3

BONSAI : A pragmatic approach to Logic Synthesis and Formal Verification

H.N. Nguyen and L. Ducousso

Hardware Development Paris-Angers
***BULL S.A.** 78340 Les Clayes-sous-Bois, FRANCE*
Tel : +33.1.30.80.60.19, email : H.N.Nguyen@frcl.bull.fr

Abstract

This paper describes the large scale application of logic synthesis and formal verification to the design of the CPU and caches of the high-end series of the Bull DPS7000 mainframe family. The logic CAD suite used for supporting the design of this system proved its efficiency on very complex integrated circuits. The key feature of this logic design environment is the methodology that integrates a set of logic synthesis and formal verification techniques to build an effective logic-design system to support the design of high-performance circuits.

Keywords

Computer-Aided Design, Logic Synthesis, Formal Verification.

1. INTRODUCTION AND MOTIVATION

This paper describes the original computer-aided logic design approach used in developing the high range version of the Bull DPS7000 mainframe family. The task at hand was the design of a family of new high performance mainframes, based on CISC architecture, using a submicron CMOS technology in a semi-custom, very large scale integrated chip technology. Aggressive performance goals and shorter design time than prior projects of similar complexity were our main objectives.

Design tools development at Bull S.A., during the previous several years, demonstrated that a comprehensive system-scale verification process was feasible, and that our goal should be a first-pass design capable of running the proprietary GCOS7 operating system.

At the start-up phase of the development project and after considering a number of competitive alternatives, it appeared that the design objectives could be achieved by a combination of adopting a well-established structured semi-custom chip design methodology, developing and using logic synthesis and formal verification to improve the design time and provide reliability, and by establishing a closer communication between the designer and the CAD developer.

G. Saucier et al. (eds.), *Logic and Architecture Synthesis*
© IFIP International Federation for Information Processing 1995

The purpose of this paper is to present the BONSAI logic design system and its associated methodology developed to support the synthesis and formal verification of high performance circuits (sections 2 and 3). Our efforts were mainly dedicated to the timing improvement aspects which appear both in the ability to insert customized gates in the standard cell synthesis process and in the links with layout as discussed in section 4. New formal comparison techniques were also developed to handle more efficiently large and complex logic.

The development of the Logic Synthesis and Formal Verification tools was supported in part by the Jessi-AC8 Project.

2. A STRUCTURED CUSTOM CHIP DESIGN METHODOLOGY

The overall system design flow is illustrated in Figure 1.a.

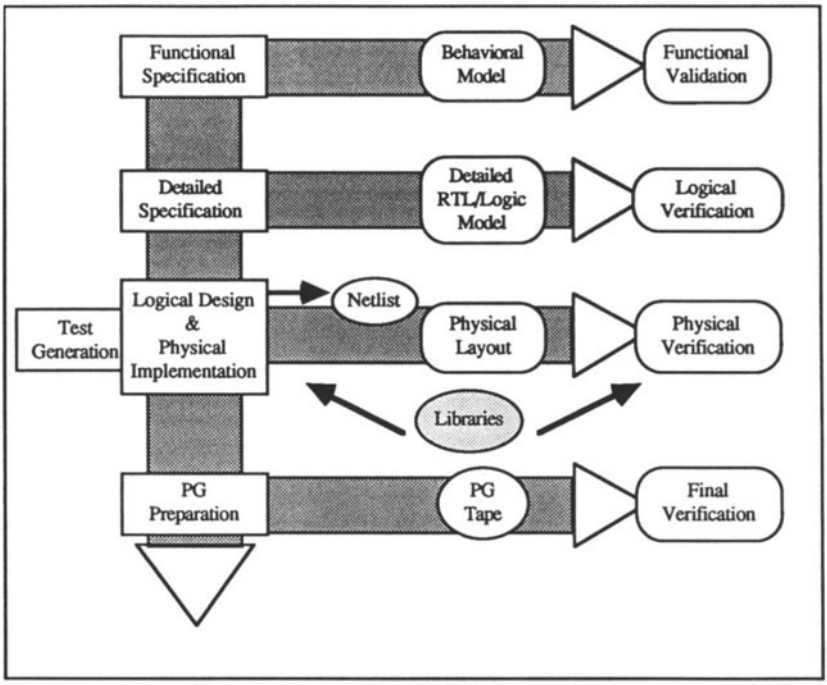

Figure 1.a : System Design Flow

2.1 Hardware Modelling and Validation

In many ways, the design process for the earlier VLSI DPS7000 mainframe computer set the tone for all subsequent VLSI designs at Bull. The process was characterized by massive simulation of a full system model, running in all modes of operation. Two types of simulation models were used (cf. Figures 1.a and 1.b) : the first type was designed as a high-level software breadboard used to develop and check out the microcode before the chip hardware was available. The second type was developed as a relatively detailed register transfer level model of the actual physical partitions and design concepts of the chip themselves. This model was dedicated to pilot the synthesis tool suite or used directly by the logic designers to develop the circuit-level representations of parts of the design. The models were described in a proprietary hardware description language oriented toward synchronous design (called LDS), where object (signal, latch) names follow an imposed naming convention that guarantees unique identification and improves readability. Due to the large amount of data, a Zycad SDE hardware accelerator was used for non-regression simulation.

2.2 Hardware Design Methodology

The hardware design methodology is targeted for a mixed custom and semi-custom design environment (Figure 1.b). In the adopted top-down approach, the design starts at the highest level and progresses toward lower levels of the hierarchy and the upper level defines the requirements of the design for the lower levels. Performance tuning is achieved during the planning and design phase, thus providing the ability to perform a one-pass design (indeed, two passes are necessary to meet all the performance constraints). At each level of the abstraction, validation is performed by simulation of the whole model of the CPU.

2.3 Logic Synthesis Requirements

The CPU and caches of the DPS7000 system are composed of VLSI circuits which are decomposed into macro-blocks during the early floor planning phase. While structured logic blocks (e.g. operators of the data path, ROMS) are either designed manually or using general purpose [Nguyen 90] or specific module generators, logic synthesis is applied to the design of the control parts (control logic blocks and micro-sequencers) .

The logic style of the synthesized blocks is a mixed regular and random logic associated with a multiclocking and multiphase scheme. The usage of multi-input latches, precharged and tristate signals represents another characteristics of the synthesized logic. In addition to these considerations, the main design constraints for these blocks are drastic timing constraints and narrow area margins.

As a result of these considerations, synthesis is a highly iterative process and our main concern has been to ensure a rapid and smooth convergence of this process toward the desired result. This implies the ability of the logic synthesis system to take into account hand tuning needed to improve logic speed in problem areas (e.g. the use of non-standard gates as illustrated in Figure 2). In addition, logic design was synchronized with the semi-custom physical design process for complete design verification.

2.4 Formal Verification Requirements

Checking plays a significant role in our methodology. Among the many different checking functions, formal techniques are used to perform logical-to-physical checking and model verification. Logical-to-physical checking is intended to verify that the physical implementation of the blocks matches their logical descriptions while model verification is used to prove the equivalence of models (e.g. different representations or versions of a block) and to check some basic properties (e.g. scan path connectivity, exclusive control signals at a multiplexing node, ...etc.). The main requirements for the usage of formal verification is its ability to handle large and complex logic (e.g. manipulation of Boolean expression with hundreds of variables in its support).

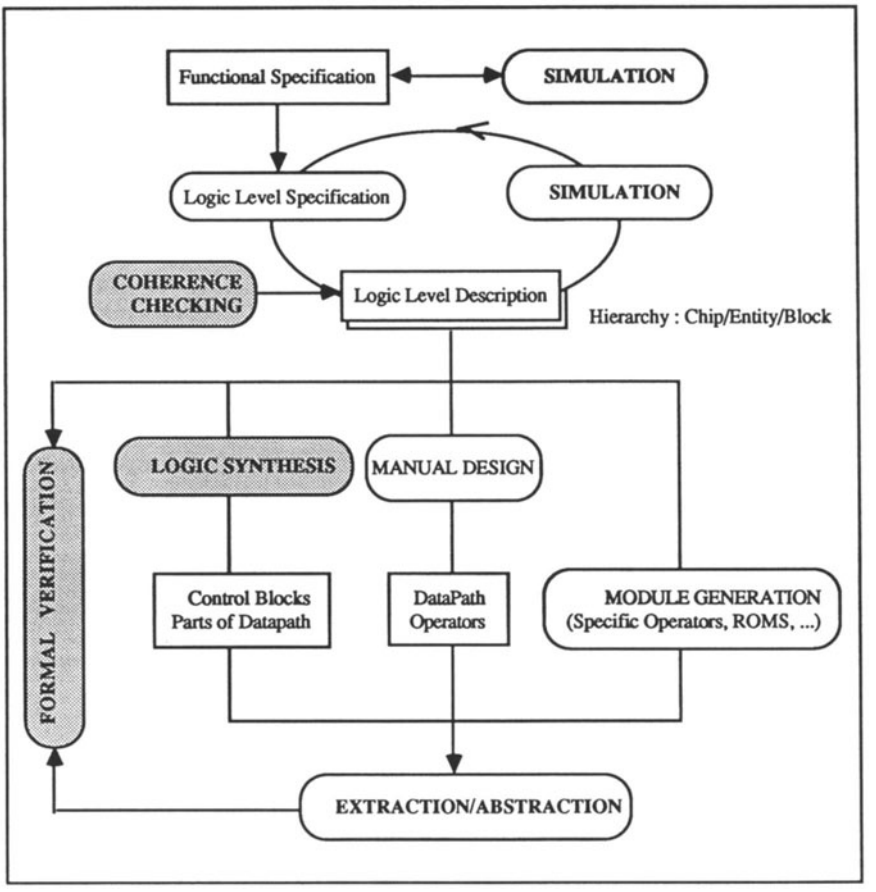

Figure 1.b Logic Desin Methodology

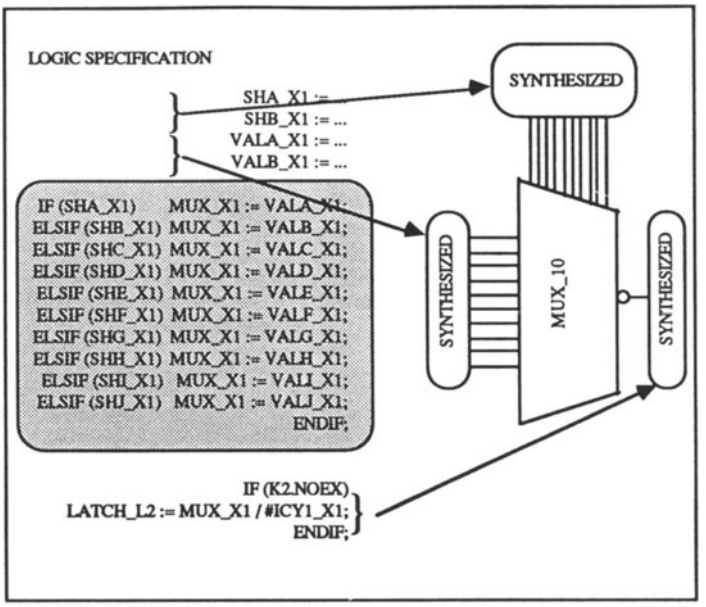

Figure 2 : Introduction of custom gate to improve timing

3. FEATURES OF THE BONSAI SYSTEM

We consider the problem of logic synthesis which is that part of the design path that starts with a RT/Logic specification of hardware that is to be mapped on a library of predefined primitives and ends with a netlist of gates which, after the physical implementation phase, satisfies both the timing and area constraints. The availability of a set of tools covering the whole logic synthesis path and addressing verification issues is essential for a fast design of VLSI circuits.

The following sections describe some of the major components and features of the BONSAI system.

3.1 Methodology of Usage

Logic synthesis involves the transformation of RT/Logic level specifications to a technology-dependent gate-level description. In order to cope with the timing and area constraints, the imposition of parts of the design as structural constraints is allowed during the synthesis process as shown in Figure 2. The structural constraints are mainly used for special purposes such as :

- Introduction of customized gates (e.g. large switch multiplexors, combination of latch and tristate gate) to improve the timing performance on critical paths,
- Hardware duplication or introduction of additional stages (logical layers) to reduce the overall circuit delay.

In presence of structural constraints, the system automatically checks for the coherence of the two overlapping views of the design and synthesizes the remaining part of the functional specification while taking into account the already designed part. Thus the convergence of the design process is ensured after a limited number of (re-)synthesis steps. This represents the pragmatic aspect of our approach in solving difficult timing problems : capture of the designer's knowledge instead of tuning multiple parameters of the algorithm until obtaining the desired results. In addition, the structural constraints are input as complement to the logic specifications, thus avoiding to rewrite the initial specification which is an error prone task (validation of the model is performed by simulation of the whole CPU model).
As a consequence, Formal Verification is built-in the system : it is used both inside the synthesis procedure to check for the coherence of the specification and the structural constraints, and independently to prove that the input models used for massive simulation are equivalent to the low-level descriptions generated by the synthesis system or abstracted from the layouts (Figure 3).

3.2 Overview of the BONSAI System

3.2.1 Features

The originality of our logic design system lies in the strategy that combines synthesis and verification techniques. The main feature of the system is built on the concept of a Boolean Network as detailed in [Hachtel 87][De Geus 87]. However, it is necessary to extend those definitions in order to type the nodes and to handle sequential components although only the combinational parts interconnecting these components are considered in our works.

Definition 1
A Boolean Network N is a 3-uple (F,PO,M) where $F = \{Fj / j=1,...,m\}$ is an incompletely specified function. With each Fj is associated a logic variable vj in the set $V = \{vj / j=1,...,m\}$ called the intermediate variable set. The specified Primary Output set PO $\{1,...,m\}$ identifies the subset $\{vj / j \in PO\}$ of the observable outputs of the Fj while the memory set $M \subset \{1,...,m\}$ defines the subset $\{vj / j \in M\}$ of the memory variables. In our case, these memory variables are used to model latches. The behaviour of a latch with input d, output q and control c is defined by the following function :

if c(t+1) then q(t+1) := d(t+1) else q(t+1) := d(t) endif;

Definition 2
A representation of a Boolean Network is a 4-uple (F,DC,PO,M) such that $\{Fj / j=1,...,m\}$ (resp. $\{DCj / j=1,...,m\}$) is a set of m given representations of the on-sets (resp. DC don't care sets) of Fj.

Let us note the support set of Fj by SUPP(Fj), then we will have $t = / \cup j \ SUPP(Fj) / > m$. The last t - m logic variables vj, j > m have no associated Fj and are identified as the Primary Input set PI. In addition, we can state without lost of generality that : for all j=1,...,m SUPP (DCj)$\subset$ SUPP(Fj).
Note that the don't care set plays an important role in the verification process : For an operator of a data path, the set of control signals is most of the time generated by a control logic block and therefore it is not complete. Besides, the DC set is also necessary to handle tristate signals.

Notation : In the following sections, we will note $O = PO \cup M$ and $I = PI \cup M$.

Definition 3

For each variable vj, j=1,...,m+n we define the Fan In set FIj as follows :

if j > m then FIj = ϕ else FIj = {k / vk $\in$ SUPP(FIj)}

Similarly, The Fan Out set FOj is defined as : FOj = { k / vj $\in$ FIk }The transitive Fan In and Fan Out sets associated to a variable vj are defined in the same way as follows :

TFIj = { k / $\exists$ a sequence [k1,...,kr] such that k1 = k, kr = j

and for q = 1, ...,r-1, q $\in$ FIq+1, and for q, 1<q<r, q$\notin$ M }

TFOj = {k / j $\in$ TFIk}

3.2.2 Functions

As diagrammed in Figure 3, the main modules of the BONSAI system are the following :

• Assistance to specification : The system verifies that the Boolean Network is well-defined, i.e. satisfying the following criterions :

i/ $\forall$ j =1,...,m, j $\notin$ TFIj

ii/ $\forall$ j $\in$ I, $\exists$ i $\in$ O such that i $\in$ TFOj.

iii/ The control signals at all nodes of M are mutually exclusive

These rules express the facts that (i) there is no functional loop and (ii) every intermediate variables or primary inputs are effectively used, so checking these criterions allow to detect errors in writing the model.

In addition, it provides consistency checking and symbolic simulation to detect incoherence in the specification such as unused or unassigned variables, ill-defined conditional assignment or multiple assignments exclusivity of the control signals at a multiplexing node.

• (Re-)Synthesis with structural constraints : The synthesis procedure is classically implemented in two steps : Logic minimization and technology mapping. While logic minimization adopts a symbolic method to transform the initial Network into a primary and non redundant Boolean Network [Hachtel 87], the technology mapping is based on a mixed algorithmic and rule-based approach. The rule-based mapping procedure is used improve the netlist produced by the algorithmic step and to solve local problems such as fitting fan in/Fan out, synchronization of latch controls, etc ...

• Formal Proof : The role of the Formal Prover is to check for the equivalence of two Boolean Networks defined as follows :

Boolean Networks N1=(F1,PO,M) and N2=(F2,PO,M) are said to be equivalent if : $\forall$ j $\in$ O (= PO $\cup$ M), z1(j) = z2(j) on the definition domain of j, where z denotes the IO map a Boolean Network (as defined in [Hachtel 87]) with the following extension z : I --> O instead of z : PI --> PO).

The definition domain of a node is defined as the complement of the set ($\cup$ DC(k) / k $\in$ TFIj).

Practically, the adopted approach is based on a canonical representation of Boolean functions called Compacted Decision Diagram [Nguyen 92] which is an outgrowth of the Binary Decision Diagram [Bryant 86]. Used in conjunction with a swap strategy, this representation allows to handle efficiently large designs by reducing the memory occupation and improving the computation time.

• Netlist Optimization : The improvement of the quality of the synthesized netlist taking into account the interconnect delays extracted from the layout is described in details in the next section.

• Schematic Generation : Most of the time, large schematics are not readable. To tackle this problem, the schematic generator provides local views (eg. path between two nodes, logical cone whose output belongs to O and inputs are elements of I, and introduces timing barriers (represented by memory nodes or I/O ports) to decompose the network into successive logical slices .

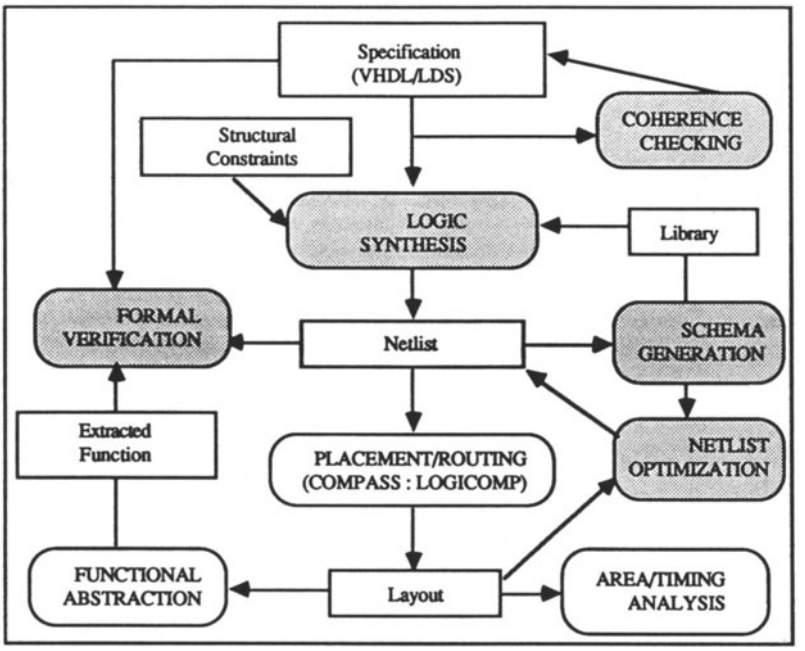

Figure 3.a : Features of the BONSAI System

3.3 Implementation

The BONSAI system is implemented in Le_Lisp [Chailloux 90] which is a Lisp dialect enhanced with object-oriented facilities, and runs on a large range of workstations (BULL/DPX5000, BULL/DPX20, IBM/RS6000, SUN3/4).

4. LOGIC SYNTHESIS LINKS WITH LAYOUT

Because interconnect delays become very critical as feature sizes shrink and tends to become predominant (especially in the design of submicron ASICs), some of the benefits provided by synthesis at the front end design cycle are lost during the layout phase : There is a growing disparity between the estimates done in synthesis and the real, post-route delays. As a consequence, a lot of time is spent in fixing timing problems after the layout is done.

4.1 Analysis of the Interconnect Delays

Measures of the influence of routing on the timing are represented in Figures 4.a, 4.b and 4.c : Figure 4.a shows the distribution of the wire length as a function of the number of pins of the nets while in Figure 4.b we have represented the distribution of the ratio wire capacitance / Fanin . Finally, Figure 4.c represents the variation of the mean value of this ratio versus the size of the block for a set of small blocks. We also note that those measures vary in a random manner when the ratio of the block (height / width) diverges from 1. From these measures, we can deduce that it is almost impossible to a-priori predict the interconnect delays as a function of the size of the block and of the number of interconnected pins during the synthesis procedure. As a consequence estimation performed at the synthesis step is not accurate especially when the underlying technology is submicronic.

4.2 Links Between Synthesis and Layout and Related Works

There are two different approaches commonly adopted by logic synthesis vendors (e.g. SYNOPSYS Design Compiler 3.0 and COMPASS ChipPlanner) :

- Top-down approach : Constraints (net priorities, path delays, net capacitance and timing) are passed from synthesis to a timing-driven layout tool. This method appears to be of limited success due to the limitations of the layout tools : Only a "reasonable" (< 10) number of constraints can be handled efficiently (e.g. in our application, only net priorities are accepted by the COMPASS/Logicomp tool as constraints and the results become rapidly unpredictable when the number of constraints is too large).
- Feedback Approach : Resynthesis with back-annotated delay values face the difficult problem of convergence : Obviously with the change of the logic structure, the back-annotated values become wrong and the placement/routing heuristics will produce others timing discrepancy.

4.3 In-Place Optimization

Based on the previous analysis, we have finally adopted the pragmatic approach commonly called "in-place optimization" : when timing constraints are not met due to interconnect delay, the netlist optimizer (Figure 3) automatically replace cells by more powerful ones without changing their footprints, thus leaving unchanged the netlist and the placement/routing of the block [Nguyen 93]

Our implementation of this procedure has the following characteristics :

- Timing Model : Delay = Gate's Inherent Delay + F * (Σ Input Load + Wiring Delay). While the wiring delay is estimated as a function of the interconnected pins during the synthesis step, it is extracted from the layout in the netlist optimization procedure.

• Relaxation of the Fan in / Fan out constraints to allow convergence of the procedure. Effectively, a more powerful gate yields more input capacitance.
• Each logic gate can be mapped on a family of cells (having the same width), the most powerful cell of which is reserved exclusively to the in-place optimization procedure (Figure 5).

5. RESULTS

The BONSAI system has been successfully used in the Auriga2 development project for the design of control logic blocks of the circuits composing the CPU and the shared cache. One chip of the shared cache has been entirely synthesized (i.e.. including the datapath) . The results are summarized in Table 1. In addition, we have performed the following statistical measures on the improvement of timing performance :
• The structural constraints represent about 5 % of the total number of synthesized gates,
• The timing improvement due to the in-place optimization module is estimated about 10%.

SYNTHESIS		
CPU	Shared Cache	
3 Processors + Private Cache	SHA chip	SHD chip
Synthesis of all Control Blocks (Total # 200 K Transistors)	Synthesis of Control Blocks (Total # 50K Transistors)	Entirely Synthesized (Total # 100K Transistors)
FORMAL VERIFICATION		
Verification of all the Blocks (control logic blocks, datapath operators, ROMS) (# 4 M Transistors) Verification of the Scan Paths of all entities		

Table 1 : Achieved Results

Other applications of BONSAI include the design of a processor which allows communication between cluster of the Escala machine (designed at Bull) through serial links. The hardware part of this project is a chip containing about 1.2 M of transistors, designed in a CMOS 0.35 micron technology and entirely synthesized and formally verified with BONSAI.

6. CONCLUSIONS

We have presented in this paper an overview of the logic synthesis and verification system BONSAI developed to support the design of the Auriga2 CPU and caches of the DPS7000 mainframe. The originality of our work lies in the methodology that integrates a set of state-of-the-art design tools to make an effective logic-design system. A major strength of this methodology is the manner in which it allows rapid convergence of the design process in presence of drastic timing constraints and its ability to handle large and complex logic. This methodology is targeted for the semi-custom design environment, where a high degree of flexibility is required in the design process.

Current and future works include the development of tighter links with floorplanning and the evolution toward higher level of abstraction.

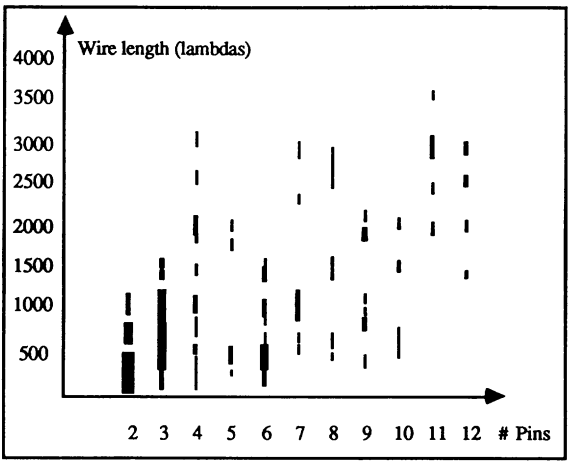

Figure 4.a : Distribution of the wire length

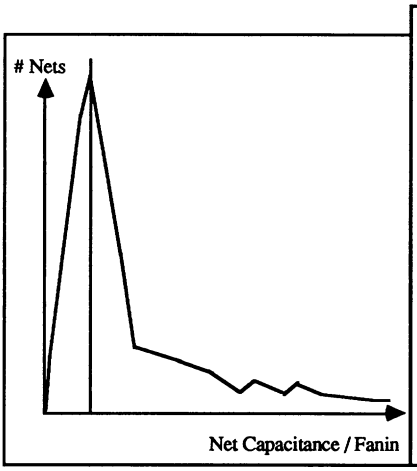

Figure 4.b : Distribution of net capacitance / fanin

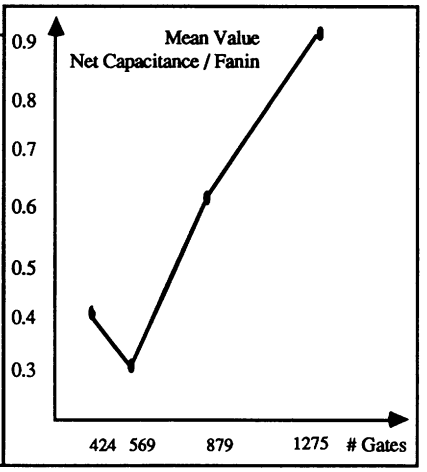

Figure 4.c : Mean value vs the size of the block

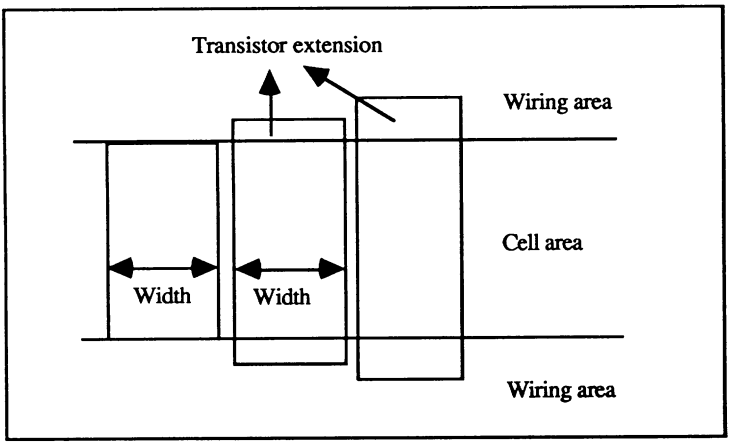

Figure 5 : Usage of cell for in-place optimization

References

[Brayton 90] R.K. Brayton, GD. Hachtel and A.L. Sangiovanni-Vincentelli : "Multilevel Logic Synthesis", Proceedings of the IEEE, Vol. 78, N° 2, pp. 264-300, Feb 1990.

[Bryant 86] R.E. Bryant : "Graph-based Algorithms For Boolean Function Manipulation", IEEE Transactions on Computers, 35(8), pp. 677-691, 1986.

[Chailloux 90] J. Chailloux, M. Devin et als, "Le_Lisp Version V15.23 : Manuel de Référence", I.N.R.I.A. , July 1990.

[De Geus 87] A.J. De Geus and D.G. Gregory : "The SOCRATES Synthesis and Optimization System", in Design Systems for VLSI Circuits, Martinus Nijhoff Publishers, pp 473-498, 1987.

[Hachtel 87] G.D. HACHTEL and R.M. JACOBY, "Verification Algorithms for VLSI Synthesis", in Design Systems for VLSI Circuits, Martinus Nijhoff Publishers, pp 249-300, 1987.

[Nguyen 90] H.N. Nguyen and L. Ducousso, "PLAYL : A General-Purpose Data Path Assembler", Proc. EUROASIC'90, Paris May 29-June 1, 1990.

[Nguyen 92] H.N. Nguyen, L. Ducousso, M. Thill and P. Vallet, "A Pragmatic Approach to the Automation of the Logic Design Process", Proceedings of the EDAC'92, Brussels, February 1992.

[Nguyen 93] H.N. Nguyen, L. Ducousso and M. Thill, "Optimal Use of Cell to Improve Timing Performance in Logic Synthesis", PATMOS'93, La Grande-Motte, October 1993.