRSIM into the same class as Verilog simulators. It is much easier to write complicated test bench simulations using Tcl and IRSIM. IRSIM can be used to validate the digital parts of several production chips at MultiGiG, including the simulation of analog behavior such as PLL locking. IRSIM version 9.5 was a long-standing and stable version that corresponded to the relatively stable Magic version 6.5. When magic was recast in a Tcl/Tk interpreter framework (versions 7.2 and 7.3), IRSIM could no longer operate as a background process. However, it was clear that if IRSIM could also be recast in the same Tcl/Tk interpreter framework, the level of interaction between it and Magic would be greatly increased.
3. INSTALLATION OF MAGIC

Magic currently runs under Linux, although versions exist for DOS, OS/2, and other operating systems. The next section describes about the installation of magic in Linux.

3.1 Installation of Magic is of two ways

3.1 Download Magic from the Official website and then install it.
3.2 Install Magic through Sudo commands.

3.1.1 Download Magic from the Official website and then install it

To download Magic we need to follow the following steps.

**Step -1:**
www.opencircuitdesign.com/magic. (Press Ctrl and then Click)

**Figure 14: MAGIC Home Page**

**Step -2:**
Go to the left side of the page and click on download. (As shown)

**Figure 15: MAGIC VLSI Field Option**
Then you move to download page (As shown)

Then press on the “.magic-8.0.196.tgz” which is of 3.7Mb.

You will start downloading the file.

After you download, you can just install it as usually we install an application in Ubuntu.

3.1.2 Install Magic through sudo commands

There are some steps to be followed to install Magic in Ubuntu

**Step -1:**

Type the command: sudo apt-get install tk
After the “tk” is installed through sudo command, then the following widow is displayed

![Figure 18: TK Version and Update](image)

**Step -2:**

Then we need to install magic by command: sudo apt-get install magic

![Figure 19: Install MAGIC](image)

During the installation, it asks for a permission which has to be permitted (for using 11.1Mb of disk space) by pressing Y/N.

You need to press Y.

After you press Y you will get the following things.

Now the installation of Magic is complete.
Now install the next package “m4”

The command for the installing m4 package is: sudo apt-get install m4
The Important areas are highlighted in the figure above.

During the installation of the package, it asks for the password.

You need to enter the admin password of the Ubuntu Operating system.

After the password is entered, it asks for the permission to install the package (for using an additional space of 404Kb in the hard disk).

Then the M4 package will be installed successfully, and it shows the message.

### 3.2 System Requirements

Although specifically for Ubuntu 9.1 Linux, they provide a good summary of the packages required on all systems. Different distributions of Linux may or may not come with all of the following packages, and missing ones will need to be installed.

In order to compile Magic on a vanilla installation of Ubuntu 9.10, the following additional packages are needed:

- **M4**
  
  `$ sudo apt-get install m4`

- **C-shell**
  
  `$ sudo apt-get install csh`

- **Xlib.h**
  
  `$ sudo apt-get install libx11-dev`

- **Tcl/Tk**
  
  `$ sudo apt-get install tcl-dev tk-dev`

### 3.3 Installation

- **Compile From Source**
  
  Enter the following commands in the Linux console
  
  `% ./configure`

  `% make`

  `% make install`
From Executables

This is much easier way of installing magic.

Add the following repositories to the list of your sources.
(System>Admin>Software Sources>3rd Party)

deb http://ppa.launchpad.net/aanjhan/ubuntu/hardy main
deb-src http://ppa.launchpad.net/aanjhan/ubuntu/hardy main

Then enter the following command in the Linux console

sudo apt-get install magic

The next section describes about the installation of magic in windows.
4. IRSIM

Magic currently runs under Linux, although versions exist for DOS, OS/2, and other operating systems. The next section describes about the installation of magic in Linux.

4.1 What Is IRSIM?

IRSIM is a tool for simulating digital circuits. It is a "switch-level" simulator; that is, it treats transistors as ideal switches. Extracted capacitance and lumped resistance values are used to make the switch a little bit more realistic than the ideal, using the $RC$ time constants to predict the relative timing of events.

IRSIM shares a history with magic, although it is an independent program. Magic was designed to produce and IRSIM to read, the ".sim" file format, which is largely unused outside of these two programs. IRSIM was developed at Stanford, while Magic was developed at Berkeley. Parts of Magic were developed especially for use with IRSIM, allowing IRSIM to run a simulation in the "background" (i.e., a forked process communicating through a pipe), while displaying information about the values of signals directly on the VLSI layout.

For "quick" simulations of digital circuits, IRSIM is still quite useful for confirming basic operation of digital circuit layouts. The addition of scheduling commands ("at", "every", "when", and "whenever") put IRSIM into the same class as Verilog simulators. It is much easier to write complicated testbench simulations using Tcl and IRSIM. IRSIM can be used to validate the digital parts of several production chips at MultiGiG, including the simulation of analog behavior such as PLL locking. IRSIM version 9.5 was a long-standing and stable version that corresponded to the relatively stable Magic version 6.5. When magic was recast in a Tcl/Tk interpreter framework (versions 7.2 and 7.3), IRSIM could no longer operate as a background process. However, it was clear that if IRSIM could also be recast in the same Tcl/Tk interpreter framework, the level of interaction between it and Magic would be greatly increased.
4.2 How To Install IRSIM?

This is the command to install magic in Ubuntu:

Sudo apt-get install irsim

Figure 23: IRSIM Installation
5. GETTING STARTED

5.1 Running Magic

Starting up Magic is usually pretty simple. Just log in and, if needed, start up your favorite window system. Then type the shell command

```
magic tut1
```

Tut1 is the name of a library cell that you will play with in this tutorial. At this point, several colored rectangles should appear on the color display along with a white box and a cursor. A message will be printed on the text display to tell you that tut1 isn’t writable (it’s in a read-only library), and a “>” prompt should appear.

Note: in the tutorials, when you see things printed in boldface, for example, magic tut1 from above, they refer to things you type exactly, such as command names and file names. These are usually case sensitive (A is different from a). When you see things printed in italics, they refer to classes of things you might type. Arguments in square brackets are optional. For example, a more complete description of the shell command for Magic is

```
magic [file]
```

You could type any file name for file, and Magic would start editing that file. It turns out that tut1 is just a file in Magic’s cell library. If you didn’t type a file name, Magic would load a new blank cell [1]

If things didn’t happen as they should have when you tried to run Magic, any of several things could be wrong. If a message of the form “magic: Command not found” appears on your screen it is because the shell couldn’t find the Magic program. The most stable version of Magic is the directory `cad/bin, and the newest public version is in `cad/new. You should make sure that both these directories are in your shell path. Normally, `cad/new should appear before `cad/bin.
5.2 The Box and the Cursor

Two things, called the box and the cursor, are used to select things on the color display. As you move the mouse, the cursor moves on the screen. The cursor starts out with a crosshair shape, but you'll see later that its shape changes as you work to provide feedback about what you're doing. The left and right mouse buttons are used to position the box. If you press the left mouse button and then release it, the box will move so that its lower left corner is at the cursor position. If you press and release the right mouse button, the upper right corner of the box will move to the cursor position, but the lower left corner will not change. These two buttons are enough to position the box anywhere on the screen. Try using the buttons to place the box around each of the colored rectangles on the screen. Sometimes it is convenient to move the box by a corner other than the lower left. To do this, press the left mouse button and hold it down. The cursor shape changes to show you that you are moving the box by its lower left corner:

![Box](image)

**Figure 26: Box**

While holding the button down, move the cursor near the lower right corner of the box, and now click the right mouse button (i.e. press and release it, while still holding down the left button). The cursor's shape will change to indicate that you are now moving the box by its lower right corner. Move the cursor to a different place on the screen and release the left button. The box should move so that its lower right corner is at the cursor position. Try using this feature to move the box so that it is almost entirely off-screen to the left. Try moving the box by each of its corners.

You can also reshape the box by corners other than the upper right. To do this, press the right mouse button and hold it down. The cursor shape shows you that you are reshaping the box by its upper right corner.
5.3 Invoking Commands

Commands can be invoked in Magic in three ways: by pressing buttons on the mouse; by typing single keystrokes on the text keyboard (these are called macros); or by typing longer commands on the text keyboard (these are called long commands). Many of the commands use the box and cursor to help guide the command.

To see how commands can be invoked from the buttons, first position the box over a small blank area in the middle of the screen. Then move the cursor over the red rectangle and press the middle mouse button. At this point, the area of the box should get painted red. Now move the cursor over empty space and press the middle button again. The red paint should go away. Note how this command uses both the cursor and box locations to control what happens.

As an example of a macro, type the g key on the text keyboard. A grid will appear on the color display, along with a small black box marking the origin of the cell. If you type g again, the grid will go away. You may have noticed earlier that the box corners didn't move to the exact cursor position: you can see now that the box is forced to fall on grid points.

Long commands are invoked by typing a colon (``;``) or semi-colon (``;``). After you type the colon or semi-colon, the ``;`` prompt on the text screen will be replaced by a `;` prompt. This indicates that Magic is waiting for a long command. At this point you should type a line of text, followed by a return. When the long command has been processed, the ``;`` prompt reappears on the text display. Try typing semi-colon followed by return to see how this works.

Each long command consists of the name of the command followed by arguments, if any are needed by that command. The command name can be abbreviated, just as long as you type enough characters to distinguish it from all other long commands. For example, :h and :he may be used as abbreviations for :help. On the other hand, :u may not be used as an abbreviation for :undo because there is another command, :upside down, that has the same abbreviation.
6. BASIC PAINTING AND SELECTION

6.1 Cells and Paint

In Magic, a circuit layout is a hierarchical collection of *cells*. Each cell contains three things: colored shapes, called *paint*, that define the circuit's structure; textual *labels* attached to the paint; and *subcells*, which are instances of other cells. The paint is what determines the eventual function of the VLSI circuit. Labels and subcells are a convenience for you in managing the layout and provide a way of communicating information between various synthesis and analysis tools.

6.2 Painting and Erasing

The two basic layout operations are painting and erasing. They can be invoked using the `:paint` and `:erase` long commands, or using the buttons. The easiest way to paint and erase is with the mouse buttons. To paint, position the box over the area you'd like to paint, then move the cursor over a color and click the middle mouse button. To erase everything in an area, place the box over the area, move the cursor over a blank spot, and click the middle mouse button. Try painting and erasing various colors. If the screen gets totally messed up, you can always exit Magic and restart it. While you're painting, white dots may occasionally appear and disappear due to violation of design rules.

It's completely legal to paint one layer on top of another. When this happens, one of three things may occur. In some cases, the layers are independent, so what you'll see is a combination of the two, as if each were a transparent colored foil. This happens, for example, if you paint metal1 (blue) on top of polysilicon (red). In other cases, when you paint one layer on top of another you'll get something different from either of the two original layers. For example, painting poly on top of ndiff produces n-transistor (try this). In still other cases the new layer replaces the old one: this happens, for example, if you paint a p-contact on top of n-transistor.
COURSEWARE DEVELOPMENT FOR OPEN SOURCE MAGIC VLSI TOOL

There is a second way of erasing paint that allows you to erase some layers without affecting others. This is the macro \(^D\) (control-D, for `Delete paint`). To use it, position the box over the area to be erased, and then move the crosshair over a splotch of paint containing the layer(s) you'd like to erase. Type ^D key on the text keyboard: the colors underneath the cursor will be erased from the area underneath the box, but no other layers will be affected. Experiment around with the ^D macro to try different combinations of paints and erases. If the cursor is over empty space then the ^D macro is equivalent to the middle mouse button: it erases everything.

You can also paint and erase using the long commands

:paint layers
:erase layers

6.3 Undo

There are probably going to be times when you'll do things that you'll later wish you hadn't. Fortunately, Magic has an undo facility that you can use to restore things after you've made mistakes. The command

:undo  (Or, alternatively, the macro u)

will undo the effects of the last command you invoked. If you made a mistake several commands back, you can type: undo several times to undo successive commands. However, there is a limit to all this: Magic only remembers how to undo the last ten or so commands. If you undo something and then decide you wanted it after all, you can undo the undo with the command

:redo  (U is a macro for this command).

6.4 The Selection

Once you have painted a piece of layout, there are several commands you can invoke to modify the layout. Many of them are based on the selection: you select one or more pieces of the design, and then perform operations such as copying, deletion, and rotation on the selected things.
“s”, is a macro for :select. The macro S (short for : select more) is just like s except that it adds on to the selection, rather than replacing it. If you accidentally type s or S when the cursor is over space, you'll select a cell. Cell selection will be discussed later.

You can also select material by area: place the box around the material you'd like to select and type a (short for : select area). This will select all of the material underneath the box. You can use the macro A to add material to the selection by area, and you can use the long command

: select [more] area layers

to select only material on certain layers.

If you'd like to clear out the selection without modifying any of the selected material, you can use the command

: select clear

or type the macro C. You can clear out just a portion of the selection by typing :select less or :select less area layers; the former deselects paint in the order that :select selects paint, while the latter deselects paint under the box (just as :select area selects paint under the box). For a synopsis of all the options to the :select command, type

: select help

6.5 Operations on the Selection

Once you've made a selection, there are a number of operations you can perform on it:

: delete :move [direction [distance]]
: copy :stretch [direction [distance]]
: upsidedown :sideways
: clockwise [degrees]

6.6 Labels

Labels are pieces of text attached to the paint of a cell. They are used to provide information to other tools that will process the circuit. Most labels are node names: they provide an easy way of referring to nodes in tools such as routers, simulators, and timing analyzers. Labels may also be used for other purposes: for example, some
labels are treated as attributes that give Crystal, the timing analyzer, information about the direction of signal flow through transistors.

The complete syntax of the :label command is

```
:label [text [position [layer]]]
```

Text must be supplied, but the other arguments can be defaulted. If text has any spaces in it, then it must be enclosed in double quotes. Position tells where the text should be displayed, relative to the point of the label. It may be any of north, south, east, west, top, bottom, left, right, up, down, center, northeast, ne, southeast, se, southwest, sw, northwest, nw. For example, if ne is given, the text will be displayed above and to the right of the label point. If no position is given, Magic will pick a position for you. Layer tells which paint layer to attach the label to. If layer covers the entire area of the label, then the label will be associated with the particular layer. If layer is omitted, or if it doesn't cover the label's area, Magic initially associates the label with the ``space`` layer, then checks to see if there's a layer that covers the whole area. If there is, Magic moves the label to that layer. It is generally a bad idea to place labels at points where there are several paint layers, since it will be hard to tell which layer the label is attached to. As you edit, Magic will ensure that labels are only attached to layers that exist everywhere under the label. To see how this works, paint the layer pdiff (brown) over the label you just created: the label will switch layers. Finally, erase poly over the area, and the label will move again. Although many labels are point labels, this need not be the case. You can label any rectangular area by setting the box to that area before invoking the label command. labeling terminals for the router (see below), and for labeling tiles used by M-pack, the tile packing program.
7. DESIGN-RULE CHECKING

7.1 Continuous Design-Rule Checking

When you are editing a layout with Magic, the system automatically checks design rules on your behalf. Every time you paint or erase, and every time you move a cell or change an array structure, Magic rechecks the area you changed to be sure you haven't violated any of the layout rules. If you do violate rules, Magic will display little white dots in the vicinity of the violation. This error paint will stay around until you fix the problem; when the violation is corrected, the error paint will go away automatically. Error paint is written to disk with your cells and will re-appear the next time the cell is read in. There is no way to get rid of it except to fix the violation.

Continuous design-rule checking means that you always have an up-to-date picture of design-rule errors in your layout. There is never any need to run a massive check over the whole design unless you change your design rules. When you make small changes to an existing layout, you will find out immediately if you've introduced errors, without having to completely recheck the entire layout. To see how the checker works, run Magic on the cell tut6a. This cell contains several areas of metal (blue), some of which are too close to each other or too narrow. Try painting and erasing metal to make the error paint go away and re-appear again.

7.2 Getting Information about Errors

In many cases, the reason for a design-rule violation will be obvious to you as soon as you see the error paint. However, Magic provides several commands for you to use to find violations and figure what's wrong in case it isn't obvious.

All of the design-rule checking commands have the form

:dr{c option

where option selects one of several commands understood by the design-rule checker.
If you're not sure why error paint has suddenly appeared, place the box around the error paint and invoke the command

:drc why

This command will recheck the area underneath the box, and print out the reasons for any violations that were found. You can also use the macro y to do the same thing. Try this on some of the errors in tut6a. It's a good idea to place the box right around the area of the error paint: :drc why rechecks the entire area under the box, so it may take a long time if the box is very large.

If you're working in a large cell, it may be hard to see the error paint. To help locate the errors, select a cell and then use the command

:drc find [nth]

If you don't provide the nth argument, the command will place the box around one of the errors in the selected cell, and print out the reason for the error, just as if you had typed :drc why. If you invoke the command repeatedly, it will step through all of the errors in the selected cell. (Remember, the “.” macro can be used to repeat the last long command; this will save you from having to retype :drc find over and over again.

A third drc command is provided to give you summary information about errors in hierarchical designs. The command is

:drc count

This command will search every cell (visible or not) that lies underneath the box to see if any have errors in them. For each cell with errors, :drc count will print out a count of the number of error areas.
7.3 Errors in Hierarchical Layouts

The design-rule checker works on hierarchical layouts as well as single cells. There are three overall rules that describe the way that Magic checks hierarchical designs:

1. The paint in each cell must obey all the design rules by itself, without considering the paint in any other cells, including its children.
2. The combined paint of each cell and all of its descendants (subcells, sub-subcells, etc.) must be consistent. If a subcell interacts with paint or with other subcells in a way that introduces a design-rule violation, an error will appear in the parent. In designs with many levels of hierarchy, this rule is applied separately to each cell and its descendants.
3. Each array must be consistent by itself, without considering any other subcells or paint in its parent. If the neighboring elements of an array interact to produce a design-rule violation, the violation will appear in the parent.

It's important to remember that each of the three overall rules must be satisfied independently. This may sometimes result in errors where it doesn't seem like there should be any. Edit the cell tut6c for some examples of this. On the left side of the cell there is a sliver of paint in the parent that extends paint in a subcell. Although the overall design is correct, the sliver of paint in the parent is not correct by itself, as required by the first overall rule above. On the right side of tut6c is an array with spacing violations between the array elements. Even though the paint in the parent masks some of the problems, the array is not consistent by itself so errors are flagged.

The three overall rules are more conservative than strictly necessary, but they reduce the amount of rechecking Magic must do. For example, the array rule allows Magic to deduce the correctness of an array by looking only at the corner elements; if paint from the parent had to be considered in checking arrays, it would be necessary to check the entire array since there might be parent paint masking some errors but not all (as, for example, in tut6c).

7.4 Turning the Checker Off (Optional)

We hope that in most cases the checker will run so quickly and quietly that you hardly know it's there. However, there will probably be some situations where the checker is irksome. This section describes several ways to keep the checker out of your hair.
If you're working on a cell with lots of design-rule violations the constant redisplay caused by design-rule checking may get in your way more than it helps. This is particularly true if you're in the middle of a large series of changes and don't care about design-rule violations until the changes are finished. You can stop the redisplay using the command

: see no errors

After this command is typed, design-rule errors will no longer be displayed on the screen. The design-rule checker will continue to run and will store error information internally, but it won't bother you by displaying it on the screen. When you're ready to see errors again, type

: see errors

There can also be times when it's not the redisplay that's bothersome, but the amount of CPU time the checker takes to recheck what you've changed. For example, if a large sub-cell is moved to overlap another large sub-cell, the entire overlap area will have to be rechecked, and this could take several minutes. If the prompt on the text screen is a “]” character, it means that the command has completed but the checker hasn't caught up yet. You can invoke new commands while the checker is running, and the checker will suspend itself long enough to process the new commands.

If the checker takes too long to interrupt itself and respond to your commands, you have several options. First, you can hit the interrupt key (often ^C) on the keyboard. This will stop the checker immediately and wait for your next command. As soon as you issue a command or push a mouse button, the checker will start up again. To turn the checker off altogether, type the command

: drc off

If you save your file and quit Magic, the information about areas to recheck will be saved on disk. The next time you read in the cell, Magic will recheck those areas, unless you've still got the checker turned off. There is nothing you can do to make Magic forget about areas to recheck :drc off merely postpones the recheck operation to a later time.
Once you've turned the checker off, you have two ways to make sure everything has been rechecked. The first is to type the command

:drc catchup

This command will run the checker and wait until everything has been rechecked and errors are completely up to date. When the command completes, the checker will still be enabled or disabled just as it was before the command. If you get tired of waiting for :drc catchup, you can always hit the interrupt key to abort the command; the recheck areas will be remembered for later. To turn the checker back on permanently, invoke the command

:drc on

7.5 Porting Layouts from Other Systems

If you are bringing into Magic a layout that was created using a different editor or an old version of Magic that didn't have continuous checking, read on. You may also need to read this section if you've changed the design rules in the technology file.

In order to find out about errors in a design that wasn't created with Magic, you must force Magic to recheck everything in the design. Once this global recheck has been done, Magic will use its continuous checker to deal with any changes you make to the design; you should only need to do the global recheck once. To make the global recheck, load your design, place the box around the entire design, and type

:drc check

If you get nervous that a design-rule violation might somehow have been missed, you can use :drc check to force any area to be rechecked at any time, even for cells that were created with Magic. However, this should never be necessary unless you've changed the design rules. If the number of errors in the layout ever changes because of a :drc check, it is a bug in Magic and you should notify us immediately.
8. CIRCUIT EXTRACTION

8.1 Introduction

The extractor computes from the layout the information needed to run simulation tools such as crystal(1) and esim(1). This information includes the sizes and shapes of transistors, and the connectivity, resistance, and parasitic capacitance of nodes. Both capacitance to substrate and several kinds of internodal coupling capacitances are extracted.

Magic's extractor is both incremental and hierarchical: only part of the entire layout must be re-extracted after each change, and the structure of the extracted circuit parallels the structure of the layout being extracted. The extractor produces a separate .ext file for each .mag file in a hierarchical design. This is in contrast to previous extractors, such as Mextra, which produces a single .sim file that represents the flattened (fully-instantiated) layout.

8.2 Basic Extraction

You can use Magic's extractor in one of several ways. Normally it is not necessary to extract all cells in a layout. To extract only those cells that have changed since they were extracted, use:

```
: load
: extract
```

The extractor looks for a .ext file for every cell in the tree that descends from the cell root. The .ext file is searched for in the same directory that contains the cell's .mag file. Any cells that have been modified since they were last extracted, and all of their parents, are re-extracted. Cells having no .ext files are also re-extracted.[2]

To try out the extractor on an example, copy all the tut8x cells to your current directory with the following shell commands:

```
cp ~cad/lib/magic/tutorial/tut8*.mag .
```
Start magic on the cell tut8a and type \texttt{:extract}. Magic will print the name of each cell (tut8a, tut8b, tut8c, and tut8d) as it is extracted. Now type \texttt{:extract} a second time. This time nothing gets printed, since Magic didn't have to re-extract any cells. Now delete the piece of poly labelled “\texttt{delete me}” and type \texttt{:extract} again. This time, only the cell tut8a is extracted as it is the only one that changed. If you make a change to cell tut8b (do it) and then extract again, both tut8b and tut8a will be re-extracted, since tut8a is the parent of tut8b.

To force all cells in the sub-tree rooted at cell \texttt{root} to be re-extracted, use \texttt{:extract all}:

\begin{itemize}
  \item \texttt{load root}
  \item \texttt{extract all}
\end{itemize}

Try this also on tut8a. You can also use the \texttt{:extract} command to extract a single cell as follows:

\texttt{:extract cell name}

will extract just the selected (current) cell, and place the output in the file \textit{name}. Select the cell tut8b (tut8b\_0) and type \texttt{:extract cell differentFile} to try this out. After this command, the file \texttt{differentFile.ext} will contain the extracted circuit for the cell tut8b. The children of tut8b (in this case, the single cell tut8d) will not be re-extracted by this command. If more than one cell is selected, the upper-leftmost one is extracted.

You should be careful about using \texttt{:extract cell}, since even though you may only make a change to a child cell, all of its parents may have to be re-extracted. To re-extract all of the parents of the selected cell, you may use

\texttt{:extract parents}

Try this out with tut8b still selected. Magic will extract only the cell tut8a, since it is the only one that uses the cell tut8b. To see what cells would be extracted by \texttt{:extract parents} without actually extracting them, use

\texttt{:extract showparents}
8.3 Feedback: Errors and Warnings

When the extractor encounters problems, it leaves feedback in the form of stippled white rectangular areas on the screen. Each area covers the portion of the layout that caused the error. Each area also has an error message associated with it, which you can see by using the: feedback command. Type: feedback help while in Magic for assistance in using the: feedback command.

The extractor will always report extraction errors. These are problems in the layout that may cause the output of the extractor to be incorrect. The layout should be fixed to eliminate extraction errors before attempting to simulate the circuit; otherwise, the results of the simulation may not reflect reality.

Extraction errors can come from violations of transistor rules. There are two rules about the formation of transistors: no transistor can be formed, and none can be destroyed, as a result of cell overlaps. For example, it is illegal to have poly in one cell overlap diffusion in another cell, as that would form a transistor in the parent where none was present in either child. It is also illegal to have a buried contact in one cell overlap a transistor in another, as this would destroy the transistor. Violating these transistor rules will cause design-rule violations as well as extraction errors. These errors only relate to circuit extraction: the fabricated circuit may still work; it just won't be extracted correctly.

In general, it is an error for material of two types on the same plane to overlap or abut if they don't connect to each other. For example, in CMOS it is illegal for p-diffusion and n-diffusion to overlap or abut.

In addition to errors, the extractor can give warnings. If only warnings are present, the extracted circuit can still be simulated. By default, only some types of warnings are reported and displayed as feedback. To cause all warnings to be displayed, use:

: extract warn all
The command

: extract warn warning

may be used to enable specific warnings selectively; see below. To cause no warnings to be displayed, or to disable display of a particular warning, use respectively

: extract warn no all or : extract warn no warning

Three different kinds of warnings are generated. The dup warning checks to see whether you have two electrically unconnected nodes in the same cell labeled with the same name. If so, you are warned because the two unconnected nodes will appear to be connected in the resulting .ext file, which means that the extracted circuit would not represent the actual layout. This is bad if you're simulating the circuit to see if it will work correctly: the simulator will think the two nodes are connected, but since there's no physical wire between them, the electrons won't! When two unconnected nodes share the same label (name), the extractor leaves feedback squares over each instance of the shared name.

It's an excellent idea to avoid labeling two unconnected nodes with the same name within a cell. Instead, use the "correct" name for one of the nodes, and some mnemonic but textually distinct name for the other nodes. For example, in a cell with multiple power rails, you might use Vdd! for one of the rails, and names like Vdd#1 for the others. As an example, load the cell tut8e. If the two nodes are connected in a higher-level cell they will eventually be merged when the extracted circuit is flattened. If you want to simulate a cell out of context, but still want the higher-level nodes to be hooked up, you can always create a dummy parent cell that hooks them together, either with wire or by using the same name for pieces of paint that lie over the terminals to be connected; see the cell tut8f for an example of this latter technique.

You can use the command

: extract unique

as an automatic means of labeling nodes in the manner described above. Run this command on the cell tut8g. A second version of this command is provided for compatibility with previous versions of Magic.
COURSEWARE DEVELOPMENT FOR OPEN SOURCE MAGIC VLSI TOOL

Running

:extract unique #

will only append a unique numeric suffix to labels that end with a “#”. Any other duplicate node names that also don't end in a “!” are flagged by feedback.

A second type of warning, fets, checks to see whether any transistors have fewer diffusion terminals than the minimum for their types. For example, the transistor type “def t” is defined in the nmos technology file as requiring two diffusion terminals: a source and a drain. If a capacitor with only one diffusion terminal is desired in this technology, the type dcap should be used instead. The fets warning is a consistency check for transistors whose diffusion terminals have been accidentally shorted together, or for transistors with insufficiently many diffusion terminals.

The third warning, labels, is generated if you violate the following guideline for placement of labels: Whenever geometry from two subcells abuts or overlaps, it's a good idea to make sure that there is a label attached to the geometry in each subcell in the area of the overlap or along the line of abutment. Following this guideline isn't necessary for the extractor to work, but it will result in noticeably faster extraction.

By default, the dup and fets warnings are enabled, and the labels warning is disabled. Load the cell tut8h, expand all its children (tut8i and tut8j), and enable all extractor warnings with :extract warn all. Now extract tut8h and all of its children with :extract, and examine the feedback for examples of fatal errors and warnings.
This section describes the step by step layout design of an inverter using magic.

```plaintext
> paint nwell
> paint pdiff

Figure 27: NWell and Pdiff

> paint pdc
> paint ndiff

Figure 28: Pdc and ndiff
>paint ndc
>paint poly
>paint m1

Figure 29: Poly

>paint nwc
>paint pwc
>label vin (similarly all input and output terminals are labeled)

Figure 30: Final Layout of Inverter
COURSEWARE DEVELOPMENT FOR OPEN SOURCE MAGIC VLSI TOOL

After the final layout of the inverter, you need save the design

**Type:** save inv4 to save

![Figure 31: Open Design](image1)

**Type:** drc find to find the drc errors.

![Figure 32: DRCs](image2)

**Type:** extract all

![Figure 33: Extraction](image3)
COURSEWARE DEVELOPMENT FOR OPEN SOURCE MAGIC VLSI TOOL

Type: ext2spice

Figure 34: Extract to Spice

Type: vi inv4.spice

Figure 35: Open Spice

After that we get this - can check W/L ratio is same or not

Figure 36: Mismatch Result
COURSEWARE DEVELOPMENT FOR OPEN SOURCE MAGIC VLSI TOOL

**Type:** ext2sim

![Figure 37: Extract to Sim](image)

**Type:** irsim inv4.sim

![Figure 38: IRSIM Report](image)

**Type:** w vin vout

and the press d

![Figure 39: In Out Signals](image)
COURSEWARE DEVELOPMENT FOR OPEN SOURCE MAGIC VLSI TOOL

Type:

l vin
s- See the output.

h vin
s – See the output.

Type: ana vin vout – to check the output in the analyzer.

The output of analyzer is
10. DESIGN OF BASIC LAYOUTS

A few basic layouts of INVERTER, NAND, NOR, TRANSMISSION GATE have been designed.

The Layouts and simulation results are as follows:

Figure 43: Inverter Layout
Figure 44: NAND Layout

Figure 45: NOR Layout
Figure 46: Transmission Gate
11. CONCLUSION

Magic includes several facilities traditionally contained in separate batch-processing programs. Magic incorporates expertise about design rules, connectivity, and routing directly into the layout editor and uses this information to provide several unusual features. They include a continuous design-rule checker that operates in background and maintains an up-to-date picture of violations; a hierarchical circuit extractor that only re-extracts portions of the circuit that have changed; an operation called plowing that permits interactive stretching and compaction; and a suite of routing tools that can work under and around existing connections in the channels. These features put magic at par with other available commercial layout tools.

Magic has the following advantages over other layout software

- Free of cost
- Easy to use
- Continuous design rule checking
- Works on multiple operating systems like windows, Linux, Solaris etc.

Owing to its numerous advantages and being open source magic software can be used in educational institutions which cannot afford costly commercial layout design tools. Magic is widely cited as being the easiest tool to use for circuit layout, even for people who ultimately rely on commercial tools for their product design flow.
REFERENCES


