An I.P. Migration and Prototyping Strategy Using Transistor Level Synthesis

Alexis Landrault, Ludovic Pellier, Alexandre Richard, Christian Jay, Michel Robert, Daniel Auvergne

An I.P. Migration and Prototyping Strategy Using Transistor Level Synthesis

Landrault (1-2), L. Pellier (1-2), A. Richard (1), C. Jay (1) M. Robert (2), D. Auvergne (2).
(1) Infineon Technologies 06560 Sophia Antipolis, France
(2) LIRMM, UMR 9928 CNRS,University of Montpellier II

***Abstract*— Designers are today facing two challenges of conflicting objectives. In one hand they need to design for optimum performance and in the other hand they need to design in a minimum time frame ("time to market") while using the latest up-to-date available technology. It becomes necessary for the designers to very quickly prototype IP blocks for any given technology. This paper describes a new approach based on transistor level layout synthesis for CMOS IP cores rapid prototyping (~100k transistors).**

***Index Terms*— layout synthesis, virtual library, prototyping, technology migration, incremental optimization, transistor level, analytical modeling**

I. INTRODUCTION

STANDARD cell libraries have been successfully used for years. However with the emergence of new technologies and the increasing complexity of designs, this concept becomes less and less attractive. Most of the time, cells are too generic and not well suited to the block being created. As a result the final design is not well optimized in terms of timing, power and area. Depending on the target application, today's SoC may need to integrate the same IP block optimised using different criteria's (Platform based design [1], [2]). Standard approaches try to resolve this problem by providing the designer with multiple libraries containing cells optimized under specific electrical constraints. As a consequence, IP's generated using such a library will, most of the time, respect one specific constraint but will be over-fitted regarding other constraints. Moreover the development needed for every new technology and the maintenance of these standard libraries are very costly in term of time and in term of human resources. This can seriously impact time-to-market consideration during migration process.

Designers must be able to handle and quickly evaluate the design performance in the newest technology. Fast prototyping enables to increase the control on the trade-off between parameters such as speed, area and power consumption.

In this paper, we describe an investigation towards the use of transistor level layout synthesis to avoid employing a standard cell library and to allow designers to very quickly prototype "application-fitted" IP blocks for any given technology. It avoids the costs due to developing a standard cell library and mostly allows a quick evaluation of the performances of different flavours of a same IP.

The paper is organized as follows. In section II, we describe the standard cell approach and the newly emerging flows. In section III, we present our new flow based on transistor level layout synthesis and all the currently developed tools. And finally in section IV, we analyze the results obtained with this new "transistor level layout synthesis" tool called I^2P^2 before to conclude in section V.

II. DESIGN METHODOLOGIES

A. The standard cell based approach

In the "Standard Cell" flow [3], the layout of the design is directly generated from a behavioral description. First, it proceeds to a logical optimization and maps the Boolean equation to the target library. Then after placing and routing all the pieces of layout, representing the individual pre-characterized cells, the final layout is generated. The use of standard cells presents a well-understood trade-off. On the positive side, a standard cell library normally provides a set of pre-packaged functionalities. Each cell is speed and power characterized. Furthermore, the existing industrial EDA tools are relatively well adapted to this approach.

On the other hand some drawbacks appear:

- With new UDSM technologies where interconnects are of significant importance. It is more difficult to predict the cell drive and to satisfy the timing constraints. Various iterations are needed between each point-tool of the flow. This can seriously damage the flow convergence.
- In fact, it is well known that the quality of designs highly depend on the library that is being used [4] and of the variety of functionalities and sizes for each primitives gates [5]. Most of the designers recognize that standard cell libraries are populated with "too generic" cells [6], and that significant improvements can be achieved just by resizing some existing cells (drive continuity) or by adding a few customized cells (design dependency) in the initial library.

B. New alternatives

New emerging approaches are proposed by industrial companies so as to avoid these problems.

- Timing consistency problem is resolved by using tools built around a unified data model [7]: data requirements relative to the physical design can be reached quickly by the tools thanks to this unique data structure.
- Problem concerning "drive continuity" is partly resolved by using the "Liquid Library"[8] concept: In this approach, a static base library is used for design synthesis. Then the structural representation is placed and routed. Usually during the optimization step, we notice that some cells of the base library are not well suited to the design. Afterward needed design dependent cells are automatically generated by a "cell Factory" and are Engineering Change Order (ECO) placed (if possible) in the design.

Nevertheless, one main drawback still exists on these approaches: they are based on an already existing target library populated by a few hundred of logic gates specified to satisfy very specific area-performance tradeoffs. Consequently, the resulting IP block structures are forced to stay within the predetermined library precluding an optimal solution based on the total available design space of the available transistor structures.

Furthermore, the effectiveness of the standard flow methodology is highly dependent on the library development and process migration, which is costly.

III. AN ADAPTIVE LIBRARY CONCEPT BASED ON LAYOUT SYNTHESIS

We propose a standard cell independent based approach, working at transistor level. Theoretically, layout synthesis offers the possibility of overcoming deficiencies of the standard cell approach. The idea is that from a structural HDL description of a circuit, a synthesis tool would generate the optimized layout for the specified technology and timing constraints.

The main motivations for this approach are:

- First to avoid supporting the library and standard cell generation. This dependency is very costly in term of time (around 6 months to develop and characterize a library) and in term of human resources (around 5 people/month) while it only requires less than a day, using I^2P^2, to migrate to a new process by creating a new technology file.
- Secondly this approach enables to work at transistor level. Using the proposed methodology, each transistor can be individually sized or re-sized continuously (no discrete size limitations) at every stage of the flow.

Although this approach is not based on a standard cell library, we have to supply the logical synthesis commercial tool with a functionality set. We call this set the "virtual library". To each function, we associate a virtual cell that can be considered as a set of connected transistors (at symbolic level, no layout generated). Virtual cells appear as an interface between synthesis, place and route and the layout generation. The vast set of functionalities in the virtual library is expected to imply less transistor count in the design.

The proposed flow takes as input a behavioral description of the circuit function (VHDL or Verilog). It is composed of the following steps:

- During the first one a high-level logic synthesis and optimization, and a technology mapping are realized using the virtual library.
- A transistor level layout synthesis step, at which the layout of the circuit is generated, using a set of constraints, a structural description of the design and the virtual library as inputs. This includes placement, routing, timing analysis, optimization and physical layout generation.

We develop a tool which achieves this last step. As it is illustrated in the Fig.1, the "transistor level layout synthesis" tool starts from a structural description of the circuit and generates the layout.

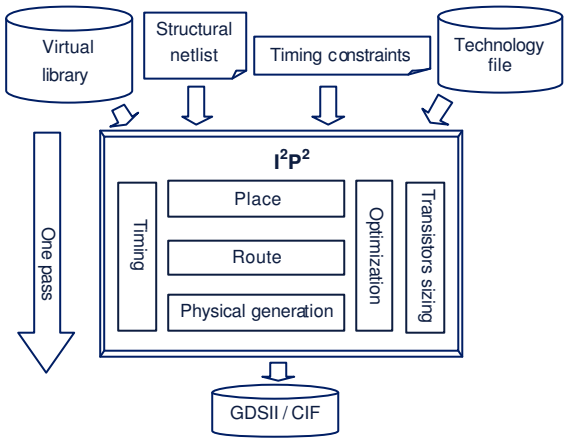

Fig.1. I^2P^2 description.

This approach is constituted of two major steps.

- The first one consists in creating the virtual cells from their functionality, resulting in an associated transistor network.
- The second phase achieves placement and routing based on physical estimations obtained from the targeted layout style. Timing analysis and optimization is realized during theses two steps. This feature enables, step-by-step, more optimized and refined performance results regarding initial design constraints. Information exchanged between these tools is made through the common data structure as it is shown on the Fig. 2. The physical layout of each row is then automatically generated, with full respect of the technology rules.

The main problems to be addressed in constructing such a flow are as follows. It is firstly necessary to provide a transistor netlist with an optimal number of transistors for the required functionality. Another challenge is to predict and

optimize the timing and power characteristics of the circuit before the final layout availability. The last challenge consists in generating a dense layout of the netlist, based on well-specified technology rules.

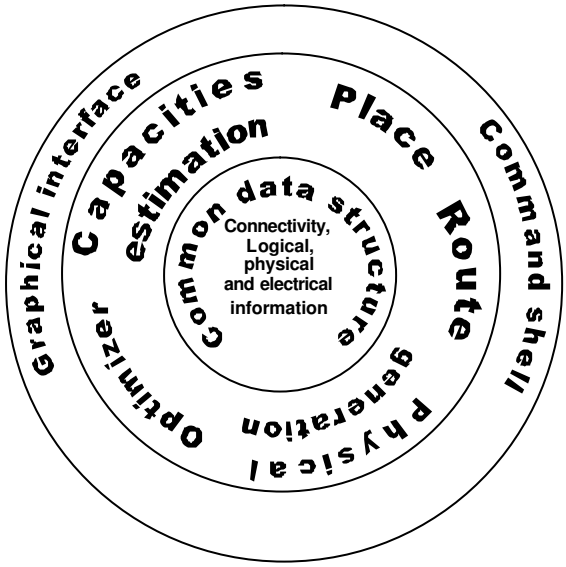

Fig.2. I^2P^2 architecture.

The main goal of our approach is to deal with all these problems and to converge to the best solution. Let us now consider in more detail the different steps of this layout synthesis approach.

A. Layout style

As virtual cells don't have predefined layout, it is necessary to deal with layout generation. The chosen style for the transistor-level generation is a variant of the popular "linear-matrix" [9]. The layout of the circuit is constructed by a sequence of NMOS and PMOS transistor rows. This style is principally characterized by the minimization of the space between the two N and P diffusion zones. Routing is realized over the diffusion zones (gain in terms of area) and avoids the use of metal 2 to save the porosity of each row and to facilitate the routing step.

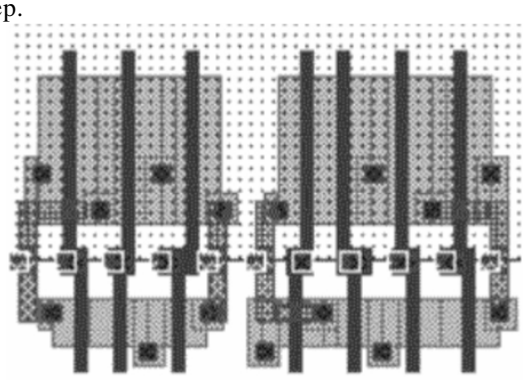

Fig. 3. Layout Style

This style has been chosen because it's perfectly adapted to software implementation (regular style with vertical placement of the poly grid) [10]. As described on Fig. 3, all the ports are placed at the center between N and P diffusions. Although our choice of topology was initially chosen to generate combinatorial cells, sequential elements such as DFF have been also fully generated using this layout style (more complex style should be used for sequential elements in order to reach quite the same area as standard cells). With respect to this style, a temporary layout can be very quickly generated for each virtual cell needed in the design. This layout used only for the prediction of the cell size and the port location will be then deleted (the final detailed layout of the design will be done only at the very end of the flow). These accurate estimations of the width, the height and the port position will then be used by place, route and timing analysis tools during all the block generation steps.

B. "Virtual Library" a new concept

The synthesis flow presented here is based on virtual libraries, using transistors generated (at symbolic level first to fill the data structure) on the fly instead of using a static set of pre-characterized cells. We have to keep in mind that, at this level, no layout is generated. Layout will be generated only for cell size prediction (temporary and then deleted) and at the very end of the flow (section III-C) at row-level.

The set of cells constituting the "Virtual Library" is mostly composed of complex gates. As a consequence the number of cells available is virtually unlimited (in terms of logical functionality). The constraints are only fixed by the target technology and performance limitations (maximum number of serial transistors).

In addition, as the final layout will be automatically generated, the driving capability of all these "virtual cells" can be continuously adapted, at the contrary of the discrete drive possibility offered in usual standard cell design.

1) Logic Optimization / Technology Mapping: the library free technology-mapping problem concerns the ability to use complex gates instead of using pre-characterized libraries. The main problem with "Virtual Libraries" is that the number of cells available can be very high and we can assume that the logic synthesis commercial tools are not especially well adapted to this kind of library.

In Table 1, we give an illustration of the number of different logic functions available in CMOS technology for a given limitation of N and P transistors in series

		Number of serial PMOS transistors			
		2	3	4	5
Number of serial NMOS Transistors	2	7	18	42	90
	3	18	87	396	1677
	4	42	396	3503	28435
	5	90	1677	28435	425803

Table1. Number of logic functions available for a given limitation of serial transistors.

For instance, for a maximum of four serial transistors (for each plan, N and P) we may dispose of a library with 3503 different functionalities. Moreover, we have a continuous sizing for each logical function that results in an almost infinite number of elements, considering the continuous transistor sizing facility.

Several approaches to realize technology mapping onto Virtual Library have already been proposed [11], [12], either as industrial products or in academic tools. Once the technology mapping on the virtual library cells has been completed, each logical function is transformed into the corresponding transistor network that is associated to a virtual cell. The network creation consists of generating a BDD for each virtual cell used in the design. Then an optimization of this BDD allows the generation of a transistor netlist for the circuit.

2) Timing & Power Modeling, Transistor Sizing: in this section, we will summarize the problem of timing analysis (already presented in one of our previous work [13]) using transistor sizing in association with the inherent problem concerning the transistors structure modeling. In [14], [15], [16] it has been shown that it is possible to develop a simple analytical model of the performance of CMOS gates with a good accuracy compared to electrical simulations. An efficient implementation of this model has been integrated. Concerning the problem of transistor sizing, [17] and [18] show that for most of the circuits, a local re-sizing of only a few selected transistors in the circuit can significantly reduce the power consumption.

C. Place and Route, Layout generation

The place and route and layout generation tools of the "transistor level layout synthesis" approach have been vastly presented in previous work [13]. In this section, we will briefly recapitulate the strategy of these tools.

1) Virtual cell Place and Route: placement and routing steps is performed from the "Virtual Cell" structural netlist representation. These operations use the predictive analytical models presented in the previous paragraph and is composed of three steps.

- First, a placement based on an iterative partitioning and global routing between each partition is performed [19]. Then cells are placed in each partition to allow rows creation. A global routing step is lastly done to generate virtual channels.
- Secondly, we create the symbolic view of each virtual cell (using the Euler trail solution [20]). Afterward the symbolic transistors rows are created then optimized (flipping, merging etc …) and routed ("inner Row" maze routing).
- Finally, all the remaining connections are routed by a virtual channel router using detail routing algorithms (constraint graphs and multi-layer maze router).

2) Layout generation: the layout generation of each row requires as inputs the symbolic view of each row and the technology rules for the targeted process. From a constraint graph, we obtain a "compacted" layout inherent of the "linear-matrix" style. The main advantage offered by this procedure is the technology independence (rules are described in an user file).

IV. RESULTS AND VALIDATION

A. The implementation

As said previously, in order to converge most efficiently during the optimization phase, the architecture of the software prototype is based on a unique data-structure done in C++. This prototype integrates plug and play facilities to ease the exchange of the different "engines" under development

B. Validation strategy

Validations have been done in a 0.13µm CMOS technology. The comparison has been done with respect to the standard flow using an In-House library. We have run our prototype on some industrial and ISCAS85 circuits. We proceed to various comparisons on transistor number (logic synthesis efficiency), transistor density (area), timing and power performances (transistor width) to evaluate our tool.

C. Logic synthesis and associated area

The technology mapping step has been completed using Design Compiler (Synopsys). The number of transistors required in the final netlist is the benchmark metric to evaluate the impact of using a virtual library for the mapping. Moreover we obtain a first estimation of the final circuit area, given by the sum of the cells area.

	Standard		**I²P²**	
Circuit	Transistor number	Area (mm^2)	Transistor number	Area (mm^2)
C3540	3860	0.0061	2998	0,0060
C5315	4712	0.0095	4518	0,0092
C6288	9876	0.0151	7424	0,0150
C7552	7008	0.0116	5902	0,012
Uniphy	30168	0.0473	13952	0,026
Mult32	47586	0.0785	40354	0.086

Table 2. Logical synthesis and associated area.

The results given in the Table 2 illustrate the decrease of the number of transistors required to realize the same circuit with our prototype, compared to a standard approach. Actually, the transistor number is decreased by an average of 27 %. Meanwhile the circuit area is not reduced in the same amount (i.e. equivalent area for smallest circuits). The reason is that automatically generated cells are not as small as standard cell.

D. Flow validation versus standard approach

To evaluate the whole flow of our approach versus the standard approach, we run a set of benchmarks on different circuits that should reach the same timing constraint for both approaches. The benchmark metric is the sum of the transistors width and the final area:

Circuit	Average transistors width (µm)	Area (mm^2)	Density (tr/mm^2)	Timing performances (ps)
C3540	0.957	0.008	374750	1653
C5315	0.869	0.013	362462	1736
C6288	0.832	0.020	371200	5389
C7552	0.855	0.016	368880	3474
Uniphy	1.003	0.065	214650	1773
Mult32	0.851	0.104	388020	9312

Table 3. Standard flow results

Circuit	Average transistor width (µm)	Area (mm^2)	Density (tr/mm^2)	Timing performances (ps)
C3540	0.357	0.0072	416390	1690
C5315	0.308	0.013	347540	1650
C6288	0.459	0.018	412450	4870
C7552	0.347	0.016	368880	2030
Uniphy	0.486	0.031	450060	1850
Mult32	0.368	0.119	339110	12930

Table 4. I^2P^2 flow results

As shown in the two tables above, the areas obtained with both methods are similar except for the Mult32 circuit. This degradation is due to the complexity of the place and route step which increases the interconnect length and so impacts the results in area and timing (developed place and route tools are not yet as efficient as commercial ones). The Fig. 4 represents an area and an average transistor width comparison for circuits satisfying almost the same timing constraint:

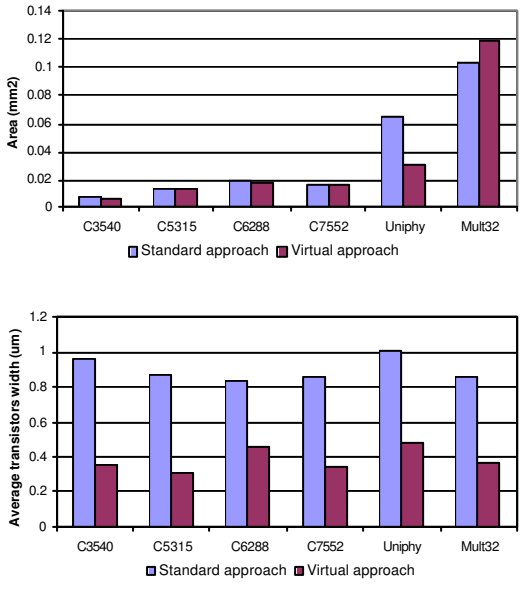

Fig 4. Area and average transistor width comparison.

Average transistor width comparison shows a significant advantage for the presented approach compared to the standard cell implementation (average reduction around 50% improvement). This result shows clearly that the possibility to resize smartly only needed transistor (specific and continuous transistor resizing) allows to significantly reduce the global width of the transistors. As a direct effect, we can expect to obtain a substantial power dissipation reduction compared to standard cell approaches (validation concerning power reduction is currently in progress using commercial power estimation tools).

Additionally, by reducing the number of transistors required to realize the same circuit (section IV-C), we can expect significant power reduction. Indeed, with new UDSM technologies, leakage current becomes of significant importance even if the transistor is not toggling.

E. Run time analysis

These results concerning area, timing and power are quite encouraging mostly if we consider the facility obtained in generating and migrating macro-blocks in very short time intervals as shown in Table 5:

Circuit	Transistors count	Run time (mn)
C3540	2998	3
C5315	4518	5
C6288	7424	8
C7552	5902	6
Uniphy	13952	15
Mult32	40354	65

Table 5. I^2P^2 tools: run time

These results, obtained on UltraSparc II 480Mhz machine, strengthen one of the main advantages offered by this approach: the possibility to quickly prototype IP block. Moreover, we remove the time associated to the development of a library and this enables fast technology migration every time a new technology is available.

F. Layout example and migration facilities

On the example given in Fig.5, a benchmark circuit done in a 0.13µm CMOS technology synthesized with our transistor-level layout generator is presented:

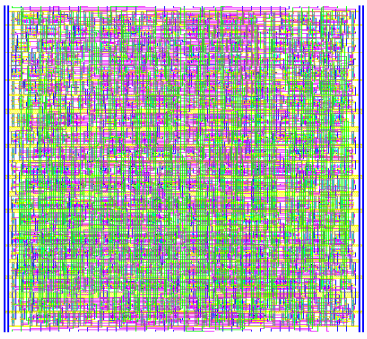

Fig 5. Layout of the c3540 (2998 transistors) circuit

We have validated the migration possibilities of our methodology by evaluating the performance of benchmark circuits in new process available. Using I^2P^2, it only requires few time to migrate to a new process.

Circuit	Density (tr/mm²) 0.18 um	Density (tr/mm²) 0.13 um	Density (tr/mm²) 90 nm	Time needed (min)
C3540	199870	416390	832780	3
C5315	173770	347540	728950	5
C6288	206220	412450	857170	8
C7552	190390	368880	725690	6
Uniphy	211393	450060	872000	15
Mult32	184270	339110	733710	65

Table 6. I^2P^2 migration from 0.18µm to 90nm.

Apart from the time necessary to update the technology file (half a day), the circuits have been migrated from one process to another in a very short time as shown in the Table 6.

G. Improvement and future work

We pointed out previously that the physical layout generation can strongly impact the final circuit. In fact the results of the comparison between the area of individual virtual cells compared with the standard cells (for the same functionality and the same technology) show almost equivalent results for combinatorial cell but prove a significant difference for sequential element. The high complexity of the intra cell interconnections of the flip-flop and our choice of topology (initially chosen to generate combinatorial cell) can explain this difference that can be overcome by revisiting some of the algorithms of the physical generation.

Moreover, the first validations of our place and route tool versus Magma's highlight some light degradation on complex circuits. Some place and route algorithms should be improved so as to reach the same area as we could get using industrial tools.

V. CONCLUSION

In this paper, we have presented an original alternative to the classical standard cell based layout synthesis. By using a "virtual cell library", we obtain a physical generation at the transistor level, integrating the different steps of physical layout generation, performance estimation and optimization. This may give great facilities in quickly evaluating and prototyping different flavors of IP blocks by using the latest available technology and moreover it removes the time and the prohibitive costs associated to the development of a standard library. We show that our tool which can be seen as an "IP Prototyper" is able to handle simple blocks (order of complexity ~10K to 100k transistors).

REFERENCES

[1] A. Sangiovanni-Vincentelli, "Platform-Based Design: A Path to Efficient Design Re-Use", First International Symposium on Quality of Electronic Design, 20 - 22 March, 2000, San Jose, California.

[2] M. Reinhardt, "Implementing a Migration-Based IP-Reuse Strategy", Electronic Engineering Times, May 1999.

[3] D. McMillen, M. Butts, R. Composano, D. Hill, T.W. Williams, "An Industrial View of Electronic Design Automation", IEEE Transactions on Computer, Vol. 19, No 12, December 2000, pp. 1428-1448.

[4] K. Keutzer, K Scott, "Improving Cell Library for synthesis", Proc. Of the International Workshop on Logic Synthesis, 1993.

[5] K. Keutzer, K. Kolwicz, M. Lega, "Impact of Library Size on the Quality of Automated Synthesis", ICCAD 1987, pp. 120-123.

[6] P. de Dood, "Approach makes mostof synthesis, place and route - Liquid Cell ease the flow", EETimes, September 10, 2001, Issue: 1183.

[7] O. Coudert, "Physical design closure" Monterey design system", Design Automation Conference 2000.

[8] P. de Dood, "Putting Automated Libraries into the Flow", EETimes, May 2001.

[9] A.D.Lopez, H.S.Law, "A Dense Gate-Matrix Layout for MOS VLSI", IEEE Transactions on Electron Devices, Vol. ED-27, No. 8, August 1980, pp. 1671-1675.

[10] F.Moraes, R.Reis, L.Torres, M.Robert, D.Auvergne, "Pre-Layout Performance Prediction For Automatic Macro-Cell Synthesis", IEEE-ISCAS'96, Atlanta (USA), Mai 1996, pp. 814-817.

[11] P. Abouzeid, R. Leveugle, G. Saucier, R. Jamier, "Logic Synthesis for Automatic Layout", Proc. Of EUROASIC, pp. 146-151, 1992.

[12] A. Reis, R. Reis, D. Auvergne, M. Robert, "The Library Free Technology Mapping Problem", IWLS, Vol. 2, pp. 7.1.1-7.1.5, 1997.

[13] A.Landrault, L.Pellier, A.Richard, C.Jay, M.Robert D.Auvergne, " A design approach of reusable cores based on transistor level synthesis ", VLSI-SOC 2001, Montpellier (France), oct 2001, pp. 298-303.

[14] J.M. Daga, D. Auvergne, "A comprehensive delay macro-modeling for submicron CMOS logics", IEEE Journal of Solid State Circuits, Dec 1998.

[15] F. Brglez, H. Fujiwara, "A Neutral Netlist of 10 Combinatorial Benchmark Circuits and a Target translator in Fortran", Int. Symposium on Circuits and Systems, June 1985.

[16] D.Auvergne, J.M. Daga, M. Rezzoug, "Signal transition time effect on CMOS delay evaluation", IEEE trans. on Circuits and Systems: Fundamental theory and applications, vol.47, n°9,pp.1362-1369, sept.2000.

[17] S. Cremoux, N. Azemard, D. Auvergne, "Path Resizing Based on Incremental Technique", Proc. of ISCAS98.

[18] S. Cremoux, N. Azemard, D. Auvergne, "Path Selection for Delay and Power Performance Optimization", Proc. of SAME98, pp. 48-53, 1998.

[19] C. M. Fiduccia, R. M. Mattheyses, "A linear-time heuristics for improving network partitions.", Proceedings of the 19th Design Automation Conference, pages 175-181, 1982.

[20] M.A. Riepe, K.A. Sakallah, "Transistor level micro-placement and routing for two dimensional digital VLSI cell synthesis", University of Michigan, Ann Arbor ISPD '99 Monterey CA USA.