Hierarchical Static Timing Analysis at Bull with HiTas

K. Dioury, A. Lester
Avertec
44, rue Monge
75005 Paris
France
karim.dioury@avertec.com

A. Debreil
Bull Les-Clayes-sous-Bois
rue Jean Jaurès
78340 Les Clayes sous Bois
France
alain.debreil@bull.net

G. Avot, A. Greiner, MM. Louërat
CNRS-UPMC/LIP6/ASIM
4, place Jussieu
75252 Paris Cedex 05
France
marie-minerve.louerat@lip6.fr

Abstract

This paper describes the method used in the design of a 26 million transistors chip at BULL to verify the timing performance using the hierarchical timing analysis tool HiTas as well as the interactive path browser Xtas. Those tools have been designed at UPMC and are now commercialized by AVERTEC. The complexity is handled by partitioning the analysis according to the hierarchical partitioning of the design phase. The propagation times within a circuit are represented using a multi-level hierarchical timing view. [1]

1. Introduction

Timing analysis is an important step in the verification of digital ULSI circuits. With the advent of deep submicronic technologies, static timing analysis has revealed itself the only feasible method for the timing verification of circuits. Unlike timing simulators, static timing analyzers require no input patterns and find longest and shortest paths between critical signals. Nevertheless this method can generate an excessively large amount of data [8, 9, 10, 12].

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 timing graph, in which the nodes correspond to events on the signals and the edges correspond to the delays between two events on two signals.

The multi-level hierarchical representation was used to develop the hierarchical timing analyzer HiTas as well as the interactive path browser Xtas. HiTas and Xtas have been successfully used by BULL to verify a 26 million transistor processor.

The paper is organized as follows : Section 2 introduces the objectives. In section 3 the hierarchical method is presented, followed by the chip timing analysis in section 4. Finally, conclusions are summarized in section 5.

2. The transistor level analysis

2.1. Objectives

For proper sequential operation, a chip is required to meet setup time and hold time constraints. Complete sequential circuit verification must ensure that *all* propagation times between *stability terminals* (external inputs/outputs and internal registers) 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*.

The timing data required by the designers consist of all the critical paths between stability terminals, together with all the gate delays and interconnection delays of each path, to enable error correction and design optimization. Using traditional static timing analysis methods, such details can only be provided for circuits of less than around 100k transistors.

With the emergence of system on chip and the resulting multi-million transistor chips, the volume of

[1]This work was supported by the European project MEDEA

timing data can no longer be handled by a flat analysis [8, 9, 10, 12]. Dealing with this complexity has required abstraction. In addition, the closer electrical interaction of gates and interconnections create challenges in accurately modeling short channel C-MOS transistors. The design of a complex chip uses a structural decomposition, consisting of breaking up the design into units that allows the design process to be parallelized. The aim of the analysis performed here is to handle the timing verification of a multi-million transistor chip 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 blocks of the hierarchy, the results are subsequently used by the hierarchical analyzer (Fig. 1).

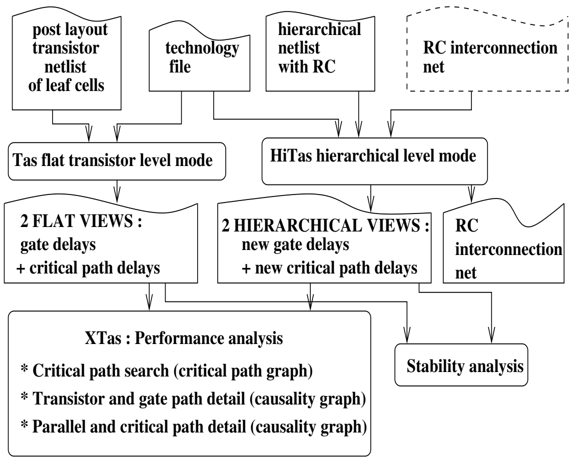

Figure 1. Flow graph of the hierarchical timing analysis

2.2. The timing analysis complexity

Two timing graphs are built. These form the basis of the timing or stability analysis. The *Critical Path Graph* gives the longest and the shortest paths of the circuit and the *Causality Graph* enables the analysis of parallel paths and provides the gate and interconnection 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].

The Causality Graph. From the extracted transistor netlist, a *pseudo-gate* netlist is built according to the algorithm used in the abstraction tool Yagle [7]. 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). From this graph HiTas 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 HiTas. The delay calculation uses the Tas Short Channel MOS model [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.

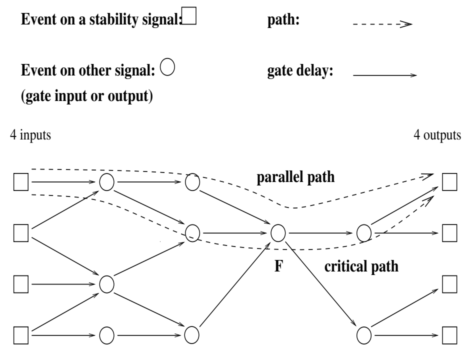

Figure 2. The timing graphs

The demands for performance implied the use of several custom blocks in the design. Thus our timing analysis starts from a flat transistor netlist. 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 (Fig. 1 and Fig. 3). .

The fundamental problem is to find the gate details of a path between two registers, where the registers belong to two separate blocks and where the path may cross several other blocks (see for example the path between I11.Rc and I12.Rd in Figure 3). 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 can increase with the square of the number of terminals. 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.

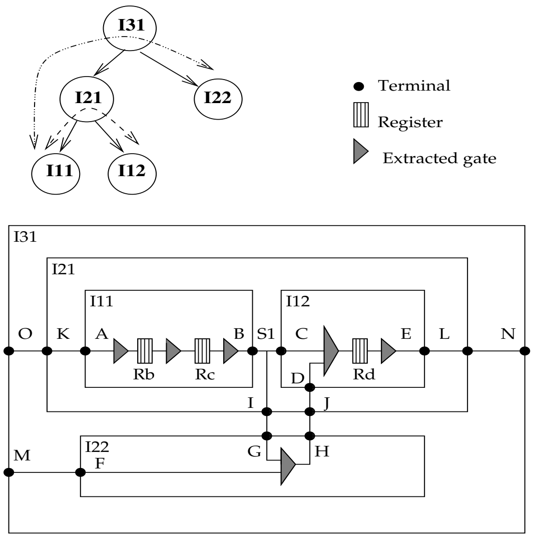

Figure 3. Problem of the hierarchical search for paths

3. Solution provided by HiTas tool

3.1. Multi-level hierarchical approach

Based on the static timing analyzer Tas [3], the hierarchical analysis builds a compact timing view of each block instantiated in the design hierarchy. 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 in the current level of the hierarchy do not need to be computed again in a higher level.

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), and 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.

The critical path detection through the hierarchy uses the hierarchy defined by the design phase. The path search is performed on a flat graph of the circuit, however this graph can be incomplete. Only the sub-graphs of the circuit blocks required for a given search are instantiated. Any unused circuit block is removed by a software cache if the maximum available memory is reached.

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.

3.2. 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 node as "factorization points", a path is cut into several segments, in order to share (or factorize) the common segments between paths. This factorization results in a significant reduction in the size of the critical path graph representation [5]. 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*. To build this graph, a two phase algorithm has been developed, sparing the designer the trouble of selecting the factorization points.

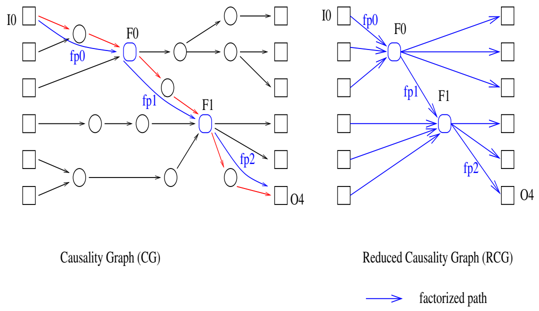

Figure 4. the Reduced Causality Graph (RCG)

Compared to the *Critical Path Graph*, the *Reduced Causality Graph* is several order of magnitude smaller because the segments that belong to several critical paths are not duplicated. 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 increased by the use of these factorization points [2].

To illustrate the ability of the tool to decrease the memory size required to represent the critical paths, the complexity of the three timing graphs the *Reduced Causality Graph*, the *Causality Graph*, and the *Critical Path Graph* has been computed on 7 test circuits.These circuits have been designed and manufactured, the last two are industrial circuits:

- Add32 : 32 bit, adder, 1500 transistors ;
- Mul32 : 32 bit multiplier, 2400 transistors ;
- Amd2901 : 4 bit data-path, 3800 transistors ;

- Hadamard : Hadamard transformation, 16000 transistors ;
- Dlxm : DLX, micro-programmed version, 38000 transistors;
- Rcube [15] : router Rcube, 280000 transistors;
- Pciddc [13] :interface between Rcube and the PCI bus, 200000 transistors.

Using these three graphs, the longest path search has been performed with HiTAS, the results are presented in Table 1 and Table 2.

circuit	RCG	CG	CPG	CG/RCG	CPG/RCG
Add32	998	2844	4670	2.85	4.68
Mul32	2222	47401	12736	21.33	5.73
Amd2901	1783	5588	26936	3.13	15.11
Hadamard	8486	21332	13358	2.51	1.57
Dlxm	13903	49446	1013425	3.56	72.89
Rcube	150796	388154	421401	2.57	2.79
Pciddc	115421	276676	355437	2.40	3.08

Table 1. Comparison of the complexity (number of nodes + number of edges) required by the three graphs : RCG (Reduced Causality Graph), CG (Causality Graph), CPG (Critical Path Graph)

circuit	RCG	CG	CPG	CG/RCG	CPG/RCG
Add32	0.02	0.08	0.12	4	6
Mul32	0.09	15.24	0.34	169.33	3.77
Amd2901	0.07	0.96	0.62	13.71	8.86
Hadamard	0.16	0.75	0.28	4.68	1.75
Dlxm	3.44	38.38	31.65	11.16	9.20
Rcube	3.81	12.22	9.61	3.21	2.43
Pciddc	3.11	12.66	8.83	4.07	2.73

Table 2. Comparison of the computation times (s) required for the longest path search with the three graphs : RCG (Reduced Causality Graph), CG (Causality Graph), CPG (Critical Path Graph)

Table 1 shows that the factorization points can reduce the number of paths stored and hence the memory by a factor up to 73 in the case of the DLX processor. Table 2 shows that the time required to perform the longest path search is decreased by using the *Reduced Causality Graph*, since the time spent in reading the timing files is reduced significantly more than the increase in the path search time.

3.3. HiTas versus other approaches

Other approaches have brought solutions to handle the complexity of static timing analysis. Yet they focused either on the recursive construction of the hierarchical *Critical Path Graph* [1, 4, 6, 14] or of the reduction of this graph [11] or of the reduction of the *Causality Graph* [5]. HiTas offers a number of distinctive advantages compared to existing tools since it handles both hierarchy and data reduction in *both* the *Causality Graph* and the *Critical Path Graph*, allowing stability analysis, interactive path browser and design optimization.

As far as designer interactivity is concerned, the HiTas tool allows 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.

4. Timing analysis of a 26 million transistor chip

HiTas is able to handle a large variety of technologies thanks to an external electrical parameter file. This parameter file is obtained by performing a series of electrical simulations using Eldo (from Mentor Graphics) on a set of reference circuits. These same circuits are analyzed using HiTas. The electrical parameters are tuned to make the HiTas results match those of Eldo.

To validate the accuracy of this parameterization phase, a set of benchmark circuits are analyzed using HiTas and Eldo. They consist of small circuits, extracted from the real design with XCalibre (from Mentor Graphics) and representative of the whole circuit : adder, rom, shifter and glue logic. For each of these benchmarks the calculated propagation delays are compared. In all the cases the results of HiTas are within 5% of Eldo.

An example is given in Figure 5 for a $0.25\mu m$ technology. The propagation delay from the falling signal A to the falling signal P through the rising signal NC is estimated with HiTas at 1157ps which compares well to the reference delay given by Eldo : 1148ps.

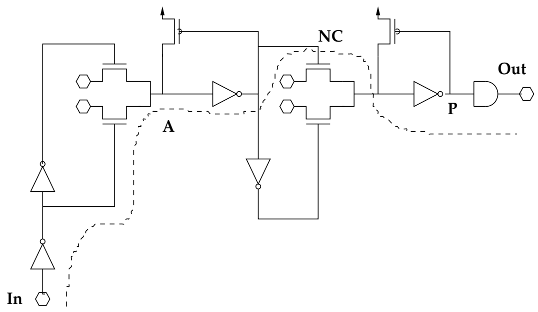

Figure 5. Schematic of an extracted path

HiTas has been used massively by BULL to perform the timing analysis of a commercial CPU chip.

The implementation was in a 0.18μ process with 6 metal layers. The chip consists of 5 units of 200k, 400k, 1.3M, 2.8M and 2.2M transistors, the last unit being instantiated 10 times. This totalizes 26 million transistors with 4 levels of hierarchy, the logic is spread into 300 blocks. The floor plan of the chip is presented in Figure 6.

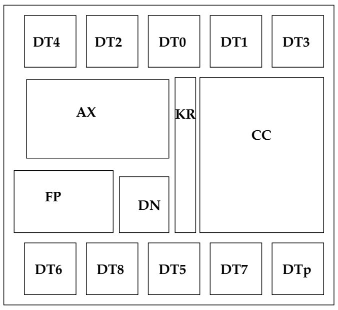

Figure 6. the 26 million transistor chip designed by Bull (18.4 mm2)

The interconnection resistances are taken into account from transistor level to chip level. The HiTas execution takes a few minutes for each block, this represents a total of 10h47 on a Sparc 10 station.

5. Conclusions and future work

The HiTas tool allowed BULL to successfully verify a 26 million transistor chip. The main advantages are the path factorization, which makes the size of the timing database linear versus the complexity of the circuit, the hierarchical approach, which makes the computation time linear and decreases the memory requirement and the computation time for the critical path search. It provides a gate-level description of any path as easily as with a flat approach.

LIP6 laboratory, BULL and AVERTEC are currently collaborating to introduce cross-talk management into HiTas.

References

[1] Y. Blaquière, M. Dagenais, and Y. 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. Louërat. Hierarchical Static Timing Analysis for CMOS ULSI circuits . In *1999 ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital System, (TAU 99)*, pages 65–70, Monterey, USA, March 1999.

[3] 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.

[4] 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.

[5] N. Kobayashi and S. Malik. Delay Abstraction in Combinational Logic Circuits. *IEEE Trans. Computer Aided Design of Integrated Circuits and Systems*, 16(10):1205–1212, October 1997.

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

[7] A. Lester, P. Bazargan-Sabet, and A. 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.

[8] L. Pillegi. Achieving Timing Closure for Giga-Scale IC Designs. In *Proc. of the Int. Worshop on Timing Issues in the Specification and Synthesis of Digital Systems*, pages 25–28, Monterey, CA, March 1999.

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

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

[11] D. Van Campenhout and T. 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, 1996.

[12] S. Venkatesh, R. Palermo, M. Mortazavi, and K. 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.

[13] 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.

[14] H. Yalcin and J. P. Hayes. Hierarchical Timing Analysis Using Conditional Delays. In *ICCAD-95 Digest of Technical Papers*, pages 371–377, San Jose, Californie, Nov. 1995.

[15] B. Zerrouk, V. Reibaldi, F. Potter, A. Greiner, and A. 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, Aug. 1996.