A CAD System for Teaching the Design of VLSI Circuits - Status and Evolution -

M. L. Anido and C. E. T. Oliveira

Núcleo de Computação Eletrônica da UFRJ
Cx. Postal 2324 - CEP 20.001 - Rio de Janeiro - Brasil
e-mail:ncd10121@ufrj.bitnet

8 July 1992

Abstract

This paper presents the main characteristics of TEDMOS - a CAD (Computer Aided Design) system for teaching the design of VLSI (Very Large Scale Integrated) circuits. The paper starts by discussing the main assets of TEDMOS, when compared with commercial CAD systems, and presents the high-level characteristics of its main constituent programs. At the end, the paper discusses the evolution of the TEDMOS system.

Keywords: CAD, VLSI, Circuit Simulation, DRC, Extractor, Graphic Editor.

1 Introduction

The development of integrated circuits depends ever more on software tools to, automatically, generate and verify the basic building blocks specified by the circuit designer.

The motivation, sometimes difficult to be understood by researchers in the First World, that led to the development of TEDMOS, can be easily explained by two reasons. Firstly, commercial and powerful CAD packages for VLSI design are very expensive and require powerful workstations. Secondly, these packages have a very long learning time. These reasons forbid these CAD packages to be made available to large groups of students, which is normally the case in undergraduate courses. On the other hand, commercial CAD packages are essential to develop highly complex designs.

TEDMOS is a rather complete CAD package for teaching VLSI design, which is distributed free of charge to universities and research centers. TEDMOS has been highly successful in several Latin American countries among the teaching community, because it runs on IBM-PC© like workstations and because its highly integrated programs increase productivity which compensate other weakness originated from system simplicity. The system is orientated towards full custom design. This design style does not provide the same productivity of standard cell or gate array designs. Nevertheless, it allows very compact designs and also allows students to learn the basic operation of many fundamental transistor structures (cells). Additionally, this approach can be used for digital, analog or mixed designs.

2 An Overview of TEDMOS and its Main Constituent Programs

The TEDMOS system comprises several well integrated programs, that is, the user can invoke some programs within others, without leaving the environment. The main programs of the TEDMOS

system are:

- *File Utility Program* - This program permits to perform several operations on files and directories, without leaving the environment, such as: file deletion, copy, rename, print, change directory, etc.

- *Text Editor* - The text editor allows file editing operations, without leaving the environment, which are sometimes necessary to edit configuration files.

- *Cell Editor* - The cell editor is a graphic editor which allows the creation and modification of cells. This editor is targeted to edit cells and not the complete layout, as opposed to the hierarchical editor described below.

- *Hierarchical Editor* - The hierarchical editor was developed to permit the assembly of the many cells which generally comprise an integrated circuit. It also allows editing individual cells, similarly to the cell editor. The hierarchical editor permits to edit much larger circuits than the cell editor does, because it deals with subtrees of the layout tree and because it uses a more compact data structure than the cell editor does.

- *PLA Generator* - This program accepts logic equations in the form of sum of products and generates the final layout, automatically.

- *Switch-Level Simulator* - Switch-Level simulation can be used before the actual layout of the chip to make sure that the designed circuit implements the planned function. It can also be employed after the layout and extraction phase to verify that the layout implements the planned function. This simulator has a graphic driver capable of displaying the waveforms generated by the simulation.

- *Design Rule Checker* - Each fabrication process is characterized by a set of parameters that describe the valid geometric constructions used in the integrated circuit. This set of rules is called geometric design rules.

- *Extractor* - The circuit extractor takes a geometrical description of the circuit and provides a transistor netlist which can be used by other programs for logical and electrical verification of the layout.

- *Electrical Simulator* - The electrical simulator is composed of three main units, namely: Preprocessor, Electrical Simulator and Graphic Driver. The Pre-Processor converts extracted files to SPICE-like inputs for simulation. The electrical simulator performs a simulation of the circuit, using a Wave Relaxation [5] algorithm , which allows speeding up the simulation. The Graphic Driver makes it possible to visualize the waveform outputs from de electrical simulator, on the screen or on plotters.

3 The Cell Editor of TEDMOS

The graphic editor is the nucleous of TEDMOS [4]. By using it, the designer can create and modify cell layouts of integrated circuits on the computer screen. The screen depicts the complete layout or part of it. The designer moves the rectangular cursor around the screen and orders the editor to fill that rectangle with a certain pattern, which represents one of the fabrication masks (see figure 1). The editor of TEDMOS was designed to operate with orthogonal (Manhattan) geommetry, that is, it is not possible to draw rectangles with inclination.

This cell editor uses an in-memory *bitmap* to represent the data structure of the mask geommetry. Each element of the *bitmap* (matrix) is composed of a set of bits. Each bit indicates the presence or absence of a mask at that point. This representation is only valid for Manhatam or orthogonal geommetries. The algorithms for this type of representation are trivial. However, the memory area required is much larger than it is using alternative data structures. This form of representation is only used internally to the system and it is converted to a more compact form of representation (CIF language) to be archived. The visualization of the editing matrix is performed by mapping each element of this matrix onto a set of screen

pixels.

In order to copy one area of the layout to the screen, it is necessary to encode each element of this area, generating a matrix of points that have to be written onto the screen pixels. This encoding task is accomplished by a look-up table, that is created from an external configuration file. This file is distributed together with the TEDMOS system, but it can be modified by the user.

4 CIRCO - The Hierarchical Editor of TEDMOS

CIRCO is a hierarchical editor, that is, it allows working with a hierarchy of cells which can be represented by blocks that can be interconnected. It was developed to faciliate the assembly of the many cells that usually comprise an integrated circuit [8]. All graphics editing functions of the cell editor can also be found in the hierarchical editor. Additionally, there are special functions for the manipulation of hierarchies. Figure 2 illustrates the interconnection of some cells in CIRCO.

The main groups of commands of CIRCO are:

- *Commands related to Cursor Position and Others* - Among these commands are: Cursor Fill, Cursor Delete, Change Layer, Select Visibility, Change Scaling Factor, Move Window, etc;
- *Area Commands* - The most important commands of this group are: Area Adjust, Area Mirror, Set Position, Area Copy, Return to Position, Repeat Area, Adjust Dimensions;
- *Node Commands* - Among the most important commands are: Create Node, Delete Node, Show/Hide Node, List Nodes;
- *Cell Manipulation and Hierarchy Commands* - The most important commands are: Cell Edit, Cell Store, Create Call to Cell, Remove Call, Move Call, Create Hierarchy, Remove Hierarchy, Expand Hierarchy, Edit Symbol Table.

4.1 The Data Structure of the Hierarchical Editor

Every graphics editing operation involves a Memory Storage Organisation and a Disk Storage Organisation, which are discussed below.

4.1.1 Memory Storage Organisation

Cell editing is an interactive process and in consequence the time required to execute the editing tasks is directly proportional to the time required to access the stored information. Therefore, the data structure has to allow the fast editing of the information.

Sequential search is an important limitation to be considered when defining a data structure, because it is a time consuming operation. In order to solve this problem, a bucket partitioning technique [2] was employed, and it is very efficient for this type of application. The layout is divided into rectangular regions - buckets - of equal size. Each bucket emcompasses a list with the boxes that intercept its region. Figure 3 illustrates the structure of buckets of a hypothetical layout. Because a certain box can occupy an area that intercepts more than one bucket, such box will be associated to several buckets. This limits the sequential search operation to the boxes of a certain area, searching only the buckets that they intercept. The selection of the convenient boxes is done by comparing the minimum and maximum co-ordinates [2]. Care has to be taken to avoid boxes already selected in previous buckets, because a box can occupy an area that incorporates several buckets.

Each box is associated to a layer. This association is represented by several lists of boxes, one for each layer of the layout. The advantages are: (a) speed in the search of boxes for certain layers, because it is not necessary to traverse all the list elements and (b) memory space, eliminating one field in each element of the list. This representation is very efficient to redraw areas by temporal priority of layers. In this case, only the boxes of the visible layers are searched, and only once, for the whole design.

Besides occupying less space in memory, this type

of implementation allows greater speed in the execution of editing operations. These operations are basically:

1. Select the target region;
2. Traverse the data structure, finding the elements which belong to the region;
3. Modify lists associated to the selected elements;
4. Redraw a screen region;

The amount of memory space required to store lists of elements of all cells would be considerably large, once the layout of a complex circuit may incorporate tens of thousands of elements. For this reason, the editing task is performed on one cell at a time. Additionally, editing its subcells is restricted to editing the contour of such subcells. Accordingly, only the elements of the cell being edited have to be stored in memory.

However, the editing task demands that all the hierarchy specified by the user be stored, either for the allocation of a subcell or for the access to its calls. This data structure has to store all the set of cells that constitute the layout and also all the hierarchic structure, created or modified by the user during the editing operation.

A symbol table is used for this purpose and is subdivided into two structures:

- *A Symbol Table* - A table describing the set of cells used.
- *A List of Calls* - Each cell points to a list of its subcells, which can be one of the lists of elements already described above. Only the list of calls of the cell being edited has to be in memory.

According to this organisation, the hierarchical structure is obtained by searching the lists of calls of each subcell. This implementation allows that several parts of the whole layout (each one with an independent hierarchical structure) can be maintained in the set of cells. This allows greater freedom in the design phase of the circuit and during updates of the hierarchical organisation. It is possible to delete a complete subtree of the layout and use it later, once all subcells remain in the symbol table and the structure still remains intact in its lists of calls. The information about each cell, contained in the symbol table is:

- *Number* : Identification used by the program, which is present in the CIF file;
- *Name* : Alternative Identification;
- *Dx,Dy* : Cell Dimensions;
- *File* : File Name Containing its description;
- *Connections* : A description of the connections of each cell.

Part of the symbol table is shown on the screen and it can be edited. The dimensions of each cell were included in the symbol table to be consulted during the drawing operation, if necessary. Therefore, to draw the contour of subcell calls it is not necessary to fetch this information from disk. The same occurs to show the connections of a certain cell, which can also be found in the symbol table. However, information about connections is not shown on the screen, although it is present in the symbol table.

The data structure chosen for the storage of the layout was the implementation of linear lists of elements, with linked lists. Linked lists are more dynamic in insertion and deletion operations, dispensing with preliminary memory space allocation.

4.1.2 Disk Storage Organisation

A disk storage organisation of one file for each cell was found to be appropriated because it is the simplest organisation. This disk organisation allows a sequential file describing its elements for each cell of the circuit, generally in textual format. One of the main aspects considered when choosing a disk layout organisation was the need to provide easy editing of cells coming from non-hierarchical editors, particularly the cell editor of TEDMOS.

In order to fullfill the requirements stated above, the stantard format CIF 2.0 (Caltech Intermediate

Format) was chosen. It allows the description of the design in a portable textual format and it also allows the definition of the hierarchy. Each file defines the elements of a single cell, including the calls to its subcells.

5 The Design Rule Checker

Each fabrication process is characterized by a set of parameters that describes the valid geometric constructions used in the integrated circuit. This set of rules is called geometric design rules.

Due to the extremely large number of transistors and masks in an integrated circuit, it is impossible to verify the correctness of the circuit by visual means. However, this can be done reliably by a computer program that sistematically verifies all the design rules of the circuit.

According to Mead and Conway [6], the geommetric verifications can be classified in categories, such as:

- *Width of Lines* - The dimension of a line of a given layer should be larger than a minimum value.
- *Spacing Rules* - Two geommetric forms of the same layer or of different layers, should keep a minimum distance between them.
- *Extension Rules* - Extension rules are used when a geommetric form should extend over another by a certain value.

The design rule checking task is performed within the editor, that is, the designer does not have to exit the editor to run the DRC. TEDMOS's verification routines are parametric, which means that both the layers and the circuit dimensions involved on each verification routine are read from a file, on execution time. A set of rules is described in this file which specifies rectangle width, rectangle spacing, extensions and transistor construction.

The DRC algorithm employed is a variant of the raster method [3], which is compatible with the form of representation used by the cell editor. The algorithm performs a double scan of the circuit for each rule been scrutinized. This scan operation is performed in two steps: line by line in the horizontal direction and column by column in the vertical direction. The layout is verified as a line of elements, where each element is the set of layers present at that point.

This algorithm is not sufficient to verify all cases and this a deficiency of this DRC program. For example, it does not detect spacing violations at corners. To solve this problem, a new and hierarchical DRC, capable of working with rectangles instead of a *bitmap* description, is under development.

6 The Circuit Extractor

The circuit extractor takes a geometrical description of the circuit and provides a transistor netlist which can be used by other programs for logical and electrical verification of the layout. Basically, the circuit extracting problem consists of the following operations:

1. *Locate transistors, including source, gate, and drain terminals.*
2. *Find transistor connections.*

The circuit extractor of TEDMOS operates on a *bitmap* description of the layout and produces an output file which contains two types of registers: transistors and nodes. The register of transistors associates a number to each transistor and gives a number to each transistor terminal. If two terminals are interconnected by a wire, they will receive the same number. The set of interconnected terminals and wires is termed node. During the editing operation, the designer can give a name to a node to identify simulation points, for example. The registers of nodes identify the name of each node.

This circuit extracting method is based on the *lakes and islands* algorithm [1], which tries to solve the following problem: given a matrix of zeros (0) and ones (1), where 1 indicates ground and 0 water, atribute a code to each contiguous region of ground (island). The solution of this problem is a bidimensional automata that scans the bitmap

from top to bottom and from left to right, attributing a code to each position. This code is a function of the codes of the positions immediately above and immediately to the left. The code zero (0) indicates that the region is water.

This algorithm is capable of extracting wires only. Nevertheless, it can be used to identify peripheral regions of transistors, determining also gates, drains and sources. In order to achieve this, every time that the bitmap contains an intersection of polisilicon with diffusion, this region is marked as a transistor channel. The algorithm is also capable of extracting the capacitances and resistances of wires and transistors, based upon the area of the elements. A new extractor program that takes as input a hierarchical description of the circuit based on rectangles, instead of a *bitmap* description is also under development.

7 The Switch-Level Simulator

A simulator is a program that describes the behaviour of a model of a system. This model can be created with several levels of detail, for example, registers, logic gates and transistors. Each model describes the system in a certain level of abstraction, which can be convenient, depending on the stage of the project. After producing the layout it is very difficult to reconstruct the model at the level of logic gates, and therefore a simulator working at the level of transistors is more adequate.

The switch-level simulator can be used during the design phase to check if a given network of transistors implements the logic operation required. Most important, it can also be used to verify if the implemented layout performs the logic operation desired. This type of verification is fundamental in a full-custom design because of the errors that can occur when editing the layout.

Transistors can be modelled with several degrees of refinement. The more refined the transistor model, the more precise will be the results. However, the computation time will increase. The circuit model used by the TEDMOS switch-level simulator is based on the MOSSIM I simulator [3] and contains two elements: transistors and nodes. Each transistor is modelled as a perfect key, with interchangeable terminals.

Circuit nodes can be in one of the states (0,1,X), where 0 and 1 correspond to low and high voltage levels, respectively, and X denotes an undefined state corresponding to an uninitialized node or a node with a signal value between 0 and 1. Nodes can also be of three types: input (when a node is a voltage power supply with an infinite current supply), Pull-up (when connected to a load transistor) and Normal (none of the previous cases).

A transistor is modelled as a three-terminal (gate, drain, source) bilateral device which could be in any of three states: 0 (open or noconducting), 1 (closed or conducting), and X (undefined, intermediate conductance between its conductance when open and when closed).

The switch-level simulator works well for synchronous circuits or time-independent circuits (e.g. combinational circuits and synchronous finite state machines). The model of transistor used is rudimentary and does not allow the calculation of precise delays or voltage levels. However, this type of simulator provides very fast responses, making it possible to simulate large circuits, which is very useful for a first validation of the circuit.

This is a *unit-delay* simulator, that is, it uses a virtual delay of one time unit for all transistors. The output of the simulator is the output of the steady state of the circuit. It is not possible to analyse the dynamic or transitory behaviour of the transistor network.

A visual interface was created to show the results of the simulation, emulating a ten (10) channel osciloscope and it is shown in the computer screen allowing the simultaneous visualization of 10 points of the circuit. Figure 4 illustrates an example of the output of the switch-level simulator.

8 The Electrical Simulator - ONDAS

The electrical simulator ONDAS uses an input format that is compatible with the SPICE II simulator, except for some restrictions [5]. This for-

mat has the advantage of being very well known among microelectronics designers. Unfortunately, this input format is not user friendly, particularly for the beginner. In order to allow the creation of input files for electrical simulation, a converter program was included in the TEDMOS sytem and it generates a SPICE format, from the output of the circuit extractor. The generated file incorporates a series of informations which are necessary for the electrical simulation and are:

- Technology parameters;
- Dimension of the Elements of the Circuit;
- Capacitance of the Elements of the Circuit;
- Simulation Commands;

ONDAS was developed aiming the reduction of the processing time required by conventional electrical simulators, such as SPICE. Its operation is based on the Wave Relaxation algorithm, that explores some of the characteristics of MOS-VLSI circuits which are stated below:

1. MOS transistors are highly unidirectional devices, that is, source and drain currents are highly dependant of the gate voltage, while the current at the gate node is almost independant of the drain and source voltages.
2. In a MOS-VLSI circuit, typically less than 20% of the node voltages varies significantly in a certain moment of time; ONDAS tries to solve only the equations of active nodes, which represent a significant time reduction.
3. The simulation process occurs in discrete time intervals, termed integration steps. In order to obtain precise results, certain waveform regions, which vary rapidly, have to be calculated in more points, that is, using smaller integration steps.

In conventional simulators, the integration steps are dictated by the worst case, that is, by the waveform that varies more rapidly in the circuit and this sequence of integration steps is used for the whole circuit. The algorithm used by ONDAS allows that one of the waveforms be discretized using its own sequence of integration steps, which implicates considerable reductions of processing time, in comparison with conventional electrical simulators.

The algorithm used by ONDAS has guaranteed convergence only when applied to MOS circuits containing only transistors and capacitors. For this reason the simulator can not be applied to the analisys of circuits containing bipolar transistors, diodes, resistors, inductors and other elements. The convergence of the algorithm used in the simulator is conditioned to the existence of non-zero capacitances connected to each active node of the circuit. By active nodes it should be understood those nodes not connected to the power supplies.

9 PLA Automatic Synthesis

Programmable Logic Arrays (PLA) are structures with regular layout, which are used to implement logic functions. These are mostly used in the control section of integrated circuits. This program accepts logic equations in the form of sum of products and generates the final layout, automatically. A technology file has to be created for each technology.

A PLA consists of two matrices called the planes of the PLA. The most common forms of PLAs are NOR-NOR (two NOR planes) and NAND-NAND (two NAND planes). In order to define a PLA, a language is used to describe the logic equations. For example, a PLA can be described as follows:

```
OP planumber 10;      * cell number
EN A B C;             * normal inputs
SN S1 S2;             * normal outputs
EQ S1 = AB;           * equation
EQ S2 = AB + A'B'C; * equation
```

10 Evolution of the TEDMOS System

The encouragement provided by TEDMOS users, from several countries, has been the driving force to improve the system. New tools to improve the system are under development. They will allow

the design of medium size to large size circuits, using PC-like workstations and other more powerful workstations. Among the improvements and new tools under developemnt are:

- A TEDMOS version for Windows;
- A hierarchical DRC;
- A hierachical Extractor;
- A new version of the Electrical Simulator - ONDAS;
- Porting the system to a SPARC workstation;
- A channel router;
- Automatic synthesis fo regular circuits, such as ROM and RAM.

11 Concluding Remarks

This paper described TEDMOS - a rather complete CAD system for teaching the design of VLSI integrated circuits.

Commercial and powerful CAD software packages for VLSI design are very expensive, require powerful workstations and they also require a continual investment in maintenance and upgrading. Additionally, these packages demand a very long learning time. These reasons prohibit these CAD packages to be made available to large groups of students, which is normally the case in undergraduate courses. TEDMOS fills this gap at zero cost - it is distributed free of charge to universities and research centers.

Presently, TEDMOS runs on IBM-PC© like workstations and its highly integrated programs increase productivity, which compensate other weakness originated from system simplicity.

12 Acknowledgements

The TEDMOS system was originally developed by Dr. Eber A. Schmitz, José A. Borges, Jonas Knopman, Júlio T. C. Silveira, João Assis, Paula Cyrillo and many other students to whom we sincerely thank for their work. We are currently working on the improvements stated on section 10.

References

[1] Baker, C. M. ,"Artwork Analysis Tools for VLSI Circuits", Cambridge, Mass., MIT, 75p, M.Sc. Thesis, (MIT LCS TR-239), 1980.

[2] Borges, J. A. S.,"Editores Gráficos para Projeto de Circuitos Integrados", Rio de Janeiro, COPPE/UFRJ, M.Sc. Thesis, 1987.

[3] Bryant, R. E., "An Algorithm for MOS Logic Simulation", Lambda, Palo Alto, CA, Redwood Systems Group, 1(3): 46-8, 50-3, Fourth Quarter, 1980.

[4] Schmitz, E. A., Borges, J. A. S. and Knopman, J., "TEDMOS - Um sistema de CAD para ensino de projeto de circuitos eletrônicos de alta integração", Revista Brasileira de Computação, Vol. 5, N. 2, pp. 45-61, Out/Dez, 1989.

[5] Knopman, J., Mesquita Filho, A. C., Schechtman, J., "Convergence Properties of Relaxation Methods in the DC Analysis of Large MOS Circuits", Proc. IEEE Int. Symp. on Circuits and Systems. Espoo, Finland, June 7-9, pp. 1639-42, 1988.

[6] Mead, C. and Conway, L. "Introduction to VLSI Systems", Reading, Mass., Addison-Wesley, 396p., 1980.

[7] Schmitz, E. and Borges, J. A. S., "Projeto de Circuitos Integrados CMOS", Rio de Janeiro, LTC, 1989.

[8] Silveira, J. T. C., "CIRCO: Um Editor Gráfico Hierárquico de Circuitos Integrados para Microcomputadores IBM-PC", M.Sc. Thesis, Instituto de Matemática da Universidade Federal do Rio de Janeiro, 61p, 1990.

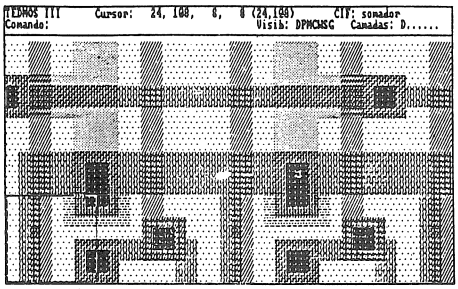

Figure 1 - The Graphic Cell Editor.

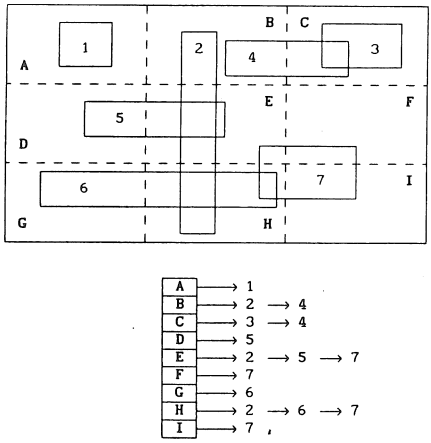

Figure 3 - Partition of The Layout in Buckets

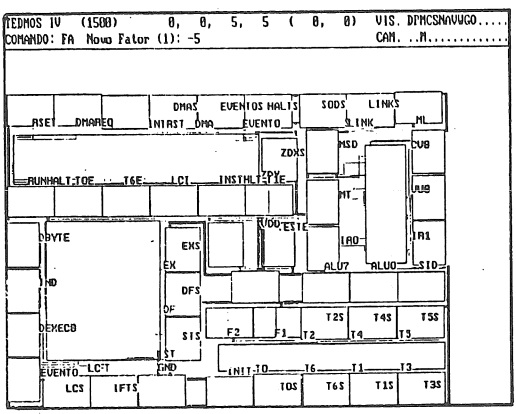

Figure 2 - The Hierarchical Editor.

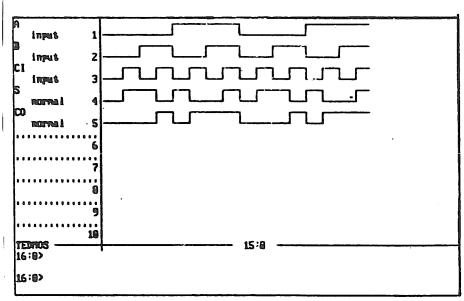

Figure 4 - A Plot Output of the Switch-Level Simulator