設計自動化 4 9 − 4
(1989. 10. 24)

CIRCEL ; VLSI /CAD software tools for IC layout design with logic cell compiler, on-line analog simulator and process fault simulator.

Etienne Sicard and Kozo Kinoshita

Department of Information Sciences
Hiroshima University
Higashisenda-machi, 1-1-89 Naka-ku
Hiroshima 730 , JAPAN

Abstract- **In this paper we present some results about VLSI computer aided design for mask-level editing, including logic-cell compiler, schematic extractor, analog simulator and process-fault simulation. All modules are integrated in the same software, for a maximum interactivity and design efficiency. We specially describe in detail algorithms used in the process fault simulation module . Then we present some results about process fault-tolerant circuit designs.**

I. INTRODUCTION

At the era of 50 millions transistors in a chip, it sounds quite strange to talk about 100 transistors-capacity mask editor and analog simulator. The main goal of CIRCEL software is the integration into a single module of such different and powerful tools as logic cell compiler, analog simulator and process-fault simulator in order to generate high-quality IC designs.

The cell-compiler translates functional-level descriptions into CMOS layout; a summary of its principles is reported in chapter 2. The schematic extractor analyses the layout topology and computes all electrical parameters of the circuit such as transistors, capacitors and nodes. Then the analog simulator gives a SPICE-like simulation of the circuit. Discussion is made in chapter 3. The chapter 4 deals with the process-fault simulator (Also called inductive fault analysis or IFA). A small parasite, represented by an extra-layer or a missing layer is injected into the layout. Its electrical consequences are analyzed and the probability of occurrence is listed. As an application of that tool and according to the IFA indications, the layout topology may be modified and the process-fault sensitivity may significantly decrease.

II. THE LOGIC COMPILER

Very powerful silicon-compilers [1] are now commonly used in Computer-Aided Design of integrated circuits and their efficiency has been widely demonstrated. The logic cell-compiler [2] included in CIRCEL mask-editor has a very limited role, which is to generate CMOS mask-level designs of a logic cell according to its functional description.

The cell-compiler includes syntax analysis, transistor placement and optimization tools, then a short-wire routing tool in order to complete the final layout. Design rules as well as design strategies have been included as described in [3] into the compiler software. Examples are shown in figure 1. The logic description file contains three logic cells. Figure 2 shows the resulting design; the left cell is a simple inverter; the middle cell is a 3 inputs NAND gate. Then the right cell is a carry cell.

```
out = not ( in );
x = not ( a.b.c );
cout = (a.b) + (cin.(a+b));
```

Figure 1 : Logic description used as input of the cell-compiler.

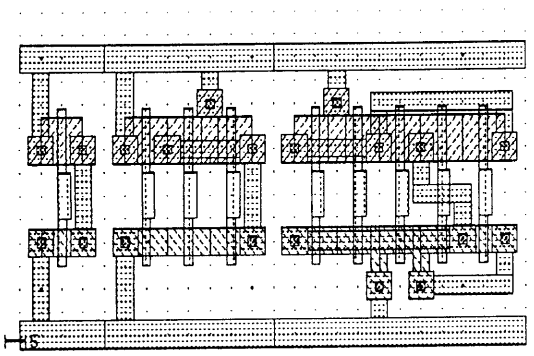


Figure 2: The logic cell compiler. Inverter, NAND and Carry examples.

The cell-compiler consists of three main software modules : the syntax analyzer, the transistor optimizer and the router. The **syntax analyzer** extracts the list of internal and external nodes, the list of n-channel and p-channel transistors from the given boolean equations. As the set of operators is restricted to the basic *and,or* and *not* , the CMOS translation becomes relatively simple [5]. The analyzer strategy can be described through the example of a 3-inputs NAND gate as follow :

x = not (a.b.c)

The left part of this equation concerns the output node. The right part consists of three input nodes called a,b and c. The not operator specifies inverted logic. Six transistors, three n-channel and three p-channel MOS are directly generated. The *and* operator is seen as an internal node in the n-MOS network while it provides identical numeration in the p-MOS network .

The **transistor optimizer** modifies the place and the orientation of the transistors in order to minimize the required silicon area. Each transistor is represented as a triplet of drain, gate and source nodes. The placement of the n-channel and p-channel is rearranged in terms of minimum active area, gates alignment and minimum internal capacitors.

The **router** establishes physical connections between the devices of the cell. Specific algorithms have been developed to handle short-wire and multi-layer routing, as internal connections usually do not exceed tenths of lambda but use n-diffusion,p-diffusion, polysilicon and metal layers. Due to their important parasitic capacity, n-diffusion and p-diffusion layers are used as less as possible. The metal 2 layer is reserved for external connection.

Then,as no option has been specified, the cell-placement strategy is a standard-cell like strategy with horizontal power busses; the p-MOS device net is placed up to the n-MOS device net. As compilation occurs, cells are attached to the right side of the other. The data file of design rules concerns CMOS process in that case. Approximate compiling time is 0.5 seconds CPU for each cell, using a PFU Σ 230 workstation. Depending on the complexity of the environment, routing tasks may require more time, raising this value up to 1 or 2 seconds.

III. EXTRACTION AND ANALOG SIMULATION

Extraction consists in the analysis of the layout and the edition of the complete list of electrical parameters of the circuit. CIRCEL automates the netlist generation, that is nodes, capacitances and transistors for a 2-metal layer, p-well or n-well CMOS technology. The palette consists of nine layers : p-well, n-well, diffusion n+, diffusion p+, polysilicon, contact, metal, via and metal 2. Figure 3 shows the extraction procedure for a CMOS 2-inputs NOR gate.

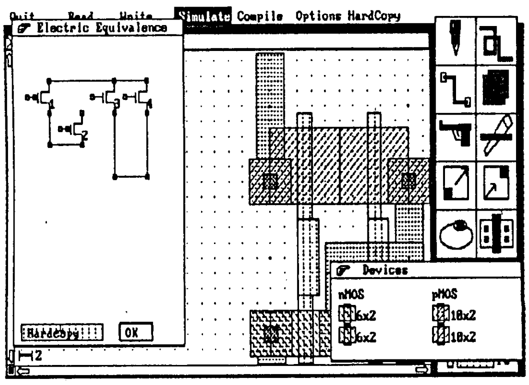


Figure 3: The netlist extraction of the CMOS 2-inputs NOR gate. The electric schema is generated automatically from the netlist.

The device netlist serves as the input data for an automatic schematic editor. The lowest primitive level is transistors, as shown in the schematic window on the left upper part of the figure. Simple CMOS structures such as basic combinational logic cells can be identified.

The next step is **analog simulation.** Parameters of interest are transistor characteristics, parasite capacitors and global network architecture. The analog simulator uses SPICE level 1 models for both n-channel and p-channel transistors while capacitors are modelized as perfect capacitors. Connective resistances are ignored.

The main goal of the analog simulator is the analyze and validation of the mask-level design. Critical parameters such as raise and fall time, or fanout ability can be deduced from the chronograms. The simulator can also be used to compare the behavior of similar cells in which some parameters, such as the transistor size have been modified. Figure 4 gives the example of a NOR cell into which the p-channel transistors have been permuted. Taking advantage of a smaller output node capacity, the right cell is 5% faster, with an identical load.

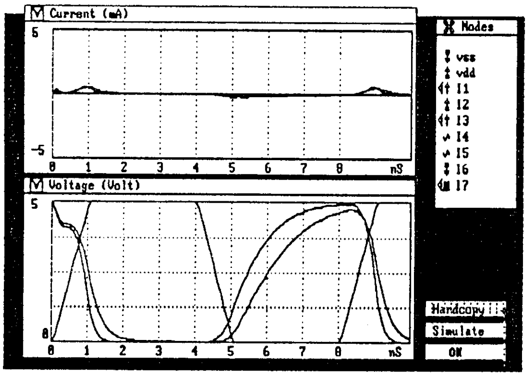


Figure 4:The simulator gives useful informations about the compared behavior of similar cells. Two designs of a NOR gate are examined and the simulation chronograms demonstrate that the right cell is 5% faster.

IV. FAULT ANALYSIS

As the cell compiler may generate a great variety of similar designs starting from an identical logic behavior, one new criteria of selection might be the process fault tolerance. Inductive Fault Analysis tools, so called IFA [6] have been recently developed to analyze the impact of small physical defects in the circuit.

According to the process fault statistics and using appropriate models [7], little spots of various sizes and layers are injected into the design. Then a comprehensive list of resulting faults is automatically generated [8].

A systematic and fully automatic IFA tool has been implemented in CIRCEL software. First, a CMOS design is generated, using the mask editor or the cell compiler. Then the fault analyzer is invoked and return the complete list and the probability of faults which are likely to occur during the process.

In order to achieve realistic predictions, CIRCEL uses a statistical defect distribution data file, as shown in figure 5. The first line of the text-file declares the technology while the second line concerns the fault probability,expressed in percents according to its size, expressed in µm. The sum of all values is 100. Only fault sizes from 1 to 10 µm are analyzed, as the probability of bigger one is almost zero. One representation of this distribution is reported in figure 6. Following this distribution, the next lines of the data file contain informations about the extra layer and missing layer probability for all the masks of the process. Then the truth table expressing the layer-to-layer continuity is declared. A one in the truthtable signifies that electrical conduction exists between the column layer and the row layer.

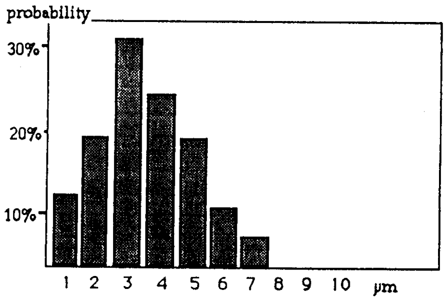


Figure 5: The defect size distribution reports the probability of occurrence in the range of 1µm .. 10µm.

```
----(process.dat)-----------------------------
technology= CMOS CMP89 nwell
distribution = 5,15,32,23,15,7,3,0,0,0
pwell = 0,0
nwell = 200,100
diffn = 1000,1000
diffp = 1000,1000
poly = 1500,1000
contact = 1000,1500
metal = 1700,1500
via = 200,1500
metal2 = 2000,2000
truth table 11 layers
10010000000
01000000000
... ....
```

Figure 6: The statistical defect distribution data file includes industrial process qualification parameters.

1. Fault primitives.

The three main groups of faults resulting from a process fault are [9] bridges, breaks and stuck-at transistors. Other faults such as extra-transistors and exceptions are rare and will not be discussed here.

Bridges are the most frequent and also the easiest faults to be analyzed. The first step is the complete electrical extraction from the mask-level design. The cell is divided into different electrical regions, using connectivity properties declared in the previous truthtable.

<u>Assumption :</u> If the distance between two regions is smaller or equal to the size of the defect, and if there is electrical conduction between the two associated layers and the fault, a bridge is created.

In the example of figure 7, a 2μm diameter contact defect may shorten the polysilicon gate and the metal wire in all the enlightened area.

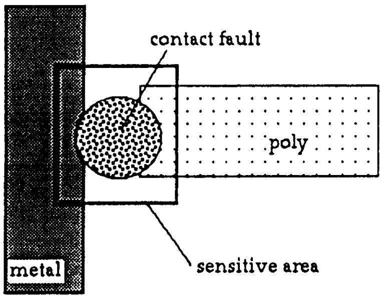


Figure 7: A small contact defect creates a bridge between the polysilicon and metal wires. The dangerous area is enlighten.

Breaks appear also very frequently but their systematic analysis is less simple. The break extraction software is very similar to the software for design rule checking. It can be stated as follow:

<u>Assumption</u>:If the size of on regions becomes inferior or equal to the size of the missing-layer defect, a break is created.

An illustration of this assumption is given in figure 8. A break occurs in a polysilicon wire resulting in a floating line.

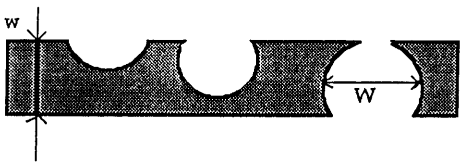


Figure 8 : Only a missing defect which size is greater than the line size can generate a break.

Unfortunately a break in one part of the region does not always signify that a new electrical region is created. The figure 9 shows the example of a redundant design, following the technique described in [10], where the polysilicon wire tolerates the breaks.

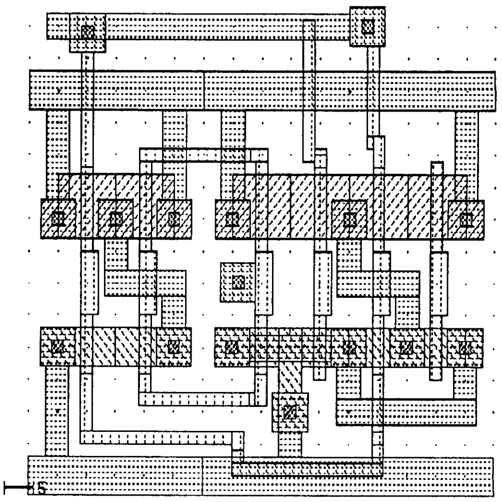


Figure 9 : A redundant polysilicon gate can tolerate missing-layer faults.

Design redundancy is detected while performing the global circuit extraction. Let us simplify the discussion by using only rectangles as graphic primitives. Each rectangle is attached to a node. Initially the node value is zero. The first rectangle of the data base is declared as *node one*. Using a recursive procedure, all rectangles touching this one and satisfying the connectivity condition also get this *node one*. Redundancy can be described as follow.

Assumption: If one rectangle satisfying the intersection and connectivity conditions already contains a non-zero node property, redundancy occurs.

Stuck-at transistor faults consist of stuck-at-ON and stuck-at-OFF transistors. A transistor is stuck-at-ON if its source and drain are shorten together because of a process fault, resulting in a transistor always conducting. A transistor is stuck-at-OFF if the conducting path between its source and drain is broken; the transistor is always non-conducting. Examples of stuck-at-ON and stuck-at-OFF layouts are shown in figure 10. The analysis algorithms are very similar to the earlier breaks algorithms.

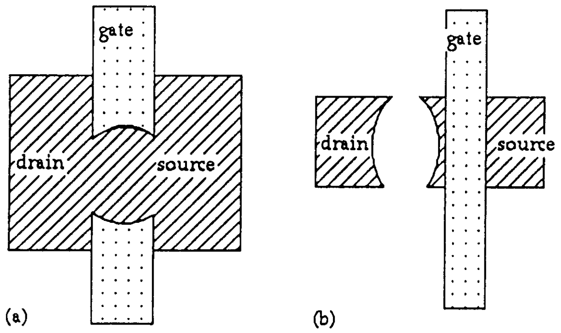


Figure 10: A missing polysilicon may induce a stuck-at-ON fault (a). A missing diffusion may induce a stuck-at-OFF transistor.

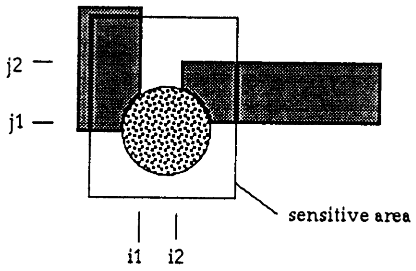


Figure 11: A small contact defect creates a bridge between the two metal wires everywhere in the sensitive area.

2. Fault processing.

We assume that the process defect can be imitated by a circular spot with a radius *r*. The radius of the defect is computed from the probability distribution given in the process data file. The IFA analyzer then gives the list of all *sensitive areas*. The sensitive area of a fault is the area in which its effect may be seen. In the example reported figure 11, the sensitive area into which a contact defect generates a short-cut between two metal boxes is enlightened.

The box coordinates are denoted by (ib,jb) for the lower-left point and (ih,jh) for the upper-right point. In the case of **bridges**, the sensitive area associated to a single defect having a radius r and generating an electrical short-cut between two wires may be approximated by the following formula :

$$a(r) = (i2-i1-2r).(j2-j1-2r) \qquad (1)$$

where a(r) is the sensitive area in lambda square, r is the radius of the defect in lambda, and
i1 is the minimum of the ih boxes coordinates,
j1 is the minimum of the jh boxes coordinates,
i2 is the maximum of the ib boxes coordinates and
j2 is the maximum of the jb boxes coordinates.

Then the fault weight is computed from the following formula :

$$w(l) = \sum_{i=1}^{n} a(i) . d(i,l) \qquad (2)$$

where w(l) is the weight of the l-layer fault, a(i) is the sensitive area associated to a i-radius defect, d(i,l) is the fault probability associated to the i-radius, l extra-layer defect, and n is the maximum defect size.

In the case of **breaks**, the sensitive area associated to a single defect having a radius r, assumed the fact that redundant condition is not satisfied, may be approximated by the following formula :

$$a(r) = (ih-ib-2r).(jh-jb-2r) \qquad (3)$$

where a(r) is the sensitive area in lambda square, r is the radius of the defect in lambda, and ih,jh,ib,jb are the wire coordinates. Then the fault weight is computed from the following formula :

$$w(l) = \sum_{i=1}^{n} a(i) \,.\, d'(i,l) \qquad (4)$$

where w(l) is the weight of the l-layer fault, a(i) is the sensitive area associated to a i-radius defect, d'(i,l) is the fault probability associated to the i-radius, l missing-layer defect, and n is the maximum defect size.

3. IFA results.

Both spatial fault distribution and exhaustive fault list are generated. The spatial distribution consists of a x,y map where all sensitive areas and associated weights are reported. The map is build by using a set of patterns(or colors) which indicates the fault density for each 1x1 lambda square of the design. An example CMOS circuit shown in figure 12 has been analyzed using the IFA procedure. The resulting fault distribution map is shown in figure 13 (a), as well as the breaks, bridges and stuck-at fault occurrence (Figure 13-b).

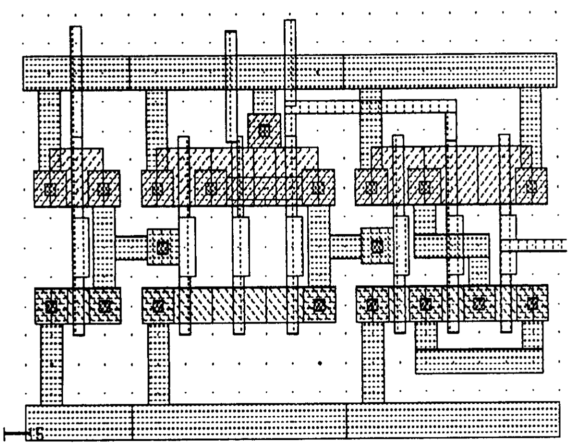
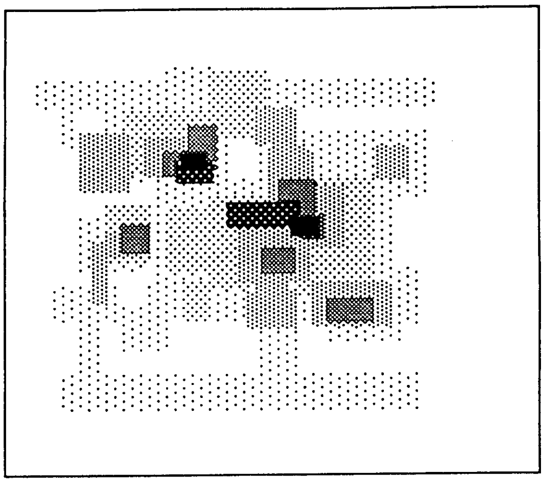


Figure 12: Example of a CMOS design used to validate the IFA procedure. Cells have been generated by the logic-cell compiler.

Comparative fault computing is a special tool which analyses the effects of layout modifications in terms of process-fault tolerance. All places where the fault-density has changed are enlightened so that the layout design improvement can be precisely evaluated.


(a) Fault map.

Size	2	3	4	5	6	7
Layer						
nwell	0	0	0	0	0	0
n-diff	0	0	3	3	8	344
p-diff	0	0	0	3	8	244
poly	0	152	152	174	174	195
contact	34	136	625	988	1089	1140
metal	0	100	286	890	1600	2630
via	0	0	0	0	0	0
metal 2	0	0	0	0	0	0

(b) Breaks netlist. The fault size is in μm. The sensitive area associated to each layer and to each fault size is in μm square.

Fault Type	Weight	Probability
Bridges	2194	35%
Breaks	3220	52%
Stuck-At On	743	12%
Stuck-At Off	75	1%

(c) Fault statistics : weight is evaluated from the bridges netlist, the breaks netlist (shown in b) and the stuck-at netlist.

Figure 13: The IFA resulting fault map includes bridges, breaks and stuck-at faults (a). The breaks netlist is shown in (b) while the fault statistics are presented in (c).

4. IFA applications.

As the nature and effects of the process-induced faults have been minutely detailed, fault model accuracy as well as test pattern set efficiency may be significantly improved. On the other hand, fault tolerance ability seems to be widely dependent on the layout topology. Then particularly undesired faults may be avoided by minor modifications of the IC layout.

New design strategies are currently being tested in the automatic design tools. The routing cost has been reevaluated in terms of short-cut sensitivity, the wire size has been adjusted to the channel density in order to decrease the break probability. Redundant gate strategy and more accurate short-wire routing algorithms seem to find some interesting applications in logic cell compiling. Research on each of these areas is under way.

V. CONCLUSION

A VLSI/CAD tool including logic cell compiler, analog simulator and fault extractor tools has been presented. As the cell-compiler generates a wide variety of layouts from a same functional description, analog simulation tool can be used to choose the most suitable solution in a given environment.

Furthermore, the IFA tool produces the list of faults, spatial distribution and other useful statistics associated to the layout, in order to evaluate the process-fault tolerance ability. New design strategies, deduced from IFA results, have been discussed. Significant process quality improvement may be expected.

REFERENCES

[1] D. Johannsen and S. Tsubota, "Intelligent Compilation", VLSI Design, April 1987.

[2] E. Sicard and J. Buxo, "EQUATIC: A CMOS logic-cell compiler for VLSI integrated circuits," in Proc. C3D Euro ASIC 88 , Grenoble, France.

[3] E. Sicard, "Contribution to the development of smart design techniques for VLSI integrated circuits ," Doctor Thesis, Toulouse University, FRANCE, no. 167 July 1987.

[4] N. H. Weste and K. Eshraghian, "Principles of CMOS VLSI design," Addison-Wesley, 1985.

[5] P. Kollaritsch and N. Weste, "TOPOLOGIZER: Expert translator from transistor connectivity into layout," IEEE Solid State Circuits, June 1985.

[6] J. P. Shen, W. Waly, and F. J. Ferguson, "Inductive fault analysis of MOS integrated circuits," IEEE Design and Test of Computers, vol. 2, no. 6, pp. 13-26, Dec. 1985.

[7]. W. Maly, "Modeling of lithography related yield losses for CAD of VLSI circuits," IEEE Trans. Computyer-Aided Design, vol. CAD-4, no. 3, pp. 166-177, July 1985.

[8] H. Walker and S. W. Director, "VLASIC: A catastrophic fault yield simulator for integrated circuits," IEEE Trans. Computer-Aided Design, vol. CAD-5, no. 4, pp. 541-556, Oct. 1986.

[9] F. J. Ferguson and J. P. Shen, "A CMOS Fault extractor for inductive fault analysis," IEEE Trans. Computer-Aided Design, vol. 7, no. 11, Nov. 1988.

[10] S. Koeppe, "Optimal layout to avoid CMOS stuck-open faults," in Proc. Design Automat. Conf., 1987, pp. 829-835.