A NOVEL FULL AUTOMATIC LAYOUT GENERATION STRATEGY FOR STATIC CMOS CIRCUITS

Cristiano Lazzari[1], Cristiano Domingues[1], José Güntzel[2], Ricardo Reis[1]
[1]*UFRGS - Universidade Federal do Rio Grande do Sul*
PPGC – Instituto de Informática
Porto Alegre – RS, Brazil
{clazz, cdviana, reis}@inf.ufrgs.br

[2] *UFPEL – Universidade Federal de Pelotas*
Departamento de Informática
Pelotas – RS, Brazil
guntzel@ufpel.edu.br

Abstract: The physical design of ASICs still relies on the standard cells because the design is well known and uses to produce good quality layouts. In addition, there are many choices of EDA tools that generate layout based on standard cells. However, in current CMOS technologies the standard cell approach is not able anymore to provide good performance predictability. Moreover, cell libraries have limited number of cells what imposes restrictions to layout synthesis. Automatic full-custom generators, on the other hand, do not use cell libraries and thus are more flexible to create optimized layouts. This chapter presents an automatic layout generator called PARROT PUNCH. Thank to a careful set of layout generation strategies and efficient algorithms, significant area and power optimization is achieved. Layouts generated by PARROT PUNCH are compared to those obtained by a similar automatic full-custom generator. Results show significant gain in area and delay.

Key words: Full automatic custom layout generation, Layout optimization, CMOS circuits.

Please use the following format when citing this chapter:
Lazzari, Cristiano, Domingues, Cristiano, Güntzel, José, Reis, Ricardo, 2006, in IFIP International Federation for Information Processing, Volume 200, VLSI-SOC: From Systems to Chips, eds. Glesner, M., Reis, R., Indrusiak, L., Mooney, V., Eveking, H., (Boston: Springer), pp. 197-211.

1. INTRODUCTION

Traditional physical level ASIC design flow still relies on standard cells libraries because it was the first automatic layout generation strategy to provide compact layouts. In fact it is not a full automatic layout generation because the cells are already designed. As the standard cells strategy has been used for more than twenty years, it is natural that this technique is well known and widely used. Moreover, within the customary pragmatism of industry, standard cells based layouts was a safe design flow because electrical performance could be accurately predicted, since the cells were pre-characterized.

However, with the advent of the deep submicron (DSM) technologies, geometries got smaller, clock frequencies increased and on-chip interconnect gains increased importance (Cong and Sarrafzadeh; 2000). In addition, problems in physical design are getting more complex and EDA (Electronic Design Automation) tools are essential to solve current design problems (Sarrafzadeh et al; 2001). Hence, the claimed predictability of the standard cells approach has been lost due to the difficulty on predicting the delays introduced by the routing.

Moreover, standard cells libraries have limited number of cells what imposes restrictions to layout synthesis. Also, different versions of each cell are required in order to drive different capacitive loads, thus increasing the total number of elements in the library to hundreds of cells. The larger is a library the more expensive is its update to a new fabrication technology.

An alternative to the cell-based layout generation approach is the automatic full-custom generation approach. An automatic full-custom layout generator does not use cells from a library. Instead, it generates each element (transistors and connections) according to a layout pattern that is intrinsically programmed within its algorithms. In addition, automatic generation can be flexible to create optimized layouts that are well tuned to a particular portion of the layout.

The pioneer works on automatic cell generation are those of Lopez and Law (Lopes and Law; 1980) and Uehara and Cleemput (Uehara and Cleemput; 1981). The former work introduced a layout style known as gate matrix, while the latter presented the so-called linear matrix layout style. Both works were originally developed having in mind the automatic generation of cell libraries. Current layout generators are mainly based on the linear matrix style and are able to generate combinational modules with up to tens of thousands of transistors.

PARROT PUNCH is an automatic full-custom generator based on the linear matrix layout style. PARROT PUNCH generates static CMOS layouts on demand for technologies with 3 or more metal layers. The tool tries to

generate layouts with full over-the-cell routing. When it is not possible, channels are created between adjacent rows aiming at a complete routed circuit.

This paper is organized as follows. Section 2 presents the layout generation strategy. Section 3 shows the topology and the layout style used in the layout generation by PARROT PUNCH. In Section 4 some optimization techniques associated to the PARROT PUNCH layout generation are presented. Some experiments are reported in section 5 and the obtained results are presented in section 6.

2. LAYOUT GENERATION STRATEGY OVERVIEW

PARROT PUNCH is an automatic full-custom layout generator able to deal with circuits containing any kind of static CMOS gates, including complex ones.

Figure *1* presents the main characteristics of a layout generated by PARROT PUNCH. In this layout it is possible to observe the transistor folding technique (used to obtain wider transistors), the internal connection positions, the body tie placement in the diffusion gaps, the use of minimal transistor spacing and input/output contacts between PMOS and NMOS transistors.

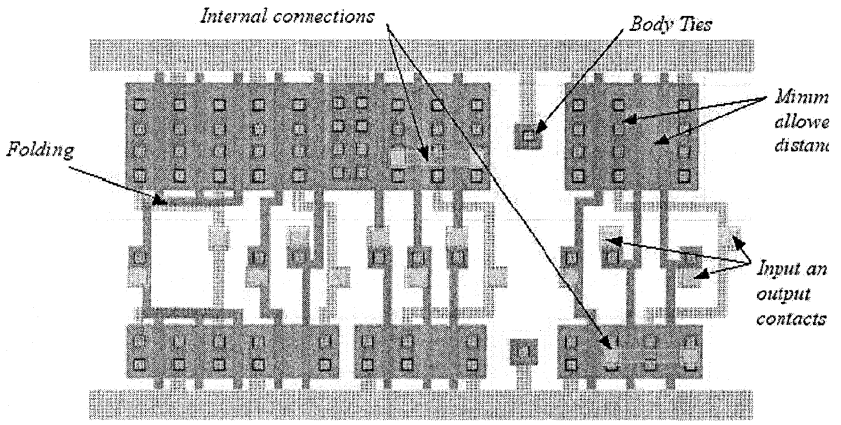

Figure 1. Main characteristics of a layout generated by Parrot Punch

In PARROT PUNCH generation follows the physical design flow shown in Figure 2. Each step is detailed in the next sub-sections.

2.1 Logic Gates Placement

Logic Gates placement consists on finding the position of each logic cell within the circuit surface. PARROT PUNCH generates layouts in which logic gates are placed in rows. Thus, the placement input is a list of logic gates partitioned in rows. Placement is performed as a separate step from the layout generation. Consequently, any placement tool can be used to realize this task.

The placement tool must be able to use an area estimate of each logic gate in the circuit because this information is not known in this step. Figure 2 shows the generation steps.

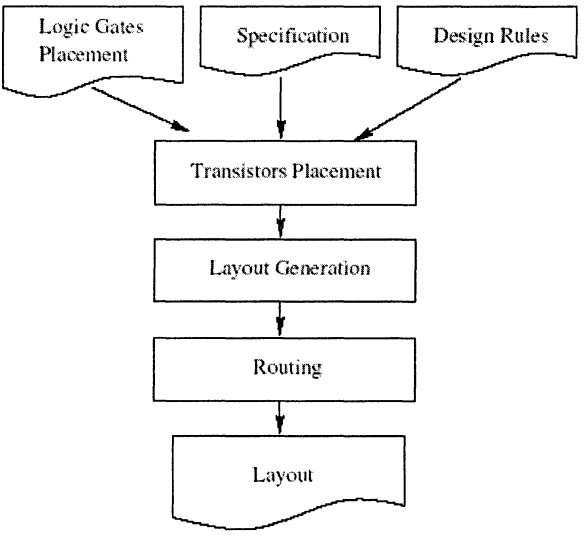

Figure 2. Physical Design Flow

2.2 Specification

Design specification is the set of desired characteristics for a circuit that is to be automatically generated. They are furnished by:

- A SPICE-like netlist
- The user parameters

PARROT PUNCH receives a SPICE-like netlist as input. This netlist is a set of logic gates defined by sub-circuits. Each sub-circuit in the netlist represents a logic gate and its position in the layout is specified in the placement output file.

User parameters consist on some definitions used in the layout generation as supply line characteristics and transistors width. These user characteristics influence on each logic cell in the circuit.

2.3 Design Rules

Design rules are the technologic rules furnished by the silicon foundry to develop the layout of masks that will be used in the fabrication process. They are basically the minimal dimensions for the rectangles or polygons in a given layer, metal enclosures to contacts and vias and minimal spacing between identical layers. The layout generation step tries to use minimal values of the chosen technology whenever it is possible.

2.4 Transistor Placement

Once logic gates and their positions in the layout are estimated, the layout can be generated. The transistor placement consists on searching the best as possible position for each transistor in a row. In the current version PARROT PUNCH only static CMOS circuits are generated. The used algorithm is responsible for ordering transistors in such a way that PMOS and NMOS transistors with common gate signal are easily connected together using only the polysilicon layer.

The Euler search algorithm is applied to a set of transistors (a logic gate) in which each transistor has an associated weight. The value of the weight is assigned to a transistor concerning the estimated position of the nearest point where it is connected to in the circuit. For example, a transistor that must be connected to another gate at its left side has more probability to be placed on the left. This is performed to reduce the wire congestion over the transistors of a gate.

2.5 Layout Generation

Layout generation consists on generating the geometries of each material (layer) that will be used to fabricate the circuit. In the proposed strategy, the layout generation can be divided into four main steps, being performed row

by row. The layout generation steps are expressed by the pseudocode of Figure 3.

```
proc generateLayout( Row ) {
  contactsPlacement( Row );
  diffusionDesign( Row );
  polyRouting( Row );
  outputAndInternalConnect( Row );
}
```

Figure 3. Layout Generation Algorithm

The placement of contacts is performed first in the layout generation because of the grid router and the layout topology. Contacts must be placed regarding spaces defined by technology rules as the minimal spacing between metal layers and taking into account the enclosures of these metal layers on contacts and vias. Figure 4 shows the grid spacing definition as the function `contactsPlacement()`. In the grid spacing definition, all metal layers are evaluated to fit the real grid spacing without violating design rules.

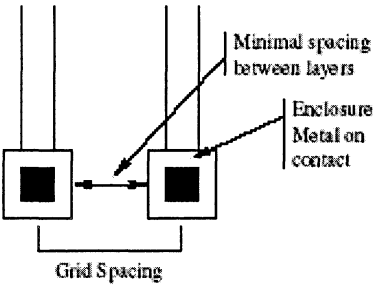

Figure 4. Grid router spacing

The used layout topology demands contacts placement as first step because contacts are inserted between PMOS and NMOS diffusion strips. Thus, the layout generation can be performed only when the positions of the contacts are already known.

The function `polyRouting()` performs polysilicon connection among contacts and gates in rows and function `diffusionDesign()` generates diffusion blocks. Once contact positions are known, diffusion blocks are performed with the aim of reducing source and drain areas of transistors. This is explained in more detail in section 3.

The layout of logic cells generated by PARROT PUNCH is performed using one metal layer (routing layers not included). Function `outputAndInternalConnect()` generates all internal connections and the output connection of each logic gate in the row.

The internal routing is divided in two parts: first, the polysilicon connections are generated and after, source/drains transistor nets are implemented.

2.5.1 Polysilicon Routing

Polysilicon routing consists on connecting transistors and input/output contacts that shares the same signal. Figure 5 shows four possible situations according to the transistor position and the contact inside the row. These situations are evaluated in the layout generation execution and the polysilicon routing is performed. They are the following:

- If the contact and the transistor are aligned with respect to the X-axis, they can be connected together by a straight line;
- If the contact and the transistor are not aligned with respect to the X-axis but there are no obstacles between the gate and the contact;
- When adjacent transistors are connected to the contact (folding technique);
- If the contact and the transistor are not aligned with respect to the X-axis and there are obstacles between the gate and the contact, the river routing algorithm is used to connect the gate and the contact together.

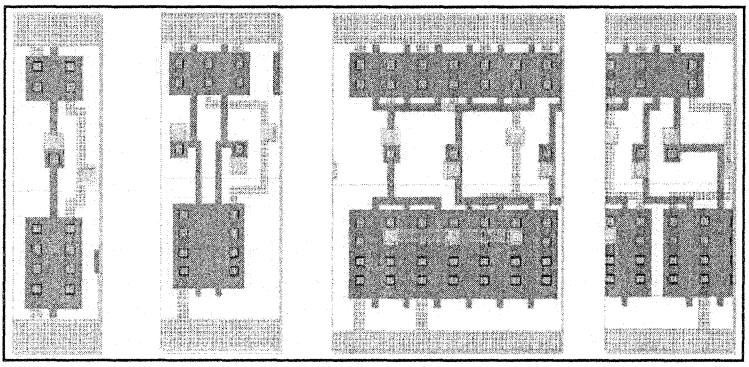

Figure 5. Four situations considered by the poly routing algorithm

2.5.2 Source/drain connections

These connections are implemented using the first and the second metal layers. The first metal layer is always used for wires between P and N diffusion strips. The second metal layer is used when the logic function demands connections over the transistors (as in the case of complex gates).

2.6 Routing

After the layout of each row is generated, connections among logic gates must be performed. A two-layer router integrated into the layout generation step realizes the routing. The first layer is used only for vertical wires while the second layer is used only for horizontal wires. The router starts at the top (north) of the layout, scanning and connecting related nets in direction to the bottom (south) of the circuit.

Once the routing is completed, the supply lines are generated. VDD and GND signals are connected to every row of the circuit and the contacts to substrate are placed in the layout.

3. PARROT PUNCH LAYOUT STYLE

The layout style generated by PARROT PUNCH is based on the *linear matrix* in which p- and n-diffusions are placed in parallel composing rows. Transistors are laid out in the diffusions strips in such a way they are easily connected to generate the layout of logic gates. The Euler Path algorithm is essential because it optimizes the transistors order in the rows, avoiding diffusion gaps. Figure 6 shows an example of layout generated by PARROT PUNCH.

In order to reduce drain and source areas in diffusions, contacts are placed inside the rows between p- and n-diffusions. When a grid router is used, contacts outside the rows can increase the diffusion areas or increase the length of polysilicon wires to connect the gate to the grid position.

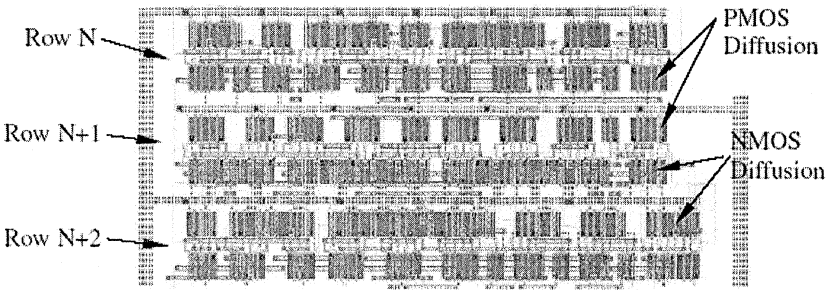

Figure 6. Layout generated by PARROT PUNCH

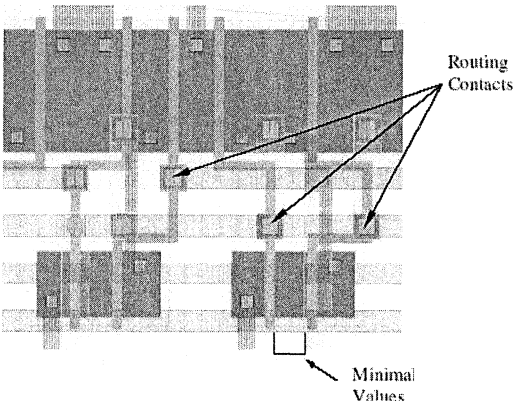

Figure 7. Contacts inside row reducing diffusion areas

Figure 7 shows a detail of a layout generated with the Parrot Punch. Due to the placement of contacts between p- and n-diffusion and according to the design rules, it is possible in the best case, to obtain minimal values of spacing between gates.

The whole layout (except routing) is performed using only one metal layer and one polysilicon layer. Supply lines are placed between adjacent rows in metal1. Rows are mirrored aiming at sharing supply lines and contacts to substrate (body ties) between each pair of adjacent rows. Figure shows an example of supply line between adjacent rows and contacts to substrate.

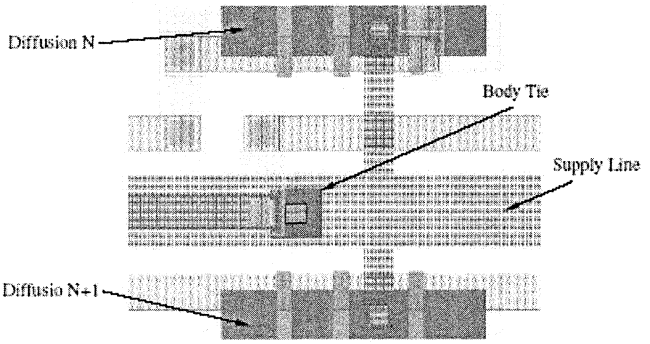

Figure 8. Supply lines and body ties are shared by adjacent rows

An important characteristic in the layout style of PARROT PUNCH is the attempt to reduce area by using full over-the-cell routing. In this routing style, routing channels between rows are only created when the routing cannot be completed using the minimal spacing between these rows.

TROPIC3 (Moraes, Reis and Lima;1997) is an automatic CMOS full-custom generator. Layouts generated by PARROT PUNCH and TROPIC3 contain some similarities. Both generators use a layout style based on the linear matrix one and both are able to deal with three metal layers. However, there are some important differences that are enumerated in table 1.

Table 1. Similarities and differences between layouts generated by PARROT PUNCH and TROPIC3

PARROT PUNCH	TROPIC3
Linear Matrix style	Linear Matrix style
3 or more metal layers	3 metal layers
Supply lines between rows (metal 1)	Supply lines inside rows (metal2)
Contacts inside the rows	Contacts outside the rows
No channel routing	Channel routing

Main differences between layouts generated by these tools rely on supply lines position, position of the contacts to routing and the attempt to use full over-the-cell routing by PARROT PUNCH.

4. OPTIMIZATION TECHNIQUES

4.1 Complex Gates and Power Optimization

Dynamic power consumption is still the dominant source of power consumption. However, high leakage current in DSM technologies is becoming a significant contributor to power consumption in DSM CMOS circuits when threshold voltage, channel length and gate oxide thickness are reduced (Roy, Mukhopadhyay, and H. Meimand; 2003).

Reis presents in (Reis, et.al; 1997) a method to map a set of Boolean equations into a set of static CMOS complex gates (SCCG) under a constraint in the number of serial transistors. In this work, a tool called TABA was developed to optimize a circuit under a set of complex gates in which the number of transistors is reduced. This kind of technology mapping is known as "library free mapping" and is particularly suited to be used along with a full automatic layout generator. Using the library free mapping together with a full automatic layout generator it is possible to generate an optimized set of cells with a reduced number of transistors in comparison to a standard cell mapping. In a logic mapping where the number of serial P and N transistors is constrained to 4, the number of possible logic functions is 3503 (Detjens and Rudell; 1987). However, this number is much greater than the possibilities offered by a regular library of cells. And in some cases it is possible to use more than 4 serial transistors, enlarging the possible number of logic cells.

The library free mapping is very important because it allows a reduction on the number of transistors of a circuit in comparison to a solution constrained by the use of library that has a restricted number of logic functions. Library free mapping can provide a 20% to 30% reduction in the number of transistors. Thus, the static power consumption is also reduced due to the reduction on the number of transistors.

4.2 The Gate Sizing Tool Integration

The accuracy of timing verification is completely dependent on the effectiveness of the used circuit model. By circuit models it is meant not only the physical delay model used to quantify the delay of each component, but also the models for computing circuit component delay and the circuit delay itself. Timing analysis associated with layout generation can improve accurate timing optimization characteristics to the circuit design. A tool

called *TICTAC::Sizing* (Santos, et.al; 2003) is able to analyze a circuit and realize gate sizing based on timing constraints. The integration between *TICTAC::Sizing* and PARROT PUNCH is done in such a way that transistors' size information, given by the sizing tool, can be easily used in the layout generation.

In timing-driven layout synthesis, transistor widths tend to vary significantly and the use of conventional layout approaches may cause inefficient area utilization (Kim and Kang; 1997). The folding technique consists on breaking a large transistor into smaller ones, connecting them in parallel and placing them continuously with diffusion sharing. The folding technique is especially important in the case of row-based layouts because different transistor sizes in a row could cause non-uniform cell heights leading to significant waste of area. The used folding algorithm (Bastian, et.al; 2004) tries to look at every folded transistor as a unique transistor, applying the folding over a given transistor placement neither modifying their placement nor inserting new diffusion breaks in the row. The algorithm introduces a new approach in which transistors are classified according to even or odd multiplicity. Then, the algorithm is able to verify when a diffusion gap can be avoided, thus reducing gate capacitances.

5. PRACTICAL EXPERIMENTS

In order to validate PARROT PUNCH some experiments were performed. Circuits were generated with PARROT PUNCH and TROPIC3 (Moraes, Reis and Lima;1997). In the set of benchmarks, circuits with the number of transistors between 294 and 1356 were chosen.

To provide a fair comparison, the quadrature placement algorithm presented in (Moraes and Velasco; 2000) was used to perform the placement for both PARROT PUNCH and TROPIC3 cases. The algorithm consists on dividing the circuit into horizontal and vertical directions, minimizing the cut-size to each direction. These quadrants are processed line-by-line in the case of horizontal partition and column-by-column when vertical partition is performed.

The quadrature placement is also more efficient because of pin propagation. Thus, logic cells with common signals are placed in adjacent quadrants to reduce the wire length.

Area, delay and power were reported and compared in section 6. After the layout generation, layouts were inserted into CADENCE Virtuoso™ physical design environment where they were extracted using DIVA™ extractor and simulated with Spectre™ electrical simulator.

The used technology in theses examples was the AMS 0.35μm. The transistors width used in the layout generation were 4.00μm to PMOS transistors (W_P=4μm) and 2.00μm to NMOS transistors (W_N=2μm).

6. RESULTS

Table 2 shows the area occupation by circuits generated with Parrot Punch and TROPIC3. It is possible to note that there is a difference between the occupied areas of the layouts in a same technology. The main reason for these values is due to the routing strategy. In TROPIC3, routing is performed basically in routing channels between p and n diffusions while in PARRROT PUNCH routing is generated over the rows whenever it is possible. Results present between 33.2% and 47.5% of area reduction for the layouts generated with PARRROT PUNCH when compared with layouts generated with TROPIC3. The average gain in the area occupation is around 38%.

Table 2. Area occupation in TROPIC3 and Parrot Punch layouts (um2)

Bench	No. trans	P. PUNCH	TROPIC3	Gain (%)
C17	24	495.6	756.6	34.4
C432	804	20782.2	35443.2	41.3
C499	1556	54000.8	86224.3	37.3
C880	1802	59925.6	96014.4	37.5
C1355	2308	74613.9	111784.4	33.2
C1908	3482	116397.0	181575.6	35.8
C6288	10112	303376.8	578082.0	47.5

Table 3. Delay and power consumption in TROPIC3 and PARROT PUNCH layouts

Bench	No.	PARROT PUNCH		TROPIC3		Gain (%)	
	Trans	Delay	Power	Delay	Power	Delay	Power
C17	24	0.47	0.42	0.55	0.45	15.1	12.0
C432	804	0.78	15.31	1.07	16.73	26.9	9.3
C499	1556	0.31	18.61	0.42	17.8	26.0	-4.0
C880	1802	1.73	17.67	2.22	17.2	22.0	-2.0
C1355	2308	0.50	26.38	0.64	29.0	21.1	9.0
C1908	3482	2.67	35.1	3.66	45.56	27.0	22.0
C6288	10112	4.28	205.1	5.45	201.9	21.4	-1.5

Table 3 presents results given by electric simulation. Delay and power consumption of layouts are considered. Results related to delay of circuits show an average gain of around 22% when compared to TROPIC3 layouts. These results are obtained due to the difference of routing strategy and the attempt for optimizing drain/source areas on PARROT PUNCH strategy.

As reported in previously sections, the effort to optimize the layout is applied in all steps of the layout generation strategy of PARROT PUNCH. In addition, the reduction on the wire lengths obtained by the FOTC routing reduces the delay of the wires.

Power consumption in PARROT PUNCH layouts has an average gain of 4% over TROPIC3 layouts. In these circuits, it was applied the same simulation patterns and the switching activity is the same. Thus, the power consumption must be almost the same in layouts generated by both tools. The power consumption analysis for circuit C1908 shows a great difference between the layouts generated by PARROT PUNCH and by TROPIC3. This difference does not appear in other layouts. Verifying the simulation waveforms, glitches were found with more frequency in the layout generated by TROPIC3. It is believed that these glitches are responsible for the greater power consumption in the TROPIC3 layout.

7. CONCLUSION AND ON GOING WORK

This work presented an automatic full-custom generator called PARROT PUNCH. Layouts generated with PARROT PUNCH are characterized mainly by linear matrix style, full over-the-cell routing and 3 metal layers to implement all circuit connections. The previously mentioned features have led to an area reduction of 18.7% up to 44%, when compared to similar linear matrix layouts (TROPIC3). Also, delay and power were reduced.

The area occupied by layouts can still be reduced taking advantage on several metal layers available in current CMOS technologies. When full over-the-cell routing is target, the use of technologies with six or up to nine metal layers increases the success of this routing technique.

As long as most of commercial tools use standard cells based layout generation, we intend to perform practical comparisons between this style and PARROT PUNCH linear matrix style.

Some improvements are planned to PARROT PUNCH. Notably:

- Layout generation with more than three metal layers;
- Take advantage on the automatic layout generation to implement techniques of layout optimization such as gate sizing and buffer insertion.

REFERENCES

Bastian F., Lazzari, C., Güntzel, J. L., Reis, R. (2004). A New Transistor Folding Algorithm Applied to an Automatic Full-Custom Layout Generation Tool, PATMOS2004, *14th*

International Workshop on Power and Timing Modeling, Optimization and Simulation, Santorini, September 15-17, 2004. LNCS 3254 Springer. p. 732-741.

Cong, J. and Sarrafzadeh, M. (2000). Incremental Physical Design. In Proceedings of the 2000 International Symposium on Physical Design, pages 84-92. ACM Press.

Detjens, E. Rudell, R. Sangiovanni-Vinccentelli, A. and Wang, A. (1987). Technology Mapping in MIS. In *ICCAD*, pages 116–119.

Kim, J. and Kang, S. M. (1997). An Efficient Transistor Folding Algorithm for Row-based CMOS Layout Design. *DAC'97 – Design Automation Conference*, pages 456–459.

Lopez, A. and Law, H. S. (1980). A Dense Gate Matrix Layout Method for MOS VLSI. IEEE Transactions on Electron Devices, ED-27(8):1671-1675.

Moraes, F., Reis, R., and Lima, F. (1997). An Efficient Layout Style for Three-Metal CMOS Macrocells. In VLSI'97, pages 415-426.

Moraes, F. and Velasco, (2002). A. J. Deterministic Versus Non-Deterministic Placement Algorithms for Automatic Layout Synthesis Tools. In DCIS'02.

Reis, A., Reis, R. , Auvergne, D. and Robert, M. Library Free Technology Mapping. (1997). VLSI: Integrated Systems on Silicon, IFIP TC10 WG10.5 International Conference in Very Large Scale Integration, pages 303–314.

Roy, K. Mukhopadhyay, S. and Meimand, H. (2003). Leakage Current Mechanisms And Leakage Reduction Techniques in Deep Submicrometer CMOS Circuits. In *Proceedings of the IEEE*, volume 91, pages 305–327.

Sarrafzadeh, M., Bozorgzadeh, E., Kastner, R., and Srivastava, A. (2001). Design And Analysis of Physical Design Algorithms. In Proceedings of the 2001 International Symposium on Physical Design, pages 82-89. ACM Press.

Santos, C. L., Wilke, G., Lazzari, C., Guntzel, J., Reis, R. A. (2003). A Transistor Sizing Method Applied to an Automatic Layout Generation Tool. SBCCI2003. *16th Symposium on Integrated Circuits and Systems Design.* São Paulo, Septembre 8-11, 2003. p.303-307.

Uehara, T. and Cleemput, W. (1981). Optimal Layout of CMOS Functional Arrays. IEEE Transactions on Computer, C-30(5):305-312.