

## Timing Constraints for Correct Performance

Habib Youssef, Eugene Shragowitz

Computer Science Department  
University of Minnesota  
200 Union Street S.E., Minneapolis, MN 55455

### ABSTRACT

With the advances in VLSI design, chip timing is becoming dominated by interconnect delays rather than macro performances. This fact requires a change in the methodology of timing analysis, verification and physical design. In this paper, we describe efficient algorithm to derive timing constraints on all the interconnects, which are consistent with the correct timing performance. Description of this algorithm is accompanied with experimental results, which demonstrate the effect of these constraints on the final layout.

### 1. Introduction

In the past, the timing characteristics of designs were dominated by the timing characteristics of the switching circuits. A whole generation of verification tools was based on this assumption [1-7]. It was possible to ignore the interconnect delays because they constituted a negligible part of the resulting delay along any logical path. Increase in the switching speed and density of transistors on the chip created a new situation.

Placement related long path timing problems are caused by large propagation delays on the interconnects. These timing problems are very difficult to correct because they may require, not only new iterations of the physical design step, but possibly, many iterations of the logic design step. This fact imposes new requirements on the organization and substance of timing verification and physical design. It is clear that timing verification can not anymore be performed prior to physical design, and physical design itself should be governed by timing requirements.

One way to prevent the occurrence of placement related long path timing problems, and to make a circuit faster without making any changes in its logic design is to reduce the propagation time to a minimum. This goal can be achieved by imposing timing constraints on the interconnects, which would be used during the physical step. This situation has motivated us to develop a methodology and a set of tools, which will help solve timing problems in VLSI.

Other attempts have been reported to develop timing information that can be used during physical design of cell-based systems [9]-[14]. Our algorithm to compute timing constraints on the nets can be compared to the *Zero-Slack Algorithm* presented in [12]. Our algorithm is different in three principal aspects: (i) it has linear time complexity, while the

algorithm presented in [12] has a quadratic complexity (in the number of nets); (ii) in our methodology, distribution of the path slacks on the nets is proportional to the distribution of the loading input delays on the nets, while in [12] a uniform distribution was adopted; (iii) we use a more detailed timing model. The reported works also differ in the application of the obtained timing information to physical design.

### 2. Cell Based Timing Analysis

Cell based timing analysis involves two steps. In the first step, the *cell library* is processed and an abstract model of each cell is built. These abstract models are used to analyze all designs which use the macros of the library. For this system, the abstract model of each cell contains structural and timing information about the cell. The structural information consists of:

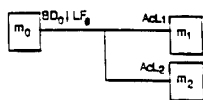
- Type of the macro, i.e. whether sequential or combinational.
- List of the input and output pins.
- The functional dependency between the input and output pins of the macro so that structural false paths are not enumerated.
- The Unateness of each output pin. The pin is positive unate if the signal is not inverted. It is negative unate if the signal is inverted. Otherwise the pin is nonunate.

As for the timing information, the following is recorded:

- The AcLoad  $AcL_i$  at each input pin "i", which is the external load capacitance on the pin.
- The Base Delay  $BD_o$  and load factor  $LF_o$  at each output pin "o". The Base Delay represents the internal delay of the cell. It is the part of the delay that is independent of the composition environment of the cell. The Load Factor is the amount of delay per unit load capacitance as specified by the AcLoad on each of the loading input pins. This parameter is used to compute the part of the switching delay that is dependent on the composition environment of the cell. Figure 1 illustrates how these parameters are used to compute the cell delay.
- The Setup and Hold time requirements in case the macro is a sequential circuit (Latch or Flip-Flop).

For each macro in the library, the above timing parameters are derived from extensive SPICE simulations.

To account for the inaccuracy of the model, upper and lower bounds are provided instead of single load factor and



$$\text{Delay}(m_0) = BD_0 + LF_0 * (AcL_1 + AcL_2)$$

Fig. 1. Computation of macro switching delays.

base delay values. Furthermore, to make the model less pessimistic, it is very important to differentiate delays between rising and falling signals. In this system, distinct values are provided for each signal type, and unateness information about each output pin of each macro is used during the path tracing, so that the proper values of the delay and load factor are selected. Also, functional dependencies between the input and output pins of each cell are used to eliminate all structural false paths in the design.

### 3. Derivation of Timing Constraints for Layout

It is common knowledge that timing requirements in synchronous designs are path oriented. On the other hand, physical layout tools are net oriented. Since the interconnect part of the delay is unknown prior to layout, our objective is to use the static timing analysis step to find the path timing constraints and transforms them into constraints on nets.

Static timing analysis is based on path tracing. A net-list description of the circuit is parsed and transformed into a directed graph. Once the circuit graph is set, the data and control paths are traced. A design is free from timing problems if every short path in the design is greater than its lower bound (the earliest required arrival time of the signal at the path sink, *ERAT*), and every long path is less than its upper bound (the latest required arrival time of the signal at the path sink, *LRAT*). Therefore, long and short paths are characterized by the following two inequalities:

There is no long-path problem on path  $\pi_i$  iff:

$$T_{\pi_i}^{\max} \leq LRAT_{\pi_i} \quad (1)$$

With Flip-Flops, there is no short-path problem on path  $\pi_i$  iff:

$$T_{\pi_i}^{\min} \geq ERAT_{\pi_i} \quad (2)$$

$T_{\pi_i}^{\min}$  and  $T_{\pi_i}^{\max}$  are the minimum and maximum delays along path  $\pi_i$ .

#### 3.1. Minimax Approach

For real VLSI designs, long paths are most difficult to avoid. The speed of the circuit is determined by the propagation delay along its longest path. Short path problems are easier to avoid and are less likely to happen in VLSI designs. Usually, they are prevented by the insertion of extra circuit components (I/O buffers) to lengthen all suspected short paths. For all the test cases we experimented with, the lower bounds of the short path inequalities, i.e. the  $L_{\pi_i}$ , were negative, which indicates that any layout of these designs will be free from short path timing problems. Hence, the timing constraints are reduced to the long path inequalities. For linear systems of that form, we can adopt a different solution approach. Instead of looking for the interconnect delays which optimize a certain linear objective function, we look

for upper bounds on these delays which satisfy all long path inequalities of (1). We call this approach **Minimax**.

Let  $\eta$  be a vertex in the circuit graph. Let  $d_{\eta}^{\max}$  be the maximum switching delay of the driving cell of the net modeled by this vertex, and  $x_{\eta}^{\max}$  be the maximum interconnect delay along the net. These quantities are expressed as follows,  $d_{\eta}^{\max} = BD_{\eta}^{\max} + LF_{\eta}^{\max} \times AcL_{\eta}$  and  $x_{\eta}^{\max} = LF_{\eta}^{\max} \times c_{\eta}$ , where  $c_{\eta}$  is the interconnect capacitance of net  $\eta$ . Based on this delay model, the interconnect delay along a given net is a function of its load factor as well as its interconnect capacitance. Note that the load factor is a layout independent timing parameter as opposed to the interconnect capacitance which is layout dependent. Note also, that though nets may have the same interconnection lengths, their contribution to the propagation delays may be significantly different because they have different drivers with significantly different load factors. Circuits are designed with the hope that only nets with small load factors (strong drivers) have large loads. These observations lead to the *Minimax* approach.

The key idea of the *Minimax* approach is based on the selection of a delay bound for each net, consistent with the timing constraints on all the paths which contain the net. Path slacks are divided between nets proportionally to their weights. The *net weight* is chosen equal to the load delay on the net. Using the already introduced notation, the weight of a net  $\eta$  is,

$$\rho_{\eta} = LF_{\eta} \times AcL_{\eta} \quad (3)$$

Hence, the distribution of the interconnect delays along the nets of a path, is approximated by the distribution of the weights of the nets of the path.

Let  $\pi = [v_1, \dots, v_i, \dots, v_n]$  be a path in the circuit graph  $G$ . The weight of path  $\pi$  is equal to the sum of the weights of its nets, i.e.,

$$P_{\pi} = \sum_{\eta \in \pi} \rho_{\eta} \quad (4)$$

Each net  $v_i$  is assigned a timing bound computed as follows,

$$x_{v_i}^{\pi} = \rho_{v_i} \times \frac{U_{\pi}}{P_{\pi}} \quad (5)$$

Let  $\Pi_{\eta}$  be the set of the paths which traverse net  $\eta$ . In order for the timing bound computed for a net to be consistent with all the bounds for paths traversing the net, the minimum one is selected. Let  $u_{\eta}$  be that minimum delay value.  $u_{\eta}$  is defined as follows,

$$u_{\eta}^* = \min_{\pi \in \Pi_{\eta}} x_{\eta}^{\pi} \quad (6)$$

#### Definition 1

The score,  $\Theta_{\pi}$  of path  $\pi$  is defined as the ratio of its slack  $U_{\pi}$  and its weight, i.e.,

$$\Theta_{\pi} = \frac{U_{\pi}}{P_{\pi}} \quad (7)$$

#### Definition 2

A path  $\pi_v$  is said to cover a vertex  $v$  in the circuit graph  $G$  if and only if it has minimum score  $\Theta_{\pi_v}$  among all paths

which traverse  $v$ .

Let for each vertex  $\eta$  in the graph  $G$  of a given circuit,  $\pi_\eta$  be the path which covers  $\eta$  as stated in Definition 2, and  $\Theta_{\pi_\eta}$  be its score. Because  $\pi_\eta$  has minimum score amongst all paths traversing the net  $\eta$ ,

$$u_\eta^* = \rho_\eta \times \Theta_{\pi_\eta} = \rho_\eta \times \frac{U_{\pi_\eta}}{P_{\pi_\eta}} = u_\eta^* .$$

### Theorem <sup>1</sup>

If the interconnect delay  $x_\eta$  of each net  $\eta \in V$  in the circuit graph  $G$  is such that,  $x_\eta \leq u_\eta^*$ , then the circuit is guaranteed to be free from long path timing problems.

The *Minimax* approach consists in finding, for all the nets of the circuit, bounds on their propagation delays as defined by equation (5). The above theorem guarantees that, if the interconnect delay  $x_\eta$  on each net is no larger than its associated bound  $u_\eta^*$ , then the design after layout will be free from long path timing problems.

### 3.2. Iterative-Minimax

Initially, all nets are assumed to have a zero delay. After the first assignment of delay bounds by *Minimax*, the slacks of all the paths are updated and *Minimax* is called again to compute delay increments for all nets which remain covered by paths with positive slacks. These delay increments are added to the already assigned net delay bounds. The path slacks are updated, and the process is repeated until all paths have a zero slack. Convergence of this procedure has been proven (proofs are omitted due to lack of space).

In practice, the number of iterations required is always very small (less than 5). Usually, after the second or third iteration, more than 90% of the paths have zero slack. It is important to mention that paths with critical timing requirements (small slacks and many nets with large loads) will have a zero slack in early iterations (first or second). It is the non critical paths (at most 10% of the paths) which require several executions of the *Minimax* procedure in order to distribute their slacks on their constituent nets. Therefore, in practical cases, no significant gain is obtained by running *Minimax* procedure for more than 4 or 5 times. Hence, *Iterative-Minimax* and *Minimax* can be assumed to have the same time complexity. Experimental results about real VLSI circuits will be provided at the end of this paper. In the following section, we focus on the *Minimax* algorithm.

### 3.3. Minimax Algorithm

The computation of the upper bounds for each net requires finding a path cover for the circuit graph. Recall that a path is said to cover a vertex  $v$  if it has minimum score among all paths which contain  $v$ . The score of a path  $\pi$  is equal to  $\Theta_\pi = \frac{U_\pi}{P_\pi}$ . We proved that the problem of finding the path with minimum score  $\Theta$  is NP-Hard. The proof consists in transforming our problem into an instance of the *PARTITION* problem which is known to be NP-

<sup>1</sup>Proofs of all theorems and lemmas are omitted due to lack of space.

*Complete*. Therefore, finding the path cover in the circuit graph as characterized by Definitions 2, requires the enumeration of all the paths in the graph, which is exponential in the size of the graph.

Next, we shall present an efficient algorithm, which finds an approximate to the minimax solution. We call this algorithm *Minimax-PERT* because it is based on a tracing similar to the *PERT* technique [15]. The idea is the following: instead of looking for a minimum path cover according to "Definition 2", we compute for each net  $\eta$ , the slack  $U_\eta^{\min}$  of the path with minimum slack among all paths which traverse the net, and the weight  $P_\eta^{\max}$  of the path with maximum weight among all paths traversing the net. Then, each net  $\eta$  is assigned a timing bound  $u_\eta = \rho_\eta \times \frac{U_\eta^{\min}}{P_\eta^{\max}}$ , where  $\rho_\eta$  is the weight of vertex  $\eta$  and is defined in equation (3).

### 3.4. Minimax-PERT

The algorithm performs a backward trace followed by a forward trace to find for each net-vertex  $\eta$ ,  $U_\eta^{\min}$  and  $P_\eta^{\max}$ . Then, each net  $\eta$  is assigned a timing bound  $u_\eta = \rho_\eta \times \frac{U_\eta^{\min}}{P_\eta^{\max}}$ . A formal description of *Minimax-PERT* is given in Figure 3.

#### Lemma

The timing bounds found by *Minimax-PERT* are less or equal to the timing bounds found by the *Minimax* algorithm.

Therefore, the timing bounds computed by *Minimax-PERT* leave the circuit free from long path problems.

---

```

Algorithm Minimax-PERT
It is assumed that the circuit graph has already been leveled.
Let  $V_i$  be the vertices in level  $i$ . The sources are at level  $L$ 
and the sinks at level 1.
FOR  $i=1$  TO  $L$  DO
  FOREACH  $v \in V_i$ 
     $D_v^- = \max_{\eta \in \Gamma_v^-} \{d_\eta + D_\eta^-\}$ ;
     $P_v^- = \max_{\eta \in \Gamma_v^-} \{\rho_\eta + P_\eta^-\}$ ;
  END FOREACH
END FOR;
(* Compute the minimum available slack at each sink *)
FOREACH  $s \in V_L$   $U_s^{\min} = LRAT_s - D_s^-$ ;
FOR  $i=L-1$  TO 2 DO
  FOREACH  $v \in V_i$ 
     $U_v^{\min} = \min_{\eta \in \Gamma_v} U_\eta^{\min}$ ;
     $P_v^{\max} = \max_{\eta \in \Gamma_v} P_\eta^{\max}$ ;
  END FOREACH
END FOR;
FOREACH  $v \in V_N$   $u_v = \rho_v \frac{U_v^{\min}}{P_v^{\max}}$ ;
END Minimax-PERT

```

---

Fig. 2. Minimax-PERT Algorithm

### 3.5. Time Complexity and Iterative Application of Minimax-PERT

Let  $|V_N| = N$  be the number of *net-vertices* in the graph  $G$ , and  $m$  be the maximum number of incident edges to any vertex in  $G$ . For each vertex  $\eta \in V_N$ , finding  $U_\eta^{\min}$  and  $P_\eta^{\max}$  takes  $O(m)$ . Since there are  $N$  vertices, the com-

plexity of *Minimax-PERT* is  $O(mN)$ . But because actual circuits have bounded fanouts, the parameter  $m$  is always bounded by a small constant. Therefore, for actual circuits, *Minimax-PERT* will usually exhibit a linear running time of  $O(N)$ .

One should note that the convergence proofs stated for *Iterative-Minimax* hold also for *Iterative-Minimax-PERT*, i.e., the repetitive application of *Minimax-PERT* eventually distributes all the available slacks on all the nets.

#### 4. Experimental Results

The algorithms presented in this paper were implemented and included in a timing analysis tool for cell-based design. The implementation was tested on four real designs provided by Control Data Corporation (CDC). The four test cases are implemented in the sea-of-gates technology. Table 1 describes the four chips, where  $N_{io}$ ,  $N_{cells}$ , and  $N_{nets}$  indicate respectively, the number of I/O pads, the number of cells, and the number of nets on the chip.

Chip	Description	$N_{io}$	$N_{cells}$	$N_{nets}$	Clock period
AA	32-bit multiplier	202	2268	3505	100 ns
AB	micro-code sequencer	231	3230	4752	100 ns
AD	main memory controller	210	4144	5470	100 ns
ETA	portion of a controller	232	5093	6367	21 ns

Table 1. The four test cases.

Input to the timing analysis tool is in the EDIF language, version 2 0 0 [8]. There are three steps in the analysis. The first step consists in the processing of the cell library. This step is executed only once per library and is used to extract the timing and physical properties of each macro in the library as described in Section 3. The library provided by CDC has more than a hundred macros and took less than five CPU minutes to process. The second step transforms a netlist description of the chip into an acyclic directed graph as described in Section 3. For each of the four CDC test cases, this step took less than 30 minutes. The third step checks the design for logic related timing problems and computes timing bounds on all the interconnects by *Minimax* algorithm.

All programs for timing verification, analysis, and derivation of the interconnect delay constraints were written in Common LISP and run on APOLLO workstations. The system has about 7,000 lines of code. *Iterative Minimax-PERT* was applied to *ETA* chip. Initially, all paths have positive slacks, and the average path slack is equal to 14.149 ns. After one iteration, the percentage of paths with zero slack is equal to 72.47%, and the average slack of the remaining paths that still have positive slacks is equal to only 2.896 ns. After 4 iterations 93.7% of the paths had zero slack, and the remaining 6.3% of the paths have an average slack equal to 1.179 ns.

Details of application of the timing information computed by this timing tool to physical design is out of the scope of this paper. Here, we briefly describe results of application of this timing information to influence the placement of cells, obtained by an in-house physical design system for the sea-of-gates design style.

A chip was placed and routed with and without the timing information. It is the *AA chip* in Table 1. For this chip, we also obtained a manifest improvement of the performance of the chip. There was also a decrease in the average connection length by about 10%. The layout of the chip without the influence of the timing constraints contained 21 long paths. No long paths were registered when timing constraints were applied.

#### 5. Conclusion

In this paper a new methodology and new algorithms for the derivation of timing constraints on all the interconnects were developed and applied to solving layout related timing problems. This new methodology is based on detailed information on timing characteristics of cells and nets. A minimax approach to identifying maximal delay bounds for nets, which do not violate the timing constraints on any of the logical paths in the design was proposed. A new approximation algorithm with proven polynomial time behavior was described. The recursive application of this algorithm results in the distribution of the whole remaining path slacks between the comprising nets, and as a result, zero slack is achieved. The obtained timing bounds were applied to produce layouts free from timing problems.

#### 6. References

- [1] Ryotaro Kamikawai et al., "A critical path delay check system", In Proc. of the 18th Design Automation Conference, pp. 118-123, 1981
- [2] Lionel C. Bening, Thomas A. Lane, and James E. Smith, "Developments in logic network path delay analysis", in Proc. of the 19th Design Automation Conference, pp. 605-609, 1982
- [3] Tohru Sasaki et al., "Hierarchical design verification for large digital systems", in Proc. of the 18th Design Automation Conference, pp. 105-112, 1981
- [4] Robert B. Hitchcock, Sr., Gordon L. Smith, and David D. Cheng, "Timing Analysis of Computer Hardware", in IBM J. RES. DEVELOP., Vol. 26, No. 1, pp. 100-116, 1982
- [5] E. Tamura et al., "Path Delay Analysis for Hierarchical Building Block Layout System", in Proc. of the 20th Design Automation Conference, pp.403-410, 1983
- [6] R. Reddi and C. Chen, "Hierarchical Timing Verification System", in CAD, vol. 18, no. 9, pp. 467-471, November 1986
- [7] Siegfried Heinkle, "A fast timing analysis tool for VLSI", in Proc., VLSI in COMPUTERS, pp. 193-197, 1987
- [8] Electronic Design Interchange Format, Version 2 0 0, Electronic Industries Association, Washington, D. C., May, 1987.
- [9] A. E. Dunlop et al., "Chip layout optimization using critical path weighting", in Proc. of the 21st Design Automation Conference, pp. 133-136, 1984
- [10] Michael Burstein and Mary N. Youssef, "Timing influenced layout design", in Proc. of the 22nd Design Automation Conference, pp. 124-130, 1985
- [11] Steven Teig, Randall L. Smith, and John Seaton, "Timing Driven Layout of Cell-Based ICs", in Design Automation Guide, pp. 94-101, 1987
- [12] Peter S. Hauge, Ravi Nair, and Ellen J. Yoffa, "Circuit placement for predictable performance", in Proc. of ICCAD'87, pp. 88-91, 1987
- [13] Ravi Nair et al., "Generation of Performance constraints for layout", in IEEE Transaction on CAD, Vol. CAD-8, No. 8, August 1989
- [14] Michael A. B. Jackson and Ernest S. Kuh, "Performance-driven Placement of Cell Based IC's", in Proc. of the 26th Design Automation Conference, pp. 370-375, 1989
- [15] T. I. Kirkpatrick, N. R. Clark, "PERT as an aid to logic design", in IBM J. RES. DEVELOP., Vol. 10, No. 2, pp. 135-141, 1966
- [16] Roy E. Marsten, "The Design of the XMP Linear Programming Library", in Transactions on Mathematical Software, vol. 7, no. 4, December, 1981.