AUTOMATED DESIGN RULE CHECKING

Carl E. Conrad
5th Year Microelectronic Engineering Student
Rochester Institute of Technology

ABSTRACT

A Fortran Coded Design Rule Checker was written to analyze the output file of the RIT Integrated Circuit Editor (ICE) program. The design rules for the RIT 4LEVELPMOS process have been successfully implemented for a die size of 1900 by 1900 square micrometers.

INTRODUCTION

When allowing for the design of Integrated Circuits (ICs) that are virtually unlimited in scope and creativity, a means of automated Design Rule Checking (DRC) becomes a necessity. The Integrated Circuit is made up of individual layers, such as Diffusion, Thin Oxide, Contact Cuts, and Metal. These layers combine together to form electrical circuits. A Design Rule Checker is a computer program that analyzes the layers, individually and together, to see if there are any design flaws, such as metal lines placed too close together, or contact cuts being designed too small. In keeping pace with the growing field of IC fabrication, a Design Rule Checker will need to be versatile in handling different technologies, increased layout complexity, and unusual layout designs.

The design rules that are checked are a combination of processing constraints and circuit requirements, that if violated, would result in non-functioning or defective ICs. Spacing and width checks are two examples of processing constraints. Circuit requirements would incorporate checks like: are there more than one output leads connected together, do all the power leads connect to the power rails, and are there too many input leads being driven by a single output lead (fanout).

Four basic methods of Design Rule Checking have been developed and published. They are: Corner Base, Expanding Polygon, Deterministic Finite State Automata, and Raster Scan method. The Corner Base method performs the checks on the corners of each structure (or box). At the corner, the program examines the four quadrants around the corner, and based on the presence or absence of the different layers, an error message is given. The Expanding Polygon method digitizes the individual structures into polygons. By performing logical operations and expansion/shrink steps on the layers with these polygons, error messages are sent when the polygons overlap each other. The third method, Deterministic Finite State Automata, scans a small structure through a rasterized file. A numerical state, possible an error state, is assigned to the center location. The value is determined by a "Look-up" table that is dependent

upon the presence or absence of different layers in the other locations of the scanning structure. The fourth method was actually the first method developed. The Raster Scan method rasterizes the individual layers into separate binary arrays. Then a square window (the size of the window depending upon the geometry of the rule being tested) is scanned throughout the individual arrays. The patterns of 1's and 0's, within the window, are compared to stored patterns within the program. Certain patterns flag errors, while other patterns indicate that there is nothing wrong within the window. For a more detailed explanation of any of the above methods, see references 1 through 3.

In this project, a Design Rule Checker was developed to analyze IC designs created using the RIT developed, Integrated Circuit Editor (ICE) program. The RIT 4LEVELPMOS Process was chosen for its simplicity (four processing layers: Contact Cut, Diffusion, Metal, and Thin Oxide).

The user generates his/her design in ICE, by interweaving boxes of the different processing layers together. ICE is capable of making boxes that are placed in either the X direction or the Y direction. Diagonal boxes, curves, and circles can not be designed by using ICE. When the design is complete, a CIF file is generated. (CIF stands for Caltech Intermediate Form.) From this CIF file, a raster file is generated by another computer program, called RASTER (developed by Thomas Kucmierz, another 5th Year Microelectronic Engineering Student). The RASTER program reads in the CIF file and converts the boxes into pixel elements. These elements compose a two dimensional array. Upon completion of reading the CIF file, the array is written to an output file, which is the Raster file. Each element in the raster file represents a defined amount of space on the actual IC surface. For example, the first version of these programs are breaking an entire 1900 micrometer by 1900 micrometer IC into 190 by 190 array elements, making each element representing ten square micrometers. Each element contains information of every layer that is present within that location. (The first page of the appendix has a 40 by 40 portion of an input RASTER file.)

Design Rules for version (1.0) of the DRC are:

1.) All boxes can not be smaller than 10 micrometers.
2.) All boxes must be on 10 micrometer increments of each other.
3.) Width of Diffusion boxes cannot be smaller than 30 micrometers. Resistors are often designed smaller than 30 micrometer widths. This program is a checker, so the results can be ignored since it can not interfere with ICE in generating the output files.
* 4.) Minimum spacing between Diffusion boxes is 10 micrometers.
* 5.) Width of Thin Oxide boxes cannot be smaller than 10 micrometers.
6.) Width of Metal boxes cannot be smaller than

30 micrometers.

★ 7.) Minimum spacing between Metal boxes is 10 micrometers.

8.) Contact Cuts (CC) are to have at least 10 micrometers of: Diffusion, Thin Oxide, and Metal surrounding the Contact Cut. (For example, you have a 20 micrometer Contact Cut box. There would need to be a Diffusion box of at least 40 micrometers under neath the CC, a Thin Oxide box of at least 40 micrometers the CC goes through, and a Metal box of at least 40 micrometers to cover the CC with.)

9.) For transistors, there must be at least 10 micrometers of Thin Oxide Indentation over the Diffusion area to allow for misalignment of the Thin Oxide layer. If there is no margin of error for the Thin Oxide, then all the transistors, unless perfectly aligned, would have incomplete gate regions and would not function.

★ Note: These rules may seem to be a re-instatement of the first rule, but they are presented since they maybe changed in future changes of the Design Rules, that do not change the over-all minimum width rule.

The method employed within this DRC is based on the Raster Scan, except that instead of a window check, there is a "physical" check. The program checks one design rule at a time. It scans the binary array for the presence of the layer in question. At each spot the program scans the physical area around the spot, to see if the rule was violated. If so, an Error array keeps track of the error number that is associated with the design rule being checked.

The output of this DRC is a multi-page, partitioning, of the Error output array and the transistor location array. Included are two tables listing the error code messages and a description of all the transistors. (See appendix for examples of the four output items: Error Array, Error Code table, Transistor Description table, and Transistor Location Array.)

EXPERIMENT

Figure 1 indicates the steps the program executes. There are three subroutines: WIDTHC, SPACE, and SURROU. WIDTHC checks the width of each box. It has two parameters passed into it, the layer to be checked and the minimum width length. SPACE checks the spacing between boxes. It has two parameters, the layer to be checked and the minimum spacing between the boxes. SURROU checks to see there is adequate surround boxes around the contact cuts. The two parameters passed into it are the layer to check for the surround around, and the minimum amount of surround needed.

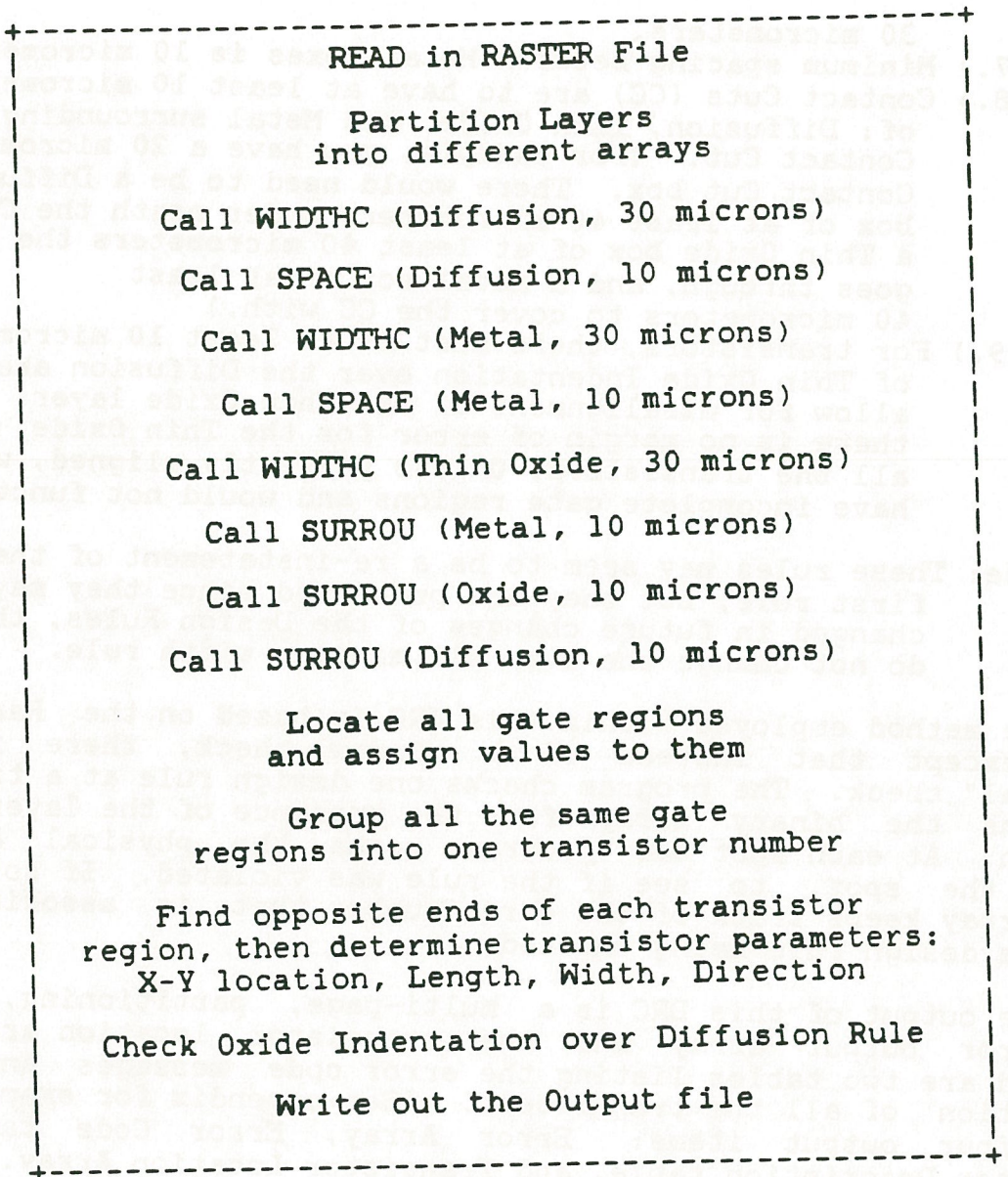

Figure 1

RESULTS & DISCUSSION

On a Sunday evening, with twenty two users on a VAX-8650 mainframe, this DRC took twenty seconds elapse time, with eleven CPU seconds. (The CPU time is the actual amount of time the computer devotes it's attention to a given task.) The input raster file was seventy two blocks of memory. The output file was two hundred seventeen blocks of memory. (Each block contains five hundred twelve bytes, which is equivalent to five hundred and twelve letters of the alphabet, that is 512 letter A's in a line.) The actual, executable machine language code took 427,000 bytes of memory. The VAX-8650 allows for over two billion bytes per task. This indicates that larger arrays (therefore larger IC's) can be analyzed.

It has been suggested that the program should generate an error output file that would be formated like a CIF file, and would contain boxes around the areas where Errors were found. This would allow the user to go back into ICE, recall his/her original work, then overlay this new CIF file over his/her original design, to easily locate where the errors are. Implementation of this process would require a complete understanding of the CIF format, and an understanding of how this DRC works, but should not be difficult to do.

If this DRC is going to be used on totally different technologies, such as NMOS or CMOS, it is recommended to make copies of the original source code and modify the copies to the technology needs. Trying to incorporate all the different technologies into one program would make the program hard to understand and would make the program very big with many duplicate parts.

CONCLUSIONS

The developed DRC worked. It was written to make future expansion easy to implement. Additional layers can be tested by adding new Error messages for the layers, and subroutine calls to check the new layers.

ACKNOWLEDGMENTS

Robert Pearson for his personal time he spent when explaining and defining the needs of the Microelectronic Engineering Department.

REFERENCES

1.) "Introduction to NMOS & CMOS VLSI Systems Design"
by: Amar Muckherjee
(The text book for: EMCR-520.)

2.) "Lyra: A New Approach to Geometric Layout Rule Checking"
by: Arnold, M. H., and J. K. Ousterhout
Proc. 19th Design Automation Conference, Las Vegas,
June 14-16, 1982 p.530

3.) "Tools for Verifying Integrated Circuit Designs"
Lambda, 4th quarter 1980, p.22-30
by: Baker, C., and C. Terman
from: "VLSI Design" periodical, Vol 1, No.3, 3rd Quarter '80