# A DESIGN TOOL SOFTWARE INTERFACE

Thomas C. Kucmierz
5th Year Microelectronic Engineering Student
Rochester Institute of Technology

## ABSTRACT

Currently, the Integrated Circuit Editor (ICE), a CAD I.C. design tool used for layouts at RIT, lacks any design rule checking simulation capabilities. This project involved writing a program that would translate the output file from ICE in the CalTech Intermediate Format (CIF) into a format that would be readable by other software tools, such as design rule checking and circuit node extraction programs.

## INTRODUCTION

The Integrated Circuit Editor (ICE) is an in-house CAD package used to layout I.C. designs at RIT. ICE stores the designs in the CalTech Intermediate Format (CIF) file structure. This file structure is set up to define the process layers and the individual components within each layer. The components are defined as grouped sequences of boxes along with scaling factors and positional information. [1] A sample CIF file is shown in Figure 1.

```
Comment ICE cif version 1.0;
DS 4 100/1;
   L OXIDE;
   Box   950 950 280,280;
DF;
DS 2 100/1;
   L CC;
   Box   950 950 220,220;
DF;
```

Figure 1: Sample CIF file.

The "rasterizer" program transposes the component symbols into a two-dimensional array that represents the original design when it was laid out in ICE. [2] The preliminary version of the program will work strictly for 1900 $um^2$ designs. Since the design rule for the RIT PMOS process is ten microns minimum geometry, then the design will be divided by ten to get 10 $um^2$ pixels. The user will be required to layout the design on ten micron boundaries.

Each cell or pixel in the array holds eight bits (one byte) of information and represents ten square microns. The individual

layers are encoded into the byte by designating each bit as one of the PMOS process layers. Since there are only four layers used in the current RIT PMOS process (diffusion, oxide, contact cuts and metal) then there are four unused bits in each pixel. Figure 2 shows the bits set for the four process steps and the bits left for future expansion.

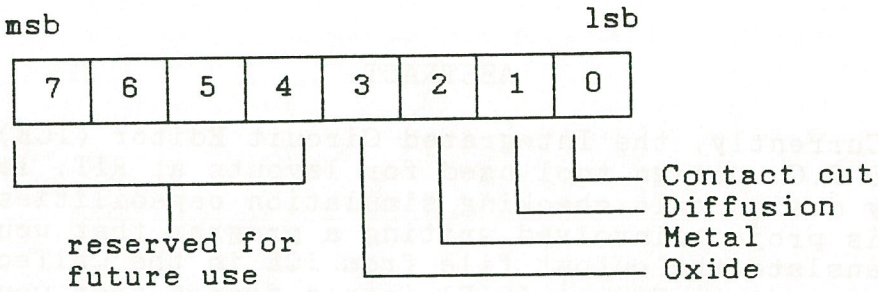



Figure 2: Process layer bit mapping.

The program will prompt the user for the CIF file to be converted and will translate the CIF commands in the following manner. If a LAYER command is found, a variable within the program will be set to the value to say three, which indicates that the current layer is OXIDE. If a BOX command is found, the size and position of the box is stored and the box is then placed into an array that will contain the overall design. This array is dimensioned as 190 X 190 bytes with each byte representing 10 um². This process is shown in Figure 3.

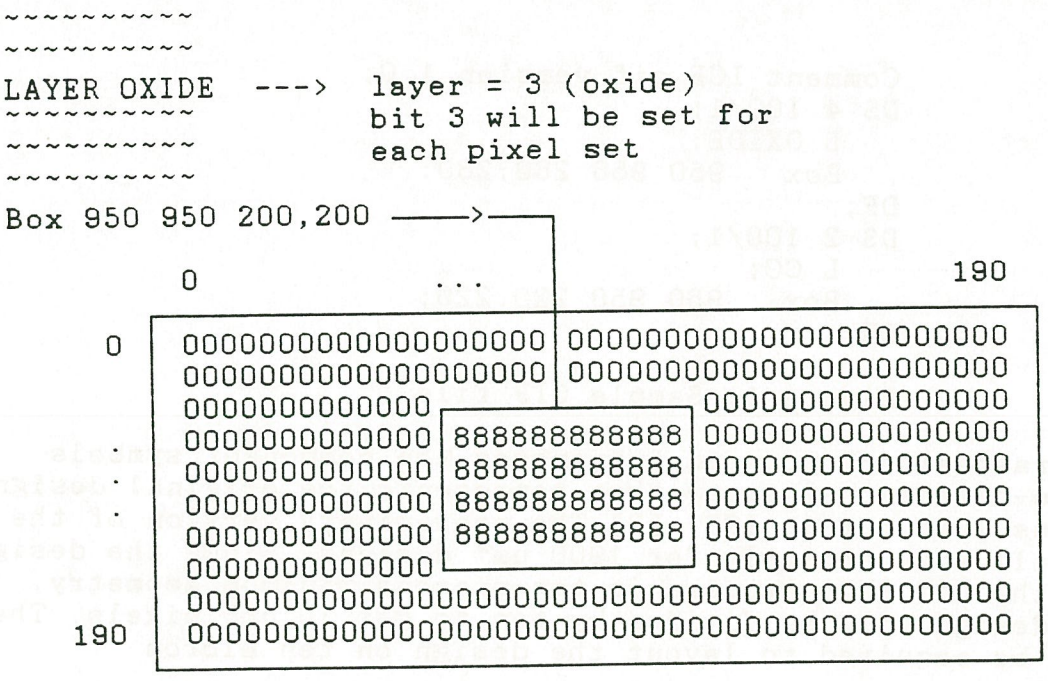



Figure 3: Interpreting the CIF file lines.

## PROGRAM DESCRIPTION

The program is laid out to accept a CIF file name from the user and then opens the file for input. The CIF file is then scanned line by line for commands. When a command line is intercepted, it is interpreted to determine which command the line represents (i.e. LAYER or BOX command). If the line is a BOX command, then the position and size of the box is used to place the box into an array at the appropriate process layer. Once all the lines have been scanned and the end of file is detected, the array is written out to a second file named "RASTER.OUT". If any errors are encountered, they will be reported and the program will terminate.

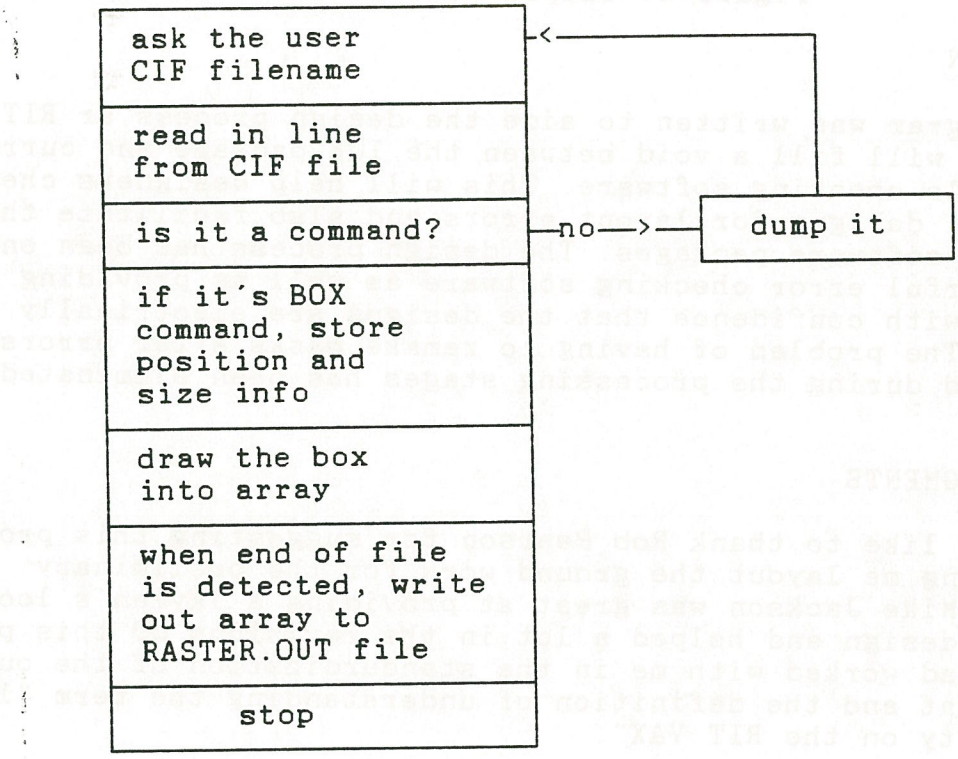
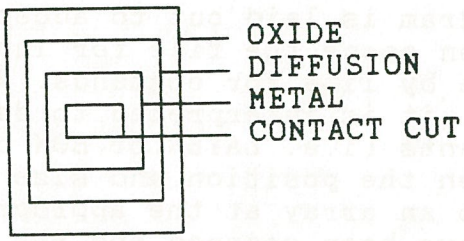



Figure 4: Program flow chart.

## RESULTS/DISCUSSION

The program name is RASTER.EXE and is executed on the VAX system by typing "RUN RASTER.EXE". The program prompts the user for a CIF file name and scans the file for CIF commands. The individual components in the original design are a series of boxes and are written into an array. Once all the boxes have been put into the array, it is written out to a file called "RASTER.OUT" in hexadecimal format. This file can then be read by design rule checking and node extractor programs to further check the I.C. design. Any errors detected will be reported to the user and the program will terminate. A sample output representing several process layers within the design is shown in Figure 5.

```
00000000000000000000
00000888888888000000
000008AAAAAAA8000000
000008AEEEEEA8000000
000008AEFFFEA8000000
000008AEFFFEA8000000
000008AEEEEEA8000000
000008AAAAAAA8000000
00000888888888000000
00000000000000000000
```




RASTER.OUT ICE SYMBOL LAYER NAME

Figure 5: Output file format.

## CONCLUSION

A program was written to aide the design process ar RIT. The interface will fill a void between the ICE package and current design rule checking software. This will help designers check their I.C. designs for layout errors and also facilitate the use of future software packages. The design process has been enhanced with powerful error checking software as well as providing the designer with confidence that the designs are electrically correct. The problem of having to remake masks after errors are discovered during the processing stages has been eliminated.

## ACKNOWLEDGMENTS

I would like to thank Rob Pearson for suggesting this project and helping me layout the ground work for the preliminary designs. Mike Jackson was great at providing a layman's look at software design and helped a lot in the revisions of this paper. Carl Conrad worked with me in the standardization of the output file format and the definition of understanding the term "lessons in futility on the RIT VAX".

## REFERENCES

1. A. Mukherjee, Introduction to nMOS & CMOS VLSI Systems Design, (Prentice Hall Publishers, New Jersey, 1986), pp. 280-282.

2. A. Mukherjee, Introduction to nMOS & CMOS VLSI Systems Design, (Prentice Hall Publishers, New Jersey, 1986), p. 283.