OPENCIF: A Fast Layout Viewer for VLSI Circuits

Rafael Araújo Rodrigues e Alessandro Girardi

Universidade da Região da Campanha – URCAMP
Campus Universitário de Alegrete
{rafael.rodrigues, girardi}@al.urcamp.tche.br

Abstract

This paper presents OpenCIF, a fast layout viewer for integrated circuits focused on high performance for read and draw CIF (Caltech Intermediate Format) descriptions containing millions of structures. The tool was implemented using OpenGL graphics technology in order to optimize graphics functions, providing real-time layout navigation. Also, the objective of this work is to make available a free tool for academic applications providing good performance in personal computers. Some comparison results with commercial tools are presented.

1. Introduction

CAD tools for integrated circuits design are fundamental nowadays, providing a wide set of tools in order to automate or semi-automate analog and digital design, which make possible to build million-transistors chips in a few weeks or months. However, the price of most commercial tools is prohibitive for educational purposes, turning the use economically unviable for most students.

This paper describes the development of a tool capable to read CIF files and draw the equivalent layout, providing to the user a graphical view for inspection on screen. The main objective of the tool is to supply a fast viewer for large circuits, capable of very fast print and real-time layout navigation. Also, we aim to provide free distribution for use in academia, friendly for even non-expert users, with a smart interface on Win32 platform.

2. CIF structure example

The CIF is a means of describing graphical items (mask features) of interest to VLSI (Very Large Scale of Integration) circuit and system designers [MEA 80]. Its purpose is to serve as a standard machine-readable representation from which other forms can be constructed for specific output devices such as plotters, video displays and pattern-generation machines. This type of textual description of integrated circuits became world standard due to the possibility of integration in different tools. The CIF is composed by a limited set of graphics primitives that describes 2D shapes in different layers of a chip, in text mode.

Fig. 1 shows a small example of CIF structure. The CIF format has a hierarchical structure based on cells, beginning with the DS command (definition start) (lines 3 and 11 in fig. 1) and ending with DF command (lines 10 and 20). Inside the cell there are layer assignments, calls and primitive function commands. The start call function is located at the end of the CIF file (line 21), calling the topmost cell of the circuit.

The box commands (lines 6, 8, 9, 18 and 19) have four parameters. This is the most used command in a generic CIF description that uses Manhattan style. The layer assignment is made by the command 'layer', lines 5, 7 and 17. Cell calls (lines 14 and 16) can have as attributes the following transformations: translation, rotation and mirroring, as can be seen in fig. 2.

Numeric functions are used for description aspects of CIF structure. In the example of fig. 1, the command '9' (lines 4 and 12) determines cell names, and the command '91' (lines 13 and 15) determines a call instance named below it. Other shapes that can appear in a CIF description are circle, polygon, path, etc.

3. The OpenCIF Render System

The strategy used is the classification of the boxes based on a data structure suitable for the representation of the points coordinates. This classification is performed according to the exact position of the boxes in relation to screen coordinates. The reading procedure of a CIF file is performed sequentially, identifying primitives and storing them in a first buffer. Using the first buffer, the reading procedure identifies the initial call for the first time CIF file is drawn and, at the same time, draw on screen and store them in the second buffer. At this moment, the circuit is plainified and the exact positions of the cells are calculated. The memory is dynamically allocated in order to optimize the memory consumption without loss of performance. The primitive shapes are stored in a second buffer based on a Cartesian plan. For the layout redraw, the software accesses the limits of screen visible coordinates and redraws only the shapes in the buffer that are inside these limits.

```
(@@ Association of Transistors);
(@@ Matrix Sea-of-Transistors);
DS 1 1 1;
9 CelBasic;
L CMET2;
  B 400 390 0 -100;
L CPOL1;
  B 30 300 -75 400;
  B 80 50 0 150;
DF;
DS 2 1 1;
9 TAT;
91 CelBasica_1;
C1 R 1 0 T 0 0;
91 CelBasica_2;
C 1 MX R -1 0 T 0 0;
L CMET1;
  B 50 50 100 -250;
  B 50 50 100 250;
DF;
C 2;
E
```

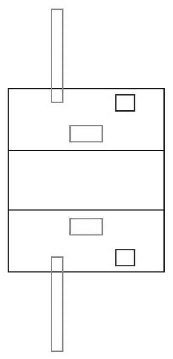
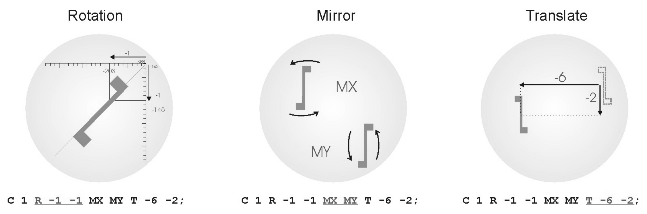


Fig. 1 – CIF description example



```
C 1 R -1 -1 MX MY T -6 -2;    C 1 R -1 -1 MX MY T -6 -2;    C 1 R -1 -1 MX MY T -6 -2;
```

Fig. 2 Demonstration of transformation tasks from parameters of the Call function.

The cell transformations are implemented by the multiplication of shapes coordinates by canonicals matrices. A point (x,y) given in the symbol is transformed to a point (x',y') in the chip coordinate system by a 3 x 3 transformation matrix T:

$$[x' \; y' \; 1] = [x \;\; y \;\; 1]T$$

The matrix T is itself the product of primitive transformations specified in the call: T= T_1 T_2 T_3, where T_1 is a primitive transformation matrix obtained from the first transformation primitive given in the call, T_2 from the second, and T_3 from the third (of course, there may be fewer or more that three primitive transformations specified in the call). These matrices are obtained using templates for each kind of primitive transformation.

One of the great problems of manipulating thousands of rectangles that compound an integrated circuit is that it is not always necessary to draw all boxes on the visible screen. To override this problem, the picture drawn can be clipped, by proper setting of the windows. OpenGL automatically clips off parts of objects that lie outside the world window [HIL 01]. Also, objects whose width or length is less than one pixel are not processed during the zoom action. These procedures highly increase the overall performance of redraw during the navigation across the layout. The navigate commands are processed only by the OpenGL API in the device context. Other necessary readings of the planning buffer are performed just for layer operations, such as color changes, layer hiding, or wired and filled view. The main feature for a good performance for drawing function with OpenCif is its pixel map setting inside OpenGL API, enabling double buffer drawing, which provides extremely fast drawing performance.

4. Comparison results

In order to compare the performance of the OpenCIF with other commercial equivalent tools, we realized a test of read from disk and draw a complete CIF file containing 1.474.073 boxes stored in 2,06MB of disk space. The other tools analyzed were CleWin 3.1 Layout Editor, from WieWeb Software and LinkCAD 5.5.19 from Bay Technology, both for Windows. Fig. 3 shows the comparison results between the tools. Opencif achieved the best time, reading and drawing the CIF file in 4.38 seconds. LinkCAD obtained 5.22s and CleWin 15.39s. The test was realized in an Intel Celeron 2.4Ghz with 256MB of RAM, running Windows XP Professional.

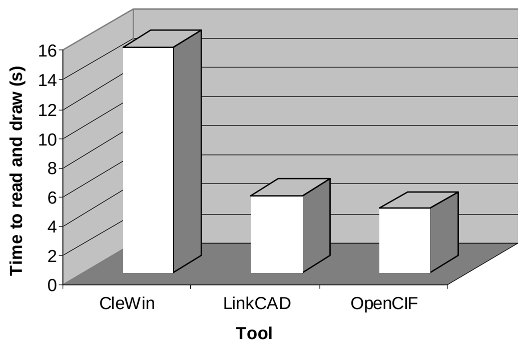


Fig. 3 – Comparison graph between the time required to open and draw a CIF file with 1.474.073 boxes

5. Conclusion

The first tests shows that OpenCif is a good tool, which can compete in performance with commercial softwares, although its focus is academic applications and free distribution. Its main objective is to supply the demand for free CIF layout viewers for Win32 platform

References

[MEA 80] Mead, C.; Conway, L.; *Introduction to VLSI Systems*; Addisson-Wesley Publishing Company, 1980.

[HIL 01] Hill, F. S.; *Computer Graphics Using OpenGL*; Prentice-Hall; 2nd edition; 2001.