HIERARCHICAL STATIC TIMING ANALYSIS FOR CMOS ULSI CIRCUITS

Karim Dioury *Alain Greiner* *Marie-Minerve Louërat*

LIP6, ASIM Department
University Pierre et Marie Curie (Paris 6)
Bldg 55-65, room 210,
4 place Jussieu,
75252 Paris Cedex 05, FRANCE
Phone: (33) 1-44 27 30 43,
e-mail:{Alain.Greiner,Karim.Dioury,Marie-Minerve.Louerat}@lip6.fr

ABSTRACT

With the advent of deep submicronic technologies, static timing analysis has revealed itself as the only feasible method for the timing verification of circuits, which can nowadays contain up to several million transistors. Nevertheless, the flat, transistor level, analysis exceeds the memory capacity of most workstations. To alleviate this problem, we have defined a method based on the hierarchical partitioning of the design phase. The propagation times within a circuit are represented using a multi-level hierarchical timing view that has allowed us to develop the hierarchical timing analysis tool HiTas as well as the interactive path browser XTas.

1. INTRODUCTION

The National Technology Road-map for Semiconductors projected that by the year 2001 feature size will shrink to 0.18μ and chips may contain tens millions of transistors [5].

Timing analysis is an important step in the verification of digital VLSI circuits. With the advent of deep submicronic technologies, static timing analysis has revealed itself as the only feasible method for the timing verification of circuits [6, 7, 9, 10, 14, 15, 17]. Nevertheless, this method can generate an excessively large amount of data.

To alleviate this problem, we have defined a method based on the hierarchical partitioning of the design phase. The propagation times within a circuit are represented using a multi-level hierarchical timing view. The delays associated to gates and RC networks are represented by a causality graph, in which the nodes correspond to events on the signals, and the arcs correspond to the delays between two events on two signals.

A method for the traversal of the causality graph has been defined, which limits the search for paths between connectors and registers to a small part of the graph.

This multi-level hierarchical timing representation allowed us to develop the hierarchical timing analysis tool HiTas as well as the interactive path browser XTas. Experimentations prove that our approach can deal with multi-million transistors chips.

The paper is organized as follows : Section 2 presents what is expected from timing analysis of electronic systems, Section 3 defines the multi-level hierarchical timing view and the path search through the hierarchy, Section 4 presents a factorization method used to reduce the size of the causality graph, Section 5 gives some experimental results. Section 6 concludes the paper.

2. TIMING ANALYSIS OF ELECTRONIC SYSTEMS

2.1. Objectives

Given a clock period and input signal arrival times, the analysis of sequential designs requires checking all set-up and hold time violations on all internal registers and external output signals. Complete sequential circuit verification must ensure that *all* delay paths lie between a lower and an upper time bound. Thus the timing analysis phase must provide the worst delays (shortest and longest) between all circuit *stability terminals* (inputs, outputs and internal registers). To handle multi-million transistor circuits, the delay model must be accurate enough to handle short channel CMOS transistors and interconnect RC networks, but it should be simple enough to reduce computation times and memory occupation.

The main goal is to obtain all the critical paths between stability terminals, with all the gate delay details for each path, to enable design optimization and error correction. Another requirement is to provide - if necessary - all parallel paths to a given critical path between two stability terminals, to enable false path detection [6, 7]. This detailed information can be provided by the traditional static timing analysis [10] approach for circuits that do not exceed about 100k transistors. With the emergence of systems on chip and the resulting massively complex chips, the volume of timing data can no longer be handled by a flat analysis [12, 14, 15].

The design of complex chips uses a decomposition, consisting of breaking up the design, at a given level of the hierarchy into components that can be designed almost independently. The aim of the analysis made here is to handle the timing verification of those multi-million transistor chips by using the same decomposition that was used for the design. The static timing analysis is performed with a flat approach on each of the leaf block level of the hierarchy, the results are subsequently used by the hierarchical analyzer (Fig. 1).

2.2. The Timing Graphs

The flat analysis of leaf block cells (Fig. 1) is performed with the static timing analyzer Tas [2]. Tas works on a post layout, extracted *transistor level* netlist. Three graphs are built, which are the basis of the timing or stability analysis.

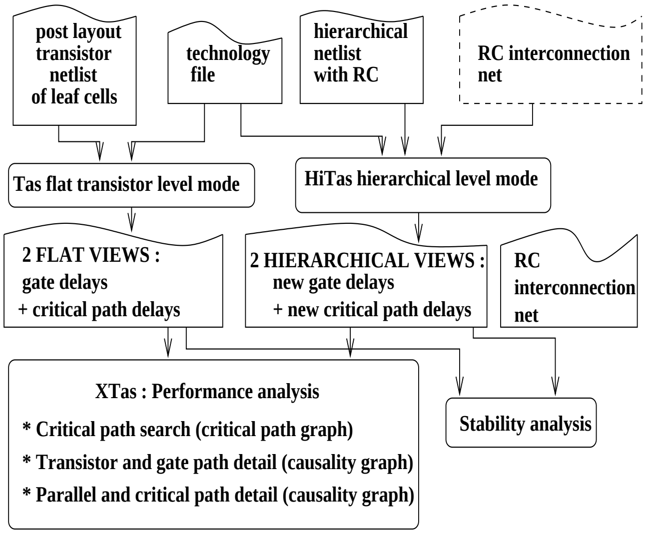

Figure 1. Flow graph of the hierarchical timing analysis

1. **The dependency graph.** From the extracted transistor netlist, a *pseudo-gate* netlist is built according to the algorithm used in the abstraction tool YAGLE [8]. A pseudo-gate is similar to the CMOS *stage* introduced by Ousterhout [10] in the timing verifier CRYSTAL. The transistor orientation problem is also solved by YAGLE. A *dependency graph* is built where the nodes are the gate output signals and the oriented edges are the dependences between signals (existence of a gate input-output relationship).
2. **The causality graph.** From this first graph Tas builds the *causality graph* where a node is one of the two possible events associated to a signal (LOW-HIGH transition and HIGH-LOW transition) and an oriented edge exists between two nodes if the transition of the output node can be actually driven by the transition of the input node (Fig 2). The edges are valuated by the gate delays computed by Tas. The delay calculation uses the Tas Short Channel MOS model [2]. It is based on analytical equations that take into account the non-linear current versus voltage characteristics of the MOS transistor, the signal slopes of *input* signal and the transient short circuit current during gate commutation.
3. **The critical path graph.** Exhaustive analysis of all paths in the causality graph is performed in order to find the resulting worst possible timing behavior using a backward breadth-first algorithm. The *critical path graph* is defined where one node is one of the two possible events associated to a *stability* terminal signal and the edges are the resulting critical path between two *stability* terminals (Fig. 2). The edges are valuated by the critical path delays that are the sum of the corresponding gate delays.

The *Critical Path Graph* gives the longest and the shortest paths of the circuit and the *Causality Graph* enables

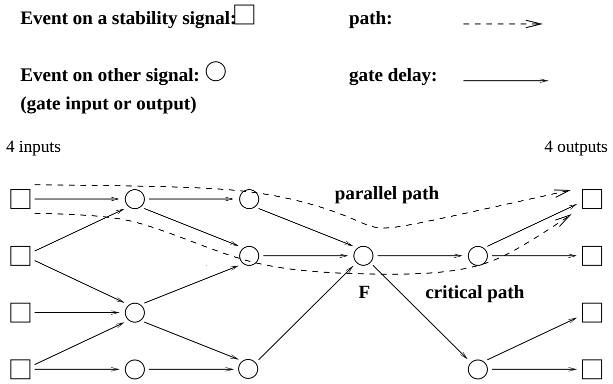

Figure 2. The Causality Graph (CG)

the analysis of parallel paths and provides the gate delay details of a given path. Both the causality graph and the critical path graph are necessary in the final stage of the timing analysis.

2.3. The Timing Analysis Complexity

The design of a multi-million transistor chip relies on decomposition into leaf-cell blocks of 10k-100k transistors followed by inter-block routing. Like others [9, 12, 17] we assume that flat, transistor level, static timing analysis is performed on each leaf-cell block. The analysis is then performed at higher levels following the design hierarchy.

The hierarchical netlists that are considered here are made up of instances and inter-block routing. Each instance being either hierarchical or a leaf cell.

The fundamental problem is to find the gate details of a path between two registers, when each of the registers belongs to two separate instances. Taking into account that this path may cross several other instances, the resulting problem solved here is to avoid building the complete causality graph or the complete critical path graph of the entire circuit.

Indeed, whilst the complexity of the causality graph grows linearly with the number of gates, the number of critical paths may grow quadratically (a register file containing n memory points writing to another register file containing n memory points can create up to $n.n$ paths). Thus the critical path graph requires significantly more memory while containing less information than the causality graph. Unfortunately the causality graph cannot be used for stability analysis since it would require as many critical path computations for the whole circuit as there are relaxation steps.

The dramatic increase of the number of critical paths for ULSI circuits is one of the main limitations of the static timing analysis.

2.4. Existing Approaches

In order to handle this complexity, several methods have been proposed to reduce the size of the critical path graph or to introduce a hierarchical representation of the delays. Campenhout and Mudge use *bus compression* [16], to replace a data vector (input or output of registers for example) by a single signal with the worst critical path. This approach is neither accurate nor automatic. Yalcin and Hayes [19] define a *hierarchical method* to build the timing representation of a circuit from the timing representations of its instances. Yet this method keeps all the paths (not only the critical ones) and thus is more dedicated to small digital arithmetic operators. Blaquière, Dagenais and Savaria use a *hierarchical multimode* approach to select paths through the circuit hierarchy [1]. The main improvement of this approach is to perform partial expansions of the critical path graph. Yet they assume that the critical path graph of a hierarchical circuit is much smaller than the union of the critical path graphs of its instances, which is generally untrue. Johannes, Claesen and De Man have improved the *false path search* exploiting the circuit hierarchy [6] but, to make large gains, the user should know rather well where relevant false path occur to define the convenient hierarchy of timing views. Kukimoto and Brayton [7] have proposed a hierarchical timing analysis technique for combinational circuits under state-of-the-art *sensitization* criteria. Their idea is to start from crude timing models based on topological delay analysis and to refine them gradually if additional accuracy is needed to get a better estimation of the circuit under analysis. Yet only *local false paths* completely contained inside a single leaf module are detected. Such a functional analysis could be used after HiTas to refine the delays required by the stability analysis.

Among commercially available tools, the closest approach is PathMill [9] from Epic-Synopsys which uses a black-box/grey-box timing view representation. The black-box representation is rapid, but contains little information (only set up time and hold time for terminals) and cannot support stability analysis which takes advantage of the latch transparency. The grey-box representation can handle latches but can generate huge amount of data proportional to the number of critical paths.

The HiTas tool offers a number of distinctive advantages compared to any of the existing tools. As far as designer interactivity is concerned, the HiTas tool allows to access to all critical paths (shortest and longest) with the gate details, and, in addition, access to details of all paths parallel to a given critical path. All this information can be obtained throughout the circuit hierarchy in a totally transparent way, emulating a simple, flat analysis.

3. HIERARCHICAL METHOD

3.1. The Multi-Level Hierarchical Timing View

The *Causality Graph*, made up of gate delays and the *Critical Path Graph*, made up of critical paths, are defined for each block in the hierarchical netlist (including the top cell representing the complete chip), as for the leaf cell blocks.

They are built recursively from the timing information contained in the timing view of each instance as well as from information coming from the interconnections between the instances.

To represent the timing view of a hierarchical level in a compact form, it is necessary to identify what is essential, i.e the new signals, the new or modified delays and the new or modified paths, versus the information contained in the timing view of the lower levels. A gate delay ending on a terminal at a level of the hierarchy is such that any path including that gate delay, in that level of the hierarchy, is characterized as an *unfinished* path. As a result, the computation of the delays including this gate in a higher level of the hierarchy will be computed again. On the other hand, critical paths between registers that do not have parallel paths including terminals in the current level of the hierarchy do not need to be computed again in a higher level.

We define the *level* of an instance in the hierarchical tree as its maximum distance from a leaf cell. The level of a leaf-cell is 1.

From the point of view of the representation on disk, each node in the hierarchical tree is described by a file representing the timing view, which includes references to the timing views of all sub-blocks, and the information relative to the causality graph that cannot be described in any of the timing views associated to each sub-block.

From the point of view of the representation in the computer memory, the *Causality Graph* and the *Critical Path Graph* are *flat*. But they can be incompletely represented : some instances can be *empty*, if the on-going analysis focuses on a path that is not related to this instance. In practical cases, the number of leaf cell blocks involved in the analysis of a given path is small. A software cache mechanism avoids to saturate the memory : an algorithm has been developed to free the memory that is unused by a particular path computation. The size of the software cache is a parameter of the hierarchical timing analyzer.

3.2. The Interconnections

At a given level of the hierarchy, the new nodes in the causality graph result from the interconnections. Figure 3 presents the two possible cases depending on the interconnect resistances.

- In the first case, the interconnect resistances are neglected, then : for each interconnection of instances in the father's view which is not ending on a father's terminal, a new signal is created (Fig. 3, signal ST).
- In the second case, the interconnect resistances can not be neglected, then : for each interconnection of instances in the father's view which is not ending on a father's terminal, a new signal is created for each terminal of the instances linked with this interconnection which belongs to the lower hierarchical level, (Fig. 3, signals T1 and T2).

This representation allows to use various models of delays resulting from interconnections [4, 11, 13] to get the sets of modified delays of new levels.

3.3. Critical Path Detection through the Hierarchy

When we want to compute the path between two registers (Rc and Rd in Fig. 4), the computation is made recursively as follows:

1. The *maximum* level of the hierarchy that has to be looked into is determined : it is the highest level of the hierarchy where both registers are referenced (Fig. 5, I31). A register is referenced in a view as soon as there exists a path ending on this register and containing a gate delay ending on a signal created in this view.

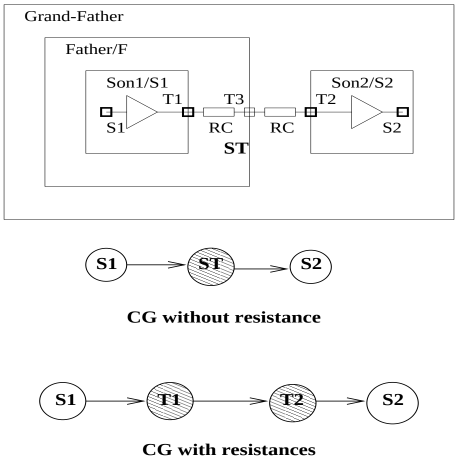

Figure 3. Multi-level causality graph with interconnections

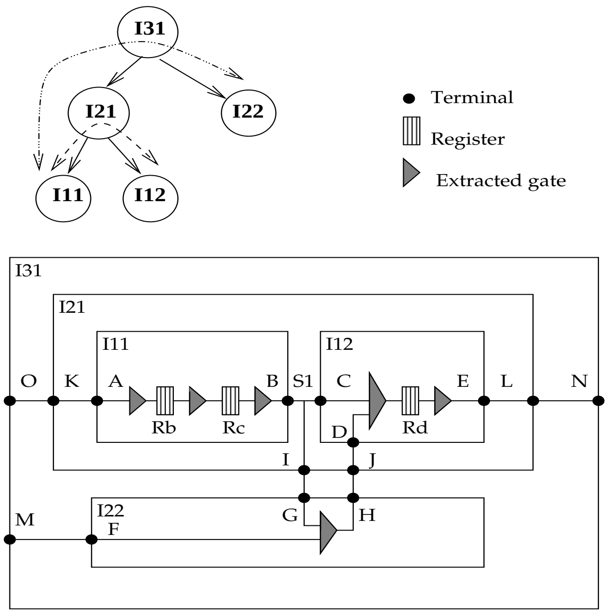

Figure 4. Problem of the hierarchical search for paths

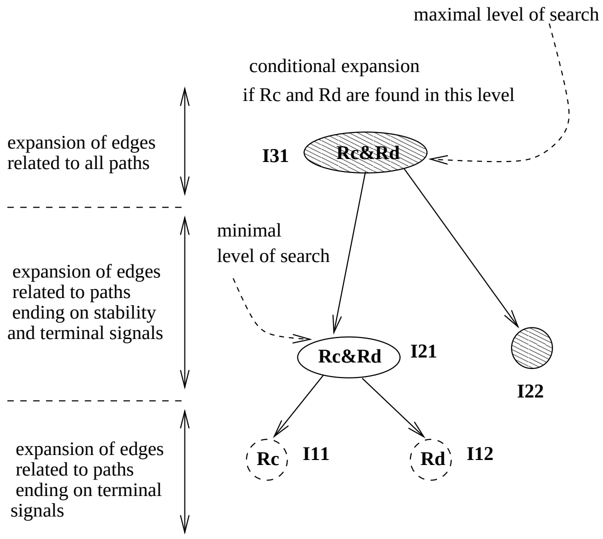

Figure 5. Partial expansion of the hierarchical timing views

2. The *minimum* level of the hierarchy that has to be looked into is determined: it is the lowest level of the hierarchy where both of the registers are referenced (Fig. 5, view I21).
3. Only few compact timing views are expansed to get the proper informations in the causality graph related to signals belonging to paths between those 2 registers. Thus a flat, but partial, causality graph is built.
4. The breadth first algorithm is used on this flat partial causality graph. When a node belonging to a view that has not been expansed yet appears, this view is also expansed recursively (Fig. 5, view I22).

4. CRITICAL PATH FACTORIZATION

In the previous section we presented the hierarchical approach to reduce the graph exploration. However, the size of the resulting path graph can still be a bottleneck. Using certain signals as stop points, to cut the paths into several segments, in order to share (or factorize) the common segments between paths can result in a significant reduction in the size of the critical path graph representation. As an example, take a register file (n memory points) writing to another register file (n memory points) through a bus (m bits, $m << n$). The possible number of paths is $n.n$. Using the bus signals as the stop points (*factorization points*) to define path segments decreases the number of paths to $2.n.m$. (Fig. 6).

The idea is to build a new graph : the *Reduced Causality Graph* where the nodes are the factorization points plus the stability nodes, and the edges are critical path *segments* (Fig. 6). To build this graph, a two phase algorithm has been developed, sparing the designer the trouble of selecting the factorization points. The complete algorithm is presented in [3].

Compared to the *Critical Path Graph*, the *Reduced Causality Graph* is always much smaller because the segments that belong to several critical paths are not dupli-

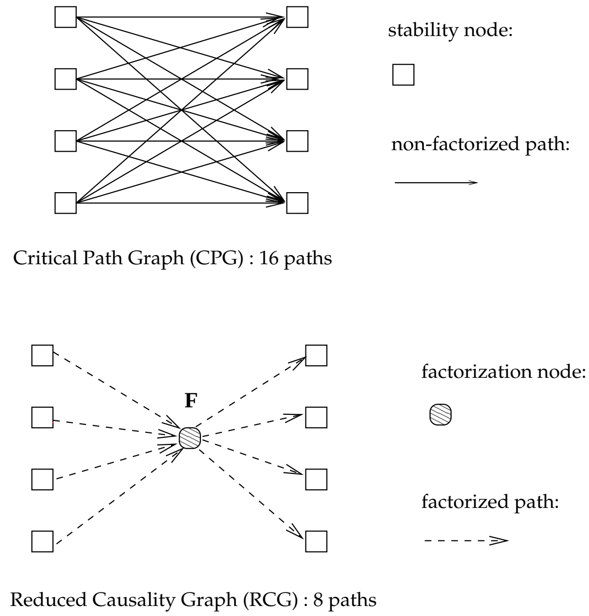

Figure 6. Critical Path Graph and Reduced Causality Graph related to the Causality Graph of Fig. 2

cated. The memory requirements of the reduced causality graph representation as well as the size of the resulting files required by the timing analysis are significantly reduced. Experimental results prove that the computation time for the critical path search is not at all increased by the use of these factorization points (Section 5.).

5. RESULTS

HiTas has been successfully used by BULL to verify a 7 million transistor chip. Here we present two smaller examples of industrial circuits, for which we have precise experimental results. Rcube [20] is a 8x8 reconfigurable router, fabricated by ST Microelectronics. It contains a cross-bar, several FIFOs and several controllers to implement the routing protocol, for a total complexity of 280 000 transistors. The Pciddc component [18] interfaces a net of Rcube and the PCI bus. It has been fabricated by Alcatel Mietec, and contains 200 000 transistors.

The flat construction of the causality graph is impossible for these two circuits. However the two circuits have been analyzed successfully thanks to our hierarchical approach.

Table 1 presents the various results obtained for the two circuits Rcube and Pciddc. Since the construction without the hierarchy is impossible, the graphs are built using the hierarchical approach but they are analyzed either in a flat mode or in a hierarchical mode to illustrate the advantages of the hierarchical analysis. In the flat mode, the complete causality graph is expanded in memory. The first analysis uses the flat approach. The second analysis uses the hierarchical approach and the factorization. The third analysis shows the effect of the software cache, in case of a limited memory workstation : a small degradation of the computing time, for a significant reduction of memory. The last line of Table 1 shows the time and memory necessary to analyze only the top cell of the hierarchy. Indeed an incremental hierarchical analysis allows to limit the path search to the highest hierarchical level since the blocks instantiated at this level have been verified and optimized in a previous step.

Table 1. Memory size (MegaBytes) and time (seconds) required to search the longest path with various types of analysis

type of analysis	Rcube memory	Pciddc memory	Rcube time	Pciddc time
flat with flattened graph	16,42	14,53	9,61	8,83
hierarchical with factorization	4,69	4,26	2,63	2,51
hierarchical with factorization and cache	4,28	2,43	2,73	2,91
only last hierarchical level	4,1	1,94	2,43	1,37

6. CONCLUSION

We have proposed a method for timing analysis of multimillion transistor chips. Based on the static timing analyzer Tas, the hierarchical analysis constructs a compact timing view of each block instantiated in the design hierarchy. This enables to build the *causality graph* and the *critical path graph*. Those graphs are flat, but incompletely instantiated. This allows the designer to make a true interactive analysis, including gate delay details and parallel paths across several blocks of the hierarchy. Yet the memory required for the timing analysis is sharply reduced versus the flat analysis case.

7. ACKNOWLEDGMENTS

The authors thank Anthony Lester for his work on the tool YAGLE and Michel Thill, Alain Debreil and Marcelo Duhalde from BULL for using HiTas intensively on high complexity circuits.

REFERENCES

[1] Yves Blaquière, Michel Dagenais, and Yvon Savaria. Timing Analysis Speed-up Using a Hierarchical and a Multimode Appproach. *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, 15(2):244–255, February 1996.

[2] K. Dioury, A. Greiner, and M-M. Rosset-Louërat. Accurate Static timing Analysis for Deep Submicronic CMOS Circuits. In *Proc. of the VLSI'97*, Gramado, Brasil, August 1997. Chapman & Hall.

[3] Karim Dioury, Alain Greiner, and Marie-Minerve Rosset-Louërat. An Efficient Algorithm for Data Reduction in Static Timing Analysis. In *Submitted to the 36th ACM/IEEE Design Automation Conference*, New Orleans, USA, June 1999.

[4] W.C Elmore. The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers. *Journal of Applied Physics*, 19(1):55–63, January 1948. 1.

[5] M. J. Flynn. What's Ahead in Computer Design ? In *Proc. of the 23rd EuroMicro Conference*, pages 4–9, Budapest, September 1997.

[6] P. Johannes, L. Claesen, and H. de Man. Performance through hierarchy in static timing verification. In J. van Leeuwen, editor, *Proc. 12th World Computer Congress*, pages 703–709, Madrid, Spain, September 1992. Elsevier Science Publishers B.V.

[7] Yuji Kukimoto and Robert K. Brayton. Hierarchical Functional Timing Analysis. In *Proc. of the 35th ACM/IEEE Design Automation Conference*, pages 580–585, San Francisco, USA, June 1998.

[8] Anthony Lester, Pirouz Bazargan-Sabet, and Alain Greiner. YAGLE, a Second Generation Functional Abstractor for CMOS VLSI Circuits . In *Proc. of the International Conference on Microelectronics, ICM'98*, Monastir, Tunisia, December 1998.

[9] Simon Napper. Static timing analysis – a demanding solution. *Electronic Engineering*, pages 35–42, January 1996.

[10] John K. Ousterhout. A Switch-Level Timing Verifier for Digital MOS VLSI. *IEEE Trans. on Computer-Aided Design*, CAD-4(3):336–349, July 1985.

[11] Lawrence T. Pillage and Ronald A. Rohrer. Asymptotic Waveform Evaluation for Timing Analysis. *IEEE Trans. on Computer-Aided Design*, 9(4):352–366, April 1990.

[12] Linda Pruitt. Verifying Mega-Gate IC Designs. *Electronic Engineering*, pages 67–68, May 1995.

[13] Jessica Qian, Satyamurthy Pullela, and Lawrence Pillage. Modeling the "Effective Capacitance" for the RC Interconnect of CMOS Gates. *IEEE Trans. on Computer-Aided Design*, 13(12):1526–1535, December 1994.

[14] A. L. Sangiovanni-Vincentelli, P. C. Mc Geer, and A. Saldanha. Verification of Electronic Systems. In *Proc. of the 33rd ACM/IEEE Design Automation Conference*, Las Vegas, USA, June 1996.

[15] Steven E. Schulz. Timing Analysis Tools and Trends. *Integrated System Design Magazine*, November 1995.

[16] David Van Campenhout and Trevor Mudge. Faster static timing analysis via bus compression. Technical Report CSE-TR-285-96, The University of Michigan, Dept. of Electrical Engineering and Computer Science Ann Arbor, Michigan, USA, 1998.

[17] S.V. Venkatesh, Robert Palermo, Mohammad Mortazavi, and Karem A. Sakallah. Timing Abstraction of Intellectual Property Blocks . In *Proc. of the IEEE 1997 Custom Integrated Circuits Conference*, pages 99–102, Santa Clara, USA, May 1997.

[18] F. Wajsburt, J-L. Desbarbieux, C. Spasevski, S. Penain, and A. Greiner. An Integrated PCI component for IEEE 1355 Networks. In *Proc. of the European, Multimedia Microprocessor Systems and Electronic Commerce Conference and Exhibition*, Florence, Italy, November 1997.

[19] Hakan Yalcin and John P. Hayes. Hierarchical Timing Analysis Using Conditional Delays. In *ICCAD-95 Digest of Technical Papers*, pages 371–377, San Jose, Californie, November 1995.

[20] Belkacem Zerrouk, Vincent Reibaldi, Frédéric Potter, Alain Greiner, and Anne Derieux. A gigabit serial links low latency adaptive router. In *In the Records of the IEEE Hot Interconnects IV*, Palo Alto CA, U.S.A, August 1996.

HIERARCHICAL STATIC TIMING ANALYSIS
FOR CMOS ULSI CIRCUITS

Karim Dioury , Alain Greiner and Marie-Minerve Louërat
LIP6, ASIM Department
University Pierre et Marie Curie (Paris 6)
Bldg 55-65, room 210,
4 place Jussieu,
75252 Paris Cedex 05, FRANCE
Phone: (33) 1-44 27 30 43,
e-mail:{Alain.Greiner,Karim.Dioury,Marie-Minerve.Louerat}@lip6.fr

With the advent of deep submicronic technologies, static timing analysis has revealed itself as the only feasible method for the timing verification of circuits, which can nowadays contain up to several million transistors. Nevertheless, the flat, transistor level, analysis exceeds the memory capacity of most workstations. To alleviate this problem, we have defined a method based on the hierarchical partitioning of the design phase. The propagation times within a circuit are represented using a multi-level hierarchical timing view that has allowed us to develop the hierarchical timing analysis tool HiTas as well as the interactive path browser XTas.