# WTA – Waveform-Based Timing Analysis for Deep Submicron Circuits

Larry McMurchie and Carl Sechen
**Department of Electrical Engineering**
**University of Washington**
**Seattle, WA**
larry@ee.washington.edu, sechen@ee.washington.edu

## ABSTRACT
Existing static timing analyzers make several assumptions about circuits, implicitly trading off accuracy for speed. In this paper we examine the validity of these assumptions, notably the slope approximation to waveforms, single-input transitions, and the choice of a propagating signal based on a single voltage-time point. We provide data on static CMOS gates that show delays obtained in this way can be optimistic by more than 30%. We propose a new approach, Waveform-based Timing Analysis that employs a state-of-the-art circuit simulator as the underlying delay modeler. We show that such an approach can achieve more accurate delays than slope-based timing analyzers at a computation cost that still allows iterations between design modification and delay analysis.

## 1. Introduction
Developed during the mid-1980's, timing analyzers such as Crystal[1] and TV[2] were written to quickly produce estimates of critical path delays. Designers wanted such quick feedback in order to shorten the cycle time of design modification, delay measurement, design modification, *etc*. Previously, designers relied on circuit-level simulation that was too slow for several reasons. One was the problem of determining input vectors that exercised the critical paths. The other was the slowness of the circuit simulator itself. Hence timing analyzers were developed, achieving a speedup by using variants of the PERT algorithm to find critical paths, as well as a simplified circuit model for delay calculation.

Because such static timing analyzers use a simplified delay model and make approximations about the functioning of a circuit, the designer of a high-performance ASIC usually doesn't completely trust the results. Normally the designer will take some number of the most critical paths (ranging from 100's to 1000's) found by the timing analyzer and simulate them with a circuit simulator to determine more accurate delay estimations [7]. Often a path that that was not near the top of the list of critical paths becomes the most critical after circuit simulation.

The nature of full-custom design has also changed since the first static timing analyzers were written. With the ever-continuing pressure to increase clock frequency, the amount of computation that is performed during a single clock cycle has been reduced. Commensurately, the number of logic levels between latches has been reduced. Designs are also more highly structured than in the 1980's. Timing analyzer development during the mid-1980's was largely directed towards finding critical paths through large numbers of channel-connected devices. Today, the timing requirements of designs effectively prevent the use of such circuits. The emphasis, instead, is to construct highly-optimized netlists of fast gates.

The continuing adoption of dynamic circuit techniques is an additional change occurring during the last decade. Some timing analyzers (*e.g*. Pathmill from Synopsys) have been modified to accommodate domino circuits. There has been work to address other types of dynamic circuit techniques, *e.g*. self-resetting domino [5]. Such work has typically only dealt with setup/hold times and not employed circuit or device-level descriptions of the operation of the circuit. Typically, designers exhaustively simulate small pieces of the design and employ verification tools to ensure that setup/hold times are met when the small pieces are assembled.

Traditional approaches to static timing analysis have employed several delay model approximations:

**The slope approximation**. Worst-case falling and rising waveforms are approximated as a slope. Delay is calculated at a single time-voltage point, usually when the waveform crosses 50% of $V_{DD}$. Such an approximation neglects changes in slope at the beginning and at the end of the real waveform calculated by a circuit simulator. Crystal [1] and TV [2] employed this method and it is still used by commercial tools such as Pathmill from Synopsys.

**Single versus multiple-input transitions**. Typically, static timing analyzers propagate the latest arriving input – the LPA (Latest Propagation Algorithm). This approximation neglects the effects of other inputs on the same channel-connected path.

0-7803-7607-2/02/$17.00 

There have been several efforts to describe waveforms in more detail. One of the earliest efforts was the ELogic technique – a relaxation-based, multi-level, multi-strength approach to modeling circuits [3]. Although better at describing waveforms than the slope model, ELogic was not extended to multiple-input transitions.

Waveform approximation was addressed in [4], as well as the multiple-input transition issue. Using a circuit simulation technique similar to SPICE and a dynamic path selection technique, the worst-case rising/falling waveforms for each gate are determined. At each step in a simulation the least conductive path is selected based upon the currents through devices during the previous timestep. This process leads to a worst-case waveform that is at least as pessimistic as the waveform for any single path. Not handled were the effects of complimentary driving paths and side paths.

In [8] the effect of slope upon path delay was analyzed as well as multiple-input transitions. An algorithm was developed using a piecewise linear waveform that is shown to obtain path delays within 1% of Hspice. Although effects of multiple-input transitions are analyzed, they were not included in the algorithm.

A data dependent delay model for determining bounds on multiple-input transitions was developed in [9]. In combination with a modified topological sort, this model obtains delays that are more accurate than those obtained with the single-input transition model, but similar in computational effort. The slope approximation to waveforms is not addressed in this work.

In [6] the effects of using a single time-voltage point (usually $V_{DD}/2$) to determine a worst-case rising/falling signal was examined. Significant improvements in accuracy were found by propagating multiple slope-approximated waveforms.

Our new approach to static timing analysis, called Waveform-based Timing Analysis (WTA), addresses the drawbacks (inaccuracies) of current static timing analyzers and has the following properties:

- It is based upon an accurate device-level model that analyzes real waveforms, not slope-based approximations to waveforms.
- It provides an upper bound to delay (within the accuracy of the underlying device-level model).
- It uses a device-level model that is the same as the model used for circuit-level simulation of a design.
- It provides an acceptable design modification, timing analysis, design modification, *etc.*, cycle.
- WTA can accommodate dynamic circuit design styles, which involve different constraints than static CMOS circuits, although this is beyond the scope of this paper.

In the following section we describe the delay model employed by WTA. Section 3 describes WTA's breadth-first method of waveform propagation through circuits. Section 4 describes results on a set of ISCAS benchmarks, and Section 5 gives our conclusions.

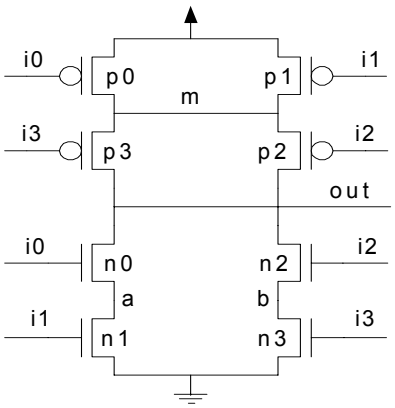



Figure 1. Labeling of AOI22

## 2. Delay Modeling

Conventional static timing analyzers model the worst-case delay of a gate by considering only single-input transitions. Consider the AOI22 gate shown in Figure 1. Without loss of generality, we focus our attention on modeling the delay for a falling output with respect to a single-input transition on input *i1*. Figure 2 shows the worst-case configuration imposed by this single-input transition on *i1*. *i1* is therefore the controlling input for the path to the output through *n0* and *n1*. (The arrows indicate the flow of positive charge.) The discharging path in the pulldown network is through *n0* and *n1*, where input *i0* is set to $V_{DD}$. A charge-sharing path exists between node *b* and *out* through *n2*. In the pullup network, the partially conducting path through *p1* and *p3* resists the discharge of *out* at the beginning of the gate's transition.

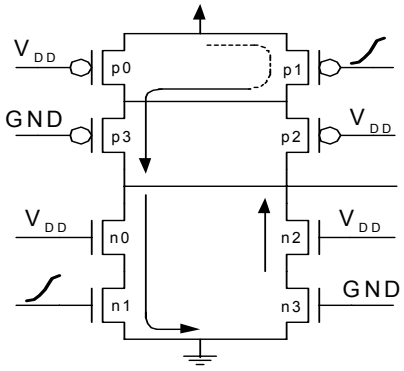
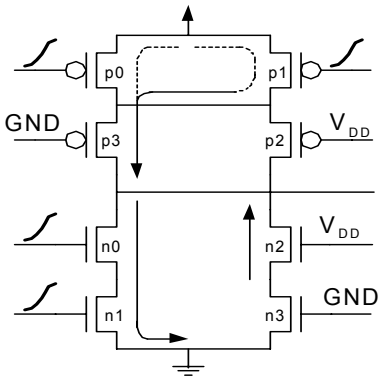
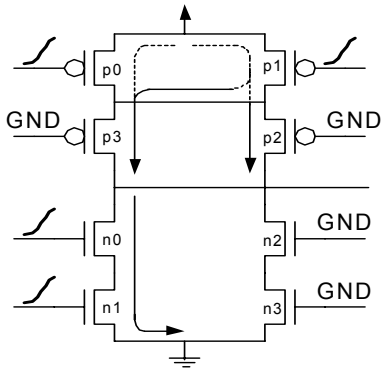


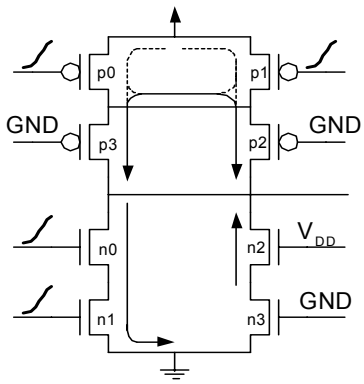

Figure 2. Single-input transition on *i1*




(a)




(b)




(c)

Figure 3. Worst-case configurations for multiple-input transitions.

Multiple-input transitions can lead to considerably longer output transition times. Figure 3a shows a delay model for multiple-input transitions on controlling inputs *i0* and *i1*. Relative to the single-input transition on *i1*, the current drawing capability of this path is reduced due to the fact that *i0* is only partly turned on. The degree to which the delay is increased depends upon the shape and offset of the waveforms for *i0* and *i1*. Just as with the single-input transition, there is a charge sharing path between node *b* and *out* through *n2*. In the pullup network a partially conducting path through both *p0* and *p1* resists the discharge of *out* at the beginning of the gate's transition. Relative to a single-input transition on *i1*, the ability of the pullup network to resist the discharge of *out* is increased, thereby increasing the delay.

Although one might expect that Figure 3a is the worst-case scenario, Figure 3b shows another possibility. Here, *i2* is set to *GND*, eliminating the charge-sharing path through *n2*, but allowing the pullup network to better resist the discharge of the output by turning on an additional path to *out* through *p2*. Depending upon the waveform of *i2* and its arrival time relative to *i0* and *i1*, either Figure 3a or Figure 3b or some combination of both will yield the worst-case delay. Since resistance to discharge dominates near the beginning of the transition and charge sharing dominates towards the end of the transition, a steeply rising waveform for *i2* that is near the middle of the gate's transition will tend to give the latest overall waveform for *out*. In order to obtain a worst-case bound on the waveform for the output of the gate, we adopt the scenario shown in Figure 3c, where *n2*'s gate is set to $V_{DD}$ and *p2*'s gate is set to *GND*. This incorporates the worst-case features of both Figures 3a and 3b. In other words, consistency of gate input values (*e.g. n2*/*p2* in this case) is relaxed to independently maximize the amount of charge that must be discharged to the output node from the pull-up and pull-down networks.

```
For each nMOS device
   If (controlling) set device gate to
      worst-case rising waveform
   Else if (device source node == GND)
      set device gate to GND
   Else set device gate to VDD
End for
For each pMOS device
   If (controlling) set device gate
      to worst-case rising waveform
   Else set device gate to GND
End for
```

Figure 4. WTA Algorithm for constructing a worst-case falling configuration with multiple inputs transitioning.

A general method for constructing a set of possible worst-case configurations for any static gate requires first enumerating all paths to *GND* for the falling output case, or all paths to $V_{DD}$ for the rising output case. In the case of a falling output, WTA constructs a worst-case configuration for each path to *GND* as shown in Figure 4. (Worst-case rising configurations are obtained analogously.) For falling outputs, the gates of all controlling devices on a path are set to their worst-case waveforms obtained either from a gate or a primary input. All other (non-controlling) paths are blocked by setting the gate of the lowest nMOS device in those paths to *GND*. Additionally, the gates of all other

Table 1. Delays for chains of static gates using various delay models (%Diff are relative to Pathmill).

| Gate | Pathmill Delay(ns) | Hspice | | | | WTA-Hspice | |
|---|---|---|---|---|---|---|---|
| | | Single-Input Transition | | Multiple-Input Transition | | | |
| | | Delay(ns) | %Diff | Delay(ns) | %Diff | Delay(ns) | %Diff |
| INV | 0.166 | 0.174 | 4.8 | 0.174 | 4.8 | 0.173 | 4.2 |
| NOR2 | 0.273 | 0.301 | 10.3 | 0.320 | 17.2 | 0.318 | 16.5 |
| NOR3 | 0.401 | 0.474 | 18.2 | 0.501 | 25.9 | 0.494 | 23.2 |
| NAND2 | 0.224 | 0.225 | 0.4 | 0.258 | 15.2 | 0.259 | 15.6 |
| NAND3 | 0.281 | 0.282 | 0.4 | 0.331 | 17.8 | 0.336 | 19.6 |
| AOI22 | 0.400 | 0.427 | 6.8 | 0.503 | 25.8 | 0.505 | 26.3 |
| AOI222 | 0.614 | 0.717 | 16.8 | 0.840 | 36.8 | 0.848 | 38.1 |
| AOI33 | 0.554 | 0.592 | 6.9 | 0.727 | 31.2 | 0.722 | 30.3 |
| Average | | | 8.1 | | 21.9 | | 21.7 |

devices are set so as to maximize the total capacitance that must be discharged.

Table 1 shows the delays obtained for chains of two static CMOS gates (one rising and the other falling) using the Pathmill slope model, the single-input and multiple-input transition models (using Hspice) and the WTA model. For the simple circuits used in the table, the single-input transition results are equivalent to an Hspice simulation of the device-level path outputted by Pathmill.

The gates were specified in a 0.18-micron (featuring a drawn channel length of 0.20 microns), 1.8V TSMC CMOS process. In this process the fanout-of-four delay for an inverter is 87 ps. The values in the table are for a chain of two gates, one rising and the other falling. Both gates in the chain have a fanout of 4 identical gates. Additional gates provide the input stimulus for the first gate and load on the second gate. In order to see the effect of multiple-input transitions, the waveforms for each input to a given gate are identical. Both inputs to a falling NAND2, for example, were constructed from the same rising NAND2. Pathmill results were obtained using technology files computed from typical-typical BSIM3 parameters for the process. The Hspice and WTA-Hspice results were obtained using the same typical-typical BSIM3 parameters

Table 1 shows that the slope model of Pathmill is typically optimistic relative to the single-input transition model. Using Hspice the single-input transition results average 8% greater than the Pathmill computed results. This result must necessarily be due to the slope approximation to the waveform for the single controlling input. The Hspice multiple-input transition results show significantly larger delays, averaging 22% over the Pathmill results. The results obtained using the WTA method in combination with the Hspice simulator are shown to be very close to the multiple-input Hspice results. Thus the WTA delay model described in this section provides a realistic and not overly-pessimistic delay.

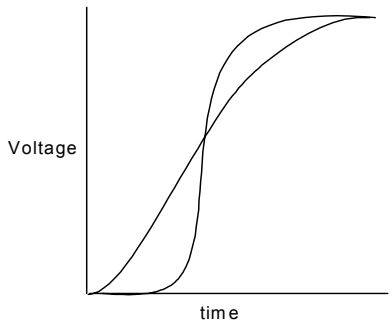



Figure 5. Example of ambiguous worst case waveforms

## 3. Propagation of Waveforms

Traditional timing analyzers determine the latest arriving input by measuring the delay at a single time-voltage point, usually the crossing of $V_{DD}/2$. This may be an incorrect assumption in many cases. An example is shown in Fig. 5, where a waveform with a short rise/fall time is chosen because it crosses $V_{DD}/2$ slightly after a waveform with a long rise/fall time. In this ambiguous case, the choice of the worst-case configuration and waveform depends upon what happens in subsequent gate levels. Blaauw, *et al*. looked at this issue in [6] using the slope approximation and allowed propagation of multiple events if ambiguity existed between the slopes resulting from two different paths in a gate. In the remainder of this section a similar approach to propagation will be described, the primary difference being that a waveform will be propagated instead of a slope.

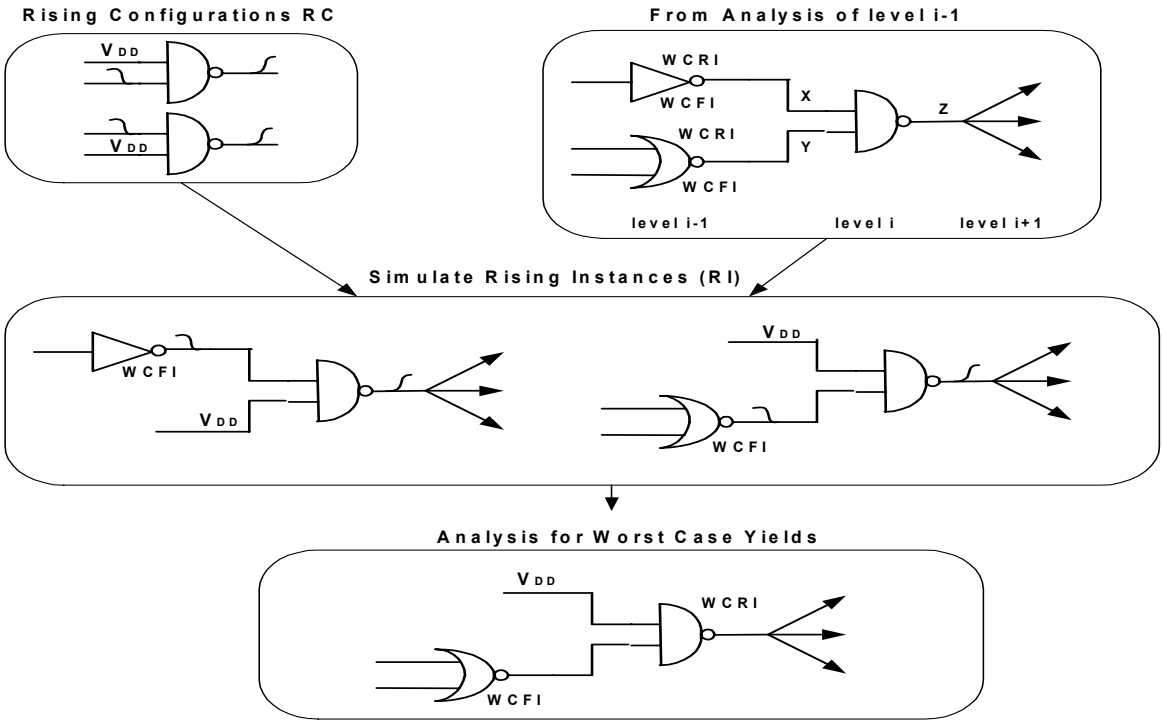



Figure 6. Example of worst case rising output analysis for a NAND2.

WTA employs a breadth-first search similar to that used in [2] and [4]. Given a levelized circuit, all the gates in a level are evaluated for worst case rising and falling output waveforms. These waveforms are then used to evaluate worst case behavior in subsequent levels.

Figure 6 shows an example of the determination of worst case rising waveforms of a NAND2. A NAND2 has two potential worst case rising configurations corresponding to falling waveforms on different inputs. During the analysis of the previous level, instances producing worst case falling waveforms WCFI (and worst case rising waveforms WCRI) for each of these inputs were determined. A simulation of both rising configurations of the NAND2 is performed and an analysis of the resulting output waveforms yields a worst case rising waveform for the NAND2.

Details of the simulation are as follows. For each gate in levels less than the current level, only the WCRI and WCFI are instantiated and simulated. The load for each gate is represented by the actual fanout gates in the netlist, each of which is configured to transition. The load for each of these fanout gates is additionally modeled as a single capacitor. This approach insures that the kickback effect of a transitioning fanout gate upon the gate of interest is taken into account. The waveform thus obtained for the gate of interest is stored as a voltage-controlled voltage source and used as an input waveform for subsequent gates.

Figure 7 shows the complete algorithm used by WTA to construct and propagate worst case waveforms from one level to another. For each gate in the current level, all instances contained in FI and RI are simulated in the same way as are instances in lower levels. Once simulation has

```
For each level from 1 to Num_levels
  For each gate in level
    Initialize list of falling gate instances
    FI to NULL
    For each falling config in FC
      Add to FI all instances formed from
      WCRI for each controlling input in config
    End For
    Initialize list of rising gate instances
    RI to NULL
    For each rising config in RC
      Add to RI all instances formed from
      WCFI for each controlling input in config
    End For
  End for
  Simulate all gate instances in WCRI and WCFI
  at levels less then current level and all gate
  instances in FI and RI for current level
  For each gate in level
    Initialize WCFI to (dominant) instance in FI
    with latest waveform transition at VDD/2
    For each instance in FI
      If (instance output waveform is later
      than dominant waveform at some voltage)
      add instance to WCFI
    End For
    Initialize WCRI to (dominant) instance in RI
       with latest waveform transition at VDD/2
    For each instance in RI
      If (instance output waveform is later
      than dominant waveform at SOME voltage)
      add instance to WCRI
    End For
  End For
End For
```

Figure 7. Algorithm for propagation of waveforms

completed, the waveforms for the FI and RI are analyzed to determine what subset must be included in WCRI and WCFI for that gate. Several methods of determining this subset are possible. Our method is to first include the leading waveform at VDD/2 (the one having the latest crossing), then add any other instances which cross the waveform of this dominant instance at other voltages. In our experience the effects of ambiguous worst case waveforms, where two or more instances need to be included in the WCFI or WCRI, are resolved in one and at most two subsequent levels and do not result in an explosion of instances in WCFI and WCRI.

The computational effort required by WTA is dependent upon the number of gates and the number of paths from output to ground (falling delay) and output to $V_{DD}$ (rising delay) for each gate. Each rising/falling path in a gate requires the construction and simulation of a single worst-case gate instance (unless there is ambiguity in the choice of worst case input waveforms) For most standard cell libraries, the complexity of cells is limited as is the list of possible paths. The 25-cell library known as lib2 has for example a maximum of 9 paths for any rising or falling cell output.

## 4. Experiments

Results of performing WTA calculations upon a set of ISCAS benchmarks are shown in Table 2. All benchmarks were synthesized and mapped using Synopsys. The target static CMOS library contained 25 cells. The cells are the same as in the lib2 library from SIS. They are 2-4 input NAND and NOR gates, XOR, XNOR, eight AOIs and eight OAIs, and an inverter. Gates were implemented in the TSMC 0.18-micron process described earlier.

Table 2 shows results for WTA using Hspice and Hsim as the underlying simulation engines in combination with typical-typical BSIM3 parameters for the TSMC process. Hsim is a circuit simulator from Nassda Corp. with controls that allow tradeoffs between running times and accuracy.

| Circuit | Pathmill | WTA- Hsim | WTA- Hspice |
|---|---|---|---|
| x3 | 0.738 | 0.794 (7.6%) | 0.804 (8.9%) |
| k2 | 1.121 | 1.234 (10.0%) | |
| i8 | 0.953 | 0.955 (0.1%) | 0.966 (1.3%) |
| rot | 0.998 | 1.069 (7.1%) | 1.080 (8.2%) |
| des | 1.182 | 1.402 (18.6%) | |
| dalu | 0.976 | 1.071 (9.7%) | |
| C5315 | 2.051 | 2.393 (16.7%) | |
| C7552 | 2.035 | 2.372 (16.6%) | |
| too_large | 0.741 | 0.816 (10.1%) | 0.830 (12.0%) |
| Average | | (10.7%) | |

Table 2. Delay comparisons on ISCAS benchmarks (ns)

(%'s are relative to Pathmill)

The results in Table 2 were obtained with the default settings for these controls. As is apparent from the 4 circuits for which WTA-Hspice results were obtained, there is little sacrifice in accuracy from the use of Hsim as the underlying circuit simulator. Running times for WTA-Hsim were about a factor of 100 less than for WTA-Hspice.

The WTA-Hsim delays are as much as 18% above the Pathmill delays, with the average being 10%. These differences are less than what was observed for the chains of gates where the inputs for each gate were set up to arrive at exactly the same time. In these random logic circuits there is considerable variation in arrival times, resulting in fewer cases where simultaneous input changes have as large an effect as with the chains of gates.

Perhaps the most surprising observation is that the difference between the Pathmill and WTA-Hsim delays varies considerably – from less than 1% to more than 18%. Clearly, the idea of allowing a fixed percentage of total delay (as determined by a conventional timing analyzer such as Pathmill) as a margin does not make sense. Circuits vary considerably in the accuracy of the slope model for critical path calculations.

| Circuit | #Gate | # Gate Levels | Pathmill (sec) | WTA-hsim (sec) |
|---|---|---|---|---|
| x3 | 583 | 13 | 3 | 123 |
| k2 | 1117 | 16 | 5 | 486 |
| i8 | 1117 | 11 | 5 | 181 |
| rot | 697 | 15 | 3 | 223 |
| des | 3236 | 18 | 16 | 1510 |
| dalu | 599 | 12 | 3 | 159 |
| C5315 | 1439 | 31 | 7 | 876 |
| C7552 | 1887 | 34 | 10 | 2176 |
| too_large | 268 | 11 | 2 | 62 |

Table 3. ISCAS benchmark characteristics and running times on a Sun Blade 1000.

Table 3 shows the number of gates and levels for each of the benchmarks, as well as the CPU seconds required for Pathmill and WTA-Hsim. WTA-Hsim is approximately 100 times slower than Pathmill, averaging 0.53 sec/gate on a Sun Blade 1000. Clearly, WTA-Hsim is more compute-intensive than Pathmill, however the computation times are still within reason for a design modification – analysis cycle. It is noteworthy that the WTA method is highly parallelizable since the evaluation of each gate configuration on a given level can be performed independently. Analysis of worst-case waveforms is minimal compared to the simulation times. This suggests considerable speedups are possible on parallel processors where circuit instances on the same level may be simulated on different processors.

## 5. Conclusions

Existing static timing analyzers make several assumptions about circuits, implicitly trading off accuracy for speed. In this paper we examined the validity of these assumptions, notably the slope approximation to waveforms, single-input transitions, and the choice of a propagating signal based on a single voltage-time point. We provide data on static CMOS gates that show delays obtained using these approximations can be optimistic by more than 30%. Some of this error results from the slope approximation, the remainder accounted for by assuming single-input transitions.

We propose a new approach, Waveform-based Timing Analysis (WTA), that employs a state-of-the-art circuit simulator as the underlying delay modeler. Using WTA, multiple-input transitions are handled in a realistic fashion, taking into account actual waveforms. When multiple controlling paths produce waveforms that are ambiguous as to worst case delay, all such waveforms are propagated forwards. We show that such an approach can achieve more accurate delay bounds than slope-based timing analyzers at a computation cost that still allows iterations between design modification and delay analysis.

## 6. Acknowledgements

We are grateful for the financial support provided by the National Science Foundation (NSF), the Semiconductor Research Corporation, MARCO, the NSF Center for the Design of Digital and Analog IC's (CDADIC), Boeing/DARPA, and the Intel Corporation. We also wish to thank Miodrag Vujkovic for providing the synthesized benchmarks, as well as the Pathmill technology files. Nassda Corp. is also acknowledged for the use of Hsim during the course of this research.

## 7. References

[1] J. Ousterhout, "A Switch-Level Timing Verifier for Digital MOS VLSI," *IEEE Trans. On Computer-Aided Design*, Vol. CAD-4, No. 3, July 1985, pp. 336-349.

[2] N. Jouppi, "Timing Analysis and Performance Improvement of MOS VLSI Designs," *IEEE Trans. On Computer Aided Design*, Vol. CAD-6, No. 4, July 1987, pp. 650-665.

[3] S. Hwang, Y. Kim and A. Newton, "An Accurate Delay Modeling Technique for Switch-level Timing Verification," *Proc. of the 23rd Design Automation Conference*, June 1986, pp. 227-233.

[4] M. Dagenais, S. Gaiotti and N. Rumin, "Transistor-Level Estimation of Worst-Case Delays in MOS VLSI Circuits," *IEEE Trans. On Computer-Aided Design*, Vol 11, No. 3, March 1992, pp. 384-395.

[5] V. Narayanan, B. Chappell, and B. Fleischer, "Static Timing Analysis for Self-Resetting Circuits," *1996 IEEE/ACM Intl. Conf. On Computer-Aided Design*, 1996 pp. 119-126.

[6] D. Blaauw, V. Zolotov, S. Sundareswaran, C. Oh, and R. Panda, "Slope Propagation in Static Timing Analysis," *2000 IEEE/ACM Intl. Conf. On Computer-Aided Design*, 2000 pp. 338-343.

[7] M. Desai and Y. Yen, "A Systematic Technique for Verifying Critical Path Delays in a 300 MHz Alpha CPU Design Using Circuit Simulation," *Proc. of the 33rd Design Automation Conference*, 1996.

[8] A. Kayssi, K. Sakallah, and T. Mudge, "The Impact of Signal Transition Time on Path Delay Computation," *IEEE Trans. on Circuits and Systems –II: Analog and Digital Signal Processing,* Vol. 40, No. 5, May 1993, pp. 302-309.

[9] S. Sun, D. Du and H Chen, "Efficient Timing Analysis for CMOS Circuits Considering Data Dependent Delays," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 17, No. 6, June 1998, pp. 546-552.