A Virtual CMOS Library Approach for Fast Layout Synthesis

F. Moraes, M. Robert, D. Auvergne

PUCRS - Av. Ipiranga, 6681 - Faculdade de Informática - Prédio 30
90619-900 - Porto Alegre - Brazil
e-mail: moraes@inf.pucrs.br

LIRMM, UMR 5506 CNRS/Uni. Montpellier 2 - 161 rue Ada
34392 - Montpellier - Cedex 5 - France
e-mail: robert@lirmm.fr, auvergne@lirmm.fr

Key words: layout synthesis, physical design, virtual libraries

Abstract: We present a layout synthesis methodology based on the use of virtual CMOS libraries, i.e. using no pre-characterized cells. The proposed methodology is organized around an automatic layout generator, allowing fast on-the-fly implementation of macro-cells. The generator eliminates the need for post-layout compaction procedures and in addition produces parasitic capacitances estimations. Results show that it is possible to quickly generate dense layouts, allowing fast prototyping of logic functions. The proposed method can change the way layout synthesis is seen today, since accurate parasitic evaluation is an important prerequisite for optimized submicronic designs.

1. INTRODUCTION

Shrinking product lifecycles, increased design complexity and time to market are some of the fundamental issues that must be addressed by the IC designers. The recent market trend towards Systems on Silicon (SOS) manufactured with deep submicronic processes has increased the design complexity dramatically, which has lead to the frequent need to reuse IP cores. At the physical level the use of virtual libraries [1] instead of cell

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35498-9_57

L. M. Silveira et al. (eds.), *VLSI: Systems on a Chip*

libraries, module generators or full custom layout is an alternative for a fast prototyping of these designs on different processes.

In this paper we propose a design methodology where layout synthesis, performance evaluation and optimization are associated in one single design flow as represented in *Figure 1.* The concept of a virtual library is based on using cells available through a layout generator, instead of using a set of pre-characterized cells. The set of available cells is a user defined constraint (number of transistors in series, maximum fanout and output isolation). The technology mapping process starts with a Boolean description and its output file is a netlist of Static CMOS complex gates at the transistor level, which is transferred to a layout generator. The first layout synthesis creates a load file, with routing and diffusion capacitance. Using this load file, with the original Spice file, accurate time analysis [2] and sizing can be performed. The second layout synthesis generates the final layout.

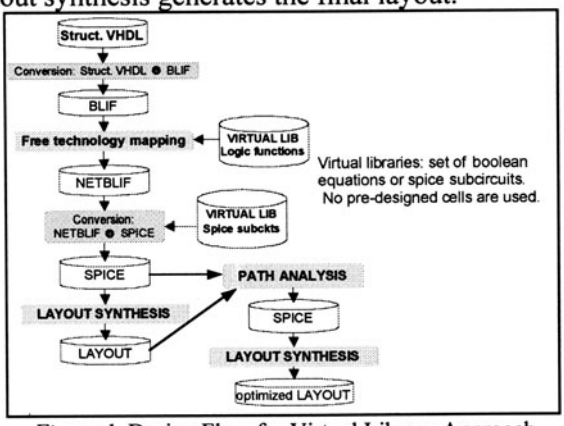

Figure 1. Design Flow for Virtual Library Approach

Using layout synthesis tools [3][4], it is possible to overcome the main limitations of the standard-cell approach: fixed number of functions and fixed transistor sizes. However, a virtual library approach may present the following limitations:

- Cells are not pre-characterized. The cell performance (area, delay, and power) can only be obtained after layout synthesis, since each cell will be generated according to its environment. Layout generation, parasitic capacitance extraction and electrical simulation are CPU time consuming. As will be shown, it is possible to break this cycle, obtaining the parasitic elements immediately after layout synthesis, reducing in this way the amount of CPU necessary to validate a circuit.
- Cell topology is fixed. Usually, only dual gates (same number of N and P transistors) and transmission gates are implemented. When a cell generator implements dynamic logic, like NORA or TSPC, the transistor density is poor [5].

– Automatic layout synthesis design flow requires more steps than standard-cell flow, like layout compaction, detailed extraction and electrical simulation. Layout generation must be entirely transparent and fast to the designer, while efficiently coupled to the logical synthesis step, to technology mapping and to performance optimization tools.

For these reasons, layout synthesis tools are not widely used in the semiconductor industry. However, virtual library methodology has recently grown in importance, since layout synthesis tools may speed up the generation of IP blocks for process migration. Additionally, this can be done without necessarily creating links to specific fabrication process or cell-library vendors.

The approach presented here is original. Our main goal is to quickly synthesize the entire layout of a block, starting from a gate level description (in fact, transistor level), without layout compaction, still maintaining reasonable transistor densities. If layout compaction is employed, more than 3 hours are necessary to translate a symbolic description to the final masks for 5000 transistor blocks, in an Ultra-Sparc 10. Using our approach, we synthesize the layout and calculate the parasitic elements in less than 2 minutes.

As CPU time for layout generation is no more a bottleneck, iterations may be used for optimization. The logic synthesis tool can perform initial iterations to get accurate information on routing length and cell area. This gives facilities for buffer and repeater insertion, since the load of each node is calculated during the layout synthesis.

The technology coding is very simple, requiring only basic rules such as distances, separations and overlapping. For parasitic estimation the data required is the area and the peripheral capacitance of each layer. In addition to the fast synthesis, easy technology migration allows to synthesize circuits in new processes, without waiting for new cell libraries.

This paper is organized as follows. Section 2 presents the layout style at the cell and circuit level, as well as the technology coding. Next, in Section 3, the evaluation of the parasitic elements is introduced. In Section 4 we present our preliminary results, and finally some conclusions and future work.

2. LAYOUT STYLE

The main difference between cell based technology mapping and library free technology mapping concerns the libraries (pre-characterized or virtual) they must cope with. Library free technology mapping implements the functions *directly at the transistor level*, while guaranteeing that the final netlist of complex gates respect some topological constraints (e.g. number of

transistors in series). The great number of available complex gates will improve the design space and lead to a minimization of the overall number of transistors, minimizing the design at the transistor level.

The output of the library free technology mapping is a netlist at the transistor level. Two alternatives can be considered at the physical synthesis phase: library or macro-cell generation. During *library generation* ([6] and [7]) a dedicated cell library is created either from symbolic templates or from a regular layout style. These cells have common characteristics (constant height and fixed pin positions), to be treated homogeneously in the same way as with a library in the cell-based approach. After library generation, placement and routing tools for standard-cells are used to obtain the final layout. In *macro-cell generation* ([4]) the complete block will be generated. The initial description is decomposed into leaf cells that will be assembled together using dedicated place and route tools, without constituting a separate library. Two instances of the same logic function can have different layouts, according to its environment. Our approach employs such a method.

Due to the use of regular layout styles for cell or macro-cell synthesis, normally followed by compaction [7], the final transistor density obtained from library free synthesis is smaller than that obtained from the standard-cell approach, where the cells are handcrafted. However, the final area for the library free approach can be *smaller*, due to the use of complex gates, which reduces the overall number of transistors.

2.1 Layout Synthesis

The target technology is CMOS, with 3 metal layers for routing and no restrictions on stacked contacts. Transistors are implemented using the linear-matrix style, where two horizontal diffusion lines are used to implement transistors. Supply lines are placed between transistors, in metal2. The contacts to the substrate (body-ties) are inserted over the supply lines. A metal1 stub is used to connect vcc/gnd nodes to supply lines, requiring a stacked via over body-ties.

The algorithm to implement the layout of a row starts from the left side of each row, processing column by column (*Figure 2*). Each node (drain/gate/source) has its status defined during the data base construction:

- *External row node*: inputs and outputs nodes to be connected to the routing regions. The pin assignment algorithm determines these nodes.
- *Internal row node*: supply, OTC, not connected (e.g., opposite side of a gate) and internal connection (e.g., abutted drains in a nand gate).

Using the design rules and the node status, it is possible to construct the real layout, without intermediate descriptions. The procedure used to implement the rows examines the node status. Only two cases occur: either

the node must be connected to the routing region or it must not. If the node must be routed, it will be *aligned* to a grid, with a step defined by the contact rules. Otherwise, the node coordinate will be fixed respecting the minimal distance between transistors. Automatic jog insertion for gates (polysilicon layer) is necessary to reduce the circuit width and the diffusion area.

Over-the-cell nodes will be implemented in metal2, over transistors. The over-the-cell routing is divided into two steps: the first one moves all internal nets of complex gates to the transistor region, and the second one fills each track with nets from the routing region. Experiments over a large set of complex gates [8], up to 4 transistors in series, gives a reduced number of tracks for internal cell routing. In this way, the width of the transistor area is not penalized by this approach.

Different layer directions are used inside row and routing regions. At the row level, diffusion and metal2 are used horizontally; while polysilicon, metal1 and metal3 are used vertically. For the routing regions, metal1 and metal3 are used horizontally; and polysilicon and metal3 are used vertically.

The connection between rows and routing regions is achieved in a dedicated track named "interface line". As can be observed in *Figure 2*, this region will connect the vertical layers. If we have polysilicon connected to polysilicon no contact is needed; otherwise, for metal1-metal3 connection a stacked contact-via1 must be implemented.

The row generation step creates the real layout for each row. Before routing, it is necessary to insert feedthroughs *over* rows. As all rows are transparent to metal3, vertical wires can be placed anywhere over the cells, with the following constraints:

- For any routing region, the number of different signals per vertical column is limited to two.
- Different feedthroughs in two adjacent rows are forbidden.
- Feedthroughs are aligned to the routing grid.

If these restrictions are not observed, vertical cycles can render the routing impossible. Automatic jog insertion over metal3 wires is performed to manage the difference of constraints between column sides.

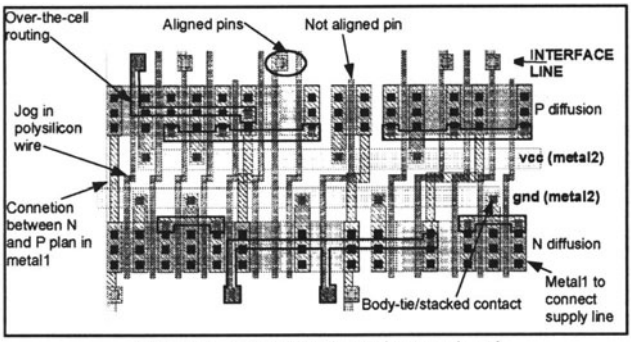

Figure 2. Layout style at the row level

The last step is the routing. A modified left-edge algorithm is employed. The choice of routing layers was made aiming the reduction on the number of contacts in the interface region. If the coupled capacitance between metal1 and metal2 (horizontal layers) is too high it is possible to change the directions to poly-metal1 and metal3-metal2 or to define the maximum superposition length between two wires as a design rule.

2.2 Technology Coding

The input of our layout generator is constituted of only two files: the Spice netlist and the design rules. The outputs are the final layout and a load report for each net. Three sets of rules are necessary:

- *Geometrical rules* (28 rules): for each layer the minimum width and minimum separation for contacts and vias the length, distance and margin to metal; transistor construction rules and power supply width. *Figure 3* illustrates the design rules used for the transistor construction.
- *CIF rules* (15 rules): define the CIF name for each layer of the layout.
- *Electrical rules* (15 rules): for each layer the area and peripheral capacitance, oxide capacitance and "load factor limit" (ratio of load to input capacitance).

The routing grid is defined by equation (1).

$$grid = 2*max(MCTO, MVIA1, MVIA2) + max(LCTO, LVIA1, LVIA2) + max(DM1, DM2, DM3) \quad (1)$$

Where:

- *MCTO, MVIA1, MVIA2*: margin of metal to contact;
- *LCTO, LVIA1, LVIA2*: width of contact;
- *DM1, DM2, DM3*: distance between same metal layer.

Pins assigned to routing are the only elements aligned to this grid. Drains, gates and sources not connected to the routing region (even OTC nodes) are *not* aligned to it. Some experiments were done with gridless routing. This approach was rejected due to the huge number of DRC errors introduced and to the resulting small area gain.

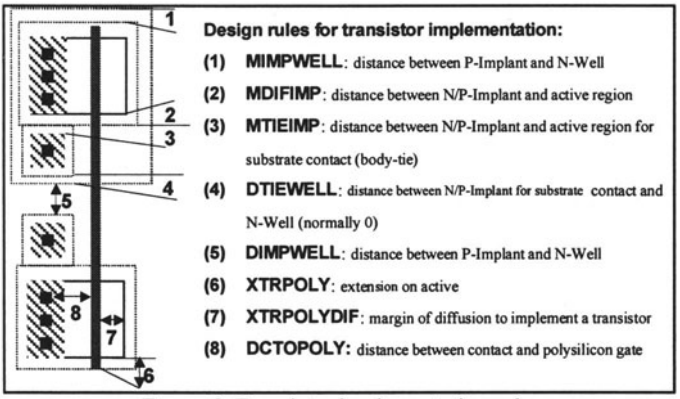

Figure 3. Transistor implementation rules

3. PARASITIC ELEMENTS ESTIMATION

This Section presents the approach used to compute the parasitic elements for each net. We considered only the capacitance to the substrate. No coupled capacitance is computed and a lumped model is used. The goal is to provide a fast evaluation tool, not a precise electrical extractor.

Two procedures can be adopted to compute parasitics:

- compute the wire length during routing,
- compute the wire length after routing, reading the layout file, as an extractor program.

The second approach is CPU time intensive, since it is necessary to find all connected polygons. Despite this problem, we adopted this solution, since this "*wire extractor*" can be used to verify the generated layout, giving information on eventual open connections and short-circuits (import tool during the development of routing algorithms). As we detail in Section 4, for a circuit with 14376 transistors, (257051 polygons), only 228 seconds (Ultra-Sparc 1) were necessary to compute the parasitic elements.

The parasitic capacitance for each net, C_{load}, is given by formula 2. *Figure 4* illustrates the three components defining the load capacitance.

$$C_{load} = C_{active} + C_{route} + C_{diffusion} \quad (2)$$

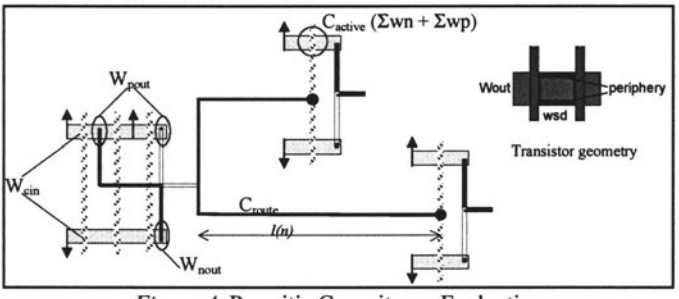

Figure 4. Parasitic Capacitance Evaluation

The active capacitance, C_{active}, is a function of all gate widths connected to the net:

$$C_{active} = (\quad W_n + \quad W_p) . l_{min} . Cox \tag{3}$$

The routing capacitance, C_{route}, is a function of the length of each layer implementing the net. It is important to subtract the polysilicon length over transistors (width) to compute polysilicon capacitance.

$$C_{route} = \sum_{n=poly,m1,m2,m3} (C_{area(n)}.w(n) + 2.C_{perimeter(n)}) . l(n) \tag{4}$$

The diffusion capacitance, $C_{diffusion}$, corresponds to the number of drain/source regions driving the net. We are assuming worst case behavior: the last transistor in a series branch is switching. The following formula is used to compute $C_{diffusion}$:

$$C_{diffusion} = \frac{(W_{nout}.Cjn + 2.Cjnsw).wsd.drainN\# + (W_{pout}.Cjp + 2.Cjpsw).wsd.drainP\# + W_{nout}.Cjnsw + W_{pout}.Cjpsw}{FDIF} \tag{5}$$

Where:

- *wsd* corresponds to the average drain/source length, and is calculated by the generator tool;
- *Cjn/Cjp* is the area capacitance and Cjnsw/Cjpsw the perimeter capacitance for the diffusion layer;
- *drainN#* and *drainP#* the number of drain/source regions driving the net. This number corresponds to the number of transistors connected to the longest series branch in each plan of the gate;
- the last two factors corresponds to the perimeter length at the end of a diffusion chain.

The input capacitance for a giving gate is given by formula 6.

$$C_{in} = (W_{n\_cin} + W_{p\_cin}) . l_{min} . Cox . drive \tag{6}$$

where:

- W_{n_cin} and W_{p_cin} corresponds respectively to the N and P average width of the gate,
- *drive* is the number of transistor pairs switching together. *Drive* is 1 for nand, nor, inverter and complex gates; and 'k' when we have an inverter with 'k' parallel transistors.

The load factor, $F_{load} = C_{load} / C_{in}$, corresponds to the ratio between the load and the input capacitance. When this factor is greater than the "load factor limit", defined into the design rule file, a warning message is sent to the user. This information can be used to define which gates must be **sized** or where **buffers** can be inserted.

The "wire extractor" supplies three output reports:

- *Flat spice netlist* with parasitic capacitances: this file can be used either by the electrical simulator or by the sizing tool.
- *Parasitic capacitances*:
  - Relative capacitances: C_{ative}/C_{in}, C_{route}/C_{in}, C_{diff}/C_{in} and C_{load}/C_{in}. If C_{load}/C_{in} is greater than the "load factor limit" a warning is printed.
  - Absolute values for capacitances (C_{ative}, C_{route}, C_{diff}, C_{load} and C_{in}).

- Total length per layer and the number of contacts in each net, data used to compute the routing capacitance and (*future work*) resistance.
- Topological data for each gate: number of inputs, fanin, fanout, number of transistors in series, average transistor width and the number of transistors connected to the longest serial branch in each plan (used to compute diffusion capacitance);
- Number of estimated buffers (C_{load}/C_{in} > load factor limit).

– *Wire length histogram*. Running the generator over a set of benchmarks (28 to 15000 transistors, 0.25 μm technology), we got an average value of 89% of connections bellow 200 μm. For this technology, 200 μm corresponds to a C_{route}/C_{in} equal to 2.5 (average transistor width equal to 2 μm). *Figure 5* shows a histogram for a circuit with 14376 transistors and 4764 nets (ISCAS c7552). In this example, 80.5% of the connections are bellow 100 μm, 9.9% between 100 and 200 μm, 7.4 % between 200 and 500 μm and 2.2% (102 connections) have the total length greater than 500 μm. The cells driving these nets must be sized, or buffer insertion must applied.

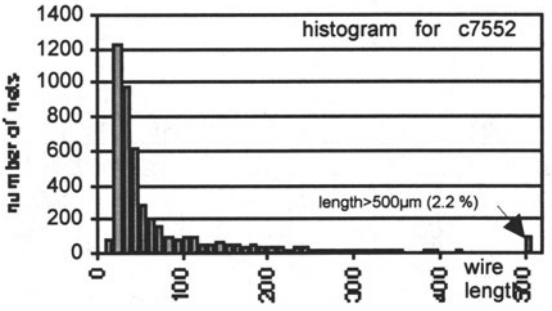

Figure 5. Histogram of wire lengths

The obtained results show that only a small fraction of nets have length larger than 200 μm. A tool for quickly finding these nets is essential in submicronic technologies, in order to guide the sizing and optimization steps. The wire length data can also be used in a second placement iteration to guide the placement of cells inside the critical path of the circuit. The placement algorithm, quadrature with pin propagation, is responsible for this homogeneous length distribution.

4. RESULTS

A method to estimate the silicon area necessary to implement layouts, starting from the number of transistors and the layout design rules, is presented in [9]. The data given in this paper represents an upper bound for the transistor densities. *Table 1* shows the maximum transistor density that can be achieved without routing. Routing area was considered as a constant

percentage of layout area, 50% and 25% for two and three metal layers respectively. Then, for a 3-metal layer 0.25 μm process, the transistor density that should be obtained with routing by a layout generator is 94000 transistors/mm^2 (125328*0.75).

Table 1. Transistor Density Roadmap (tr/mm^2) [9]

Process (μm)	Transistor Density
l=0.8	8296
l=0.6	14748
l=0.5	31332
l=0.35	62644
l=0.25	125328
l=0.18	250656

Table 2 gives the area, transistor density, CPU time for layout generation and for "wire extraction" (parasitic capacitance evaluation) obtained with our layout genenerator.

Table 2. Transistor Density and CPU time for layout generation and parasitic capacitance evaluation

Technology: 0.25 μm, 3 metal-layers and stacked contacts, regular sized transistors, w=2 μm, l=0.25 μm. CPU time in milliseconds for an Ultra-Sparc 1

Circuit	Transistors	Nets	Rows	area (mm^2)	Tr. Density (mm^2)	CPU (*ms*) Generation	CPU (*ms*) Par. Eval.
Adder	28	13	1	0.00028	100676	50	170
Addergate	40	15	2	0.00042	94482	180	310
alu	260	94	3	0.00353	73677	440	3970
Alugate	432	117	4	0.00535	80808	500	6050
rip	448	163	4	0.00500	89552	720	8360
cla	528	215	4	0.00700	75439	660	9010
Hdb3	570	191	6	0.00745	76539	420	9810
5xp1	798	308	7	0.01087	73386	790	15310
sao2	930	361	7	0.01321	70388	1030	17770
Mult6	972	308	6	0.01311	74147	1230	15800
9sym	1092	420	8	0.01547	70575	1170	22790
c499	1556	511	7	0.02279	68277	1730	28820
c1355	2244	647	9	0.02975	75437	2390	41460
c1908	3146	990	14	0.04764	66036	3850	63680
Mult2	4512	1239	13	0.07425	60768	5610	66220
c2670	4976	1762	15	0.08408	59181	6360	98230
c75523x3	6164	2101	15	0.13295	46363	14730	101100
c3540	7154	2359	15	0.13449	53193	9100	118810
Mult12	8584	2455	16	0.14757	58169	15110	133850
c6288	10112	2706	18	0.14931	67726	11030	162970
c5315	10656	3429	15	0.24822	42930	24360	167510
c7552	14376	4764	17	0.33096	43437	109700	228220

The average transistor density obtained with this first layout generator version is around 60000 tr/mm^2, to be compared to the ideal density: 94000

tr/mm^2 (0.25 µm technology). The main reason for this difference is the routing approach, which was based on the traditional channel approach (channels and feedthroughs). The number of rows was chose to minimize the area, and fixed as a parameter for the generator. Circuits with transistor density below 58000 tr/mm^2 (c3540, c7552_3x3, c5315 and c7552) have a great number of inverters (at least 55% of the cells). This fact reduces the row transparency, since there is less space to pass metal3 over the cells, reducing in this way the transistor density. One solution to reduce the used silicon area is to use a maze-based algorithm, routing as much as possible over transistors.

5. CONCLUSION

Our primary goal was to show that it was possible to quickly generate dense layouts without compaction. In submicronic technologies, the routing contribution for delay is equivalent to cell delay. We have shown, the capacitance for a wire length with 200 µm is equivalent to 2.5 cells. Then, if only cell delay is used for mapping, sizing and timing analysis, inaccurate results will be obtained. For a submicronic design flow, the layout generation step must give accurate information on parasitic in few minutes, instead hours. This fast layout generation can be used:

- in mapping tools to select which gates will be used (simple gates or and-or-inverters) or where buffers must be inserted;
- in sizing tools to correctly size transistors, using the parasitic capacitance evaluation, or insert buffers not placed in the mapping step;
- in timing analysis tools [2] to get an accurate post-layout performance evaluation.

From this first layout synthesis prototype, two directions can be investigated: (*i*) developing more accurate models for capacitance and resistance evaluation; (*ii*) implementing efficient routing algorithms for this layout style.

REFERENCES

[1] A.Reis; R.Reis; D.Auvergne; M.Robert. "*Library free technology mapping*". VLSI'97 - IX IFIP International Conference on Very Large Scale Integration, August 26-29 1997, Brazil, pp. 303-314.

[2] S.Cremoux; N.Azemard; D.Auvergne. "*Path resizing based on incremental technique*", ISCAS98, Monterey, USA, 1998.

[3] M.Guruswamy; R.L.Maziasz; D.Dulitz; S.Raman; V.Chiluvuri; A.Fernandez; L.G.Jones. "*CELLERITY: A fully automatic layout synthesis system for standard cell libraries*". DAC'97.

[4] CADENCE. "*Virtuoso layout synthesizer - LAS - user guide*". CADENCE™ Version 4.2, October 1991.

[5] S.Rekhi; J.D.Trotter; D.H.Linder. "*Automatic layout synthesis of leaf cells*". DAC'95.

[6] A.J.Velasco; X.Marín; R.Peset; J.Carrabina. "*Performance driven layout synthesis: optimal pairing and chaining*". Fith ACM/SIGDA Physical Design Workshop, April 1996, pp. 176-182.

[7] J.L.Burns; J.A.Feldman. "*C5M - A Control-Logic Layout Synthesis System for High-Performance Microprocessors*". IEEE Transactions on CAD, Vol. 17, no. 1, January 1998, pp. 14-23.

[8] R.L.Maziaz, J.P. Hayes. "*Layout minimization of CMOS Cells*". Kluwer Academic Publishers, 1992, 165p.

[9] F.Moraes; L.Torres; M.Robert; D.Auvergne. "*Estimation of Layout Densities for CMOS Digital Circuits*". PATMOS '98 - International Workshop on Power and Timing Modeling, Optimization and Simulation, October 07-09 1998, Denmark, pp. 61-70.

*The author Fernando Moraes would like to gratefully acknowledge the continued support of the "Conselho Nacional de Desenvolvimento Científico e Tecnológico" (**CNPq**), Brazil, to this work. Research grant 522939/96-1.*