A Fast Approach for Static Timing Analysis Covering All PVT Corners[*]

Sari Onaissi
University of Toronto
Toronto, ON, Canada
sari@eecg.utoronto.ca

Feroze Taraporevala
Synopsys Inc.
Mountain View, CA, USA
ferozet@synopsys.com

Jinfeng Liu
Synopsys Inc.
Mountain View, CA, USA
jinfengl@synopsys.com

Farid Najm
University of Toronto
Toronto, ON, Canada
f.najm@utoronto.ca

ABSTRACT

The increasing sensitivity of circuit performance to process, temperature, and supply voltage (PVT) variations has led to an increase in the number of process corners that are required to verify circuit timing. Typically, designers attempt to reduce this computational load by choosing, based on experience, a subset of the available corners and running static timing analysis (STA) at only these corners. Although running a few corners, which are chosen beforehand, can lead to acceptable results in some cases, this is not always the case. Our results show that in the case of setup timing analysis, one can indeed bound circuit slacks across all corners by running a small number of corners. On the other hand, we show that this is not possible in the case of hold analysis. Instead, we present an alternative method for performing fast and accurate hold timing analysis which covers all corners. In this method a full timing run is performed for a small number of corners, and partial timing runs, which cover only the clock network, are performed for others. We then combine the results of the full and partial runs to find the worst-case hold slacks over all corners. Our results show that this method is accurate and can achieve much improved runtimes.

Categories and Subject Descriptors

B.8.2 [**Hardware**]: Performance and Reliability—*Performance Analysis and Design Aids*

General Terms

Performance, reliability, verification.

Keywords

Corner analysis, PVT corners, corner dominance, clock network.

1. INTRODUCTION

Accounting for the effects of process, supply voltage, and temperature (PVT) variations on circuit performance is an integral requirement in modern chip design flows. Typically, PVT variations are divided into two groups: chip-to-chip variations and on-chip variations, which are variations seen across a single chip. In practice, on-chip variations are considered less severe than chip-to-chip variations and are handled using derates, or margins, in static timing analysis (STA) runs [1]. On the other hand, chip-to-chip variations can cover significantly wider limits within the operating range, and therefore it is not practical to apply margining techniques to this type of variations. Instead, designers employ corner analysis, where STA runs are performed at a number of PVT settings, referred to as corners. By doing this, it becomes possible to capture a circuit's worst-case behavior in the PVT space without the excessive pessimism of margining. Of course, this comes at the runtime cost of performing multiple STA runs.

With continuous technology scaling, the effect of PVT variations on circuit timing has increased, and this has led to an increase in the number of corners required to cover the PVT space.

[*]This project was supported by Synopsys Inc. and the Natural Sciences and Engineering Research Council (NSERC) of Canada.

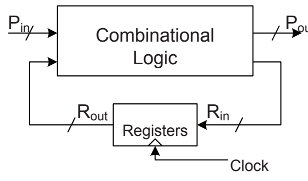

Figure 1: A general sequential circuit.

In addition, the trend towards the integration of more and more functional blocks on a single chip means that current designs run in an increasingly large number of modes. Because corner analysis must be performed in each of these modes, a large number of PVT corners can lead to an explosion in the number of timing runs. As a result, reducing the number of PVT corners at which timing must be checked has become a primary concern in today's design flows. The technique of statistical static timing analysis (SSTA), where PVT parameters are modeled as random variables (RV's) with known probability distribution functions, has been proposed as a solution to this problem [2, 3, 4]. By writing circuit element delays and signal arrival times as functions of the random process parameters, these also become RV's with known distributions. However, although SSTA is accurate when its underlying assumption of random process variables holds, this is not always the case. Recently, some methods have emerged where PVT variables are modeled simply as unknown variables with known bounds, instead of RV's with known distributions [5, 6, 7]. In these methods, circuit element delays are written as affine (linear) functions of these variables, which are then used to find the circuit's worst delays. In [5] and [6] the authors use these affine expressions to predict the corners which cause worst delays. On the other hand, in [7] the authors propagate these affine functions in the timing graph to find affine upper and lower bounds on circuit delays. However, the linear dependence of gate and interconnect delays on PVT parameters may not always hold, and this can lead to errors in the reported results of these methods. Therefore, we find that corner analysis is still the standard technique employed in industry, and that this is expected to remain the case in the foreseeable future.

Usually, foundries characterize logic cells and interconnect for a number of PVT corners and provide designers with these libraries. Designers can then use these libraries to run STA at various corners to verify circuit timing across the PVT space. This process of timing verification is generally performed at multiple stages of the design flow. For example, in addition to running STA at sign-off, timing runs are usually performed after logic synthesis and before place and route to help identify timing violations early in the flow. In particular, STA is heavily employed during circuit optimization, where circuit timing and/or power are "fixed" by iteratively introducing changes and running STA until satisfactory results are achieved. These repeated STA runs can become very expensive if timing is to be verified at all PVT corners everytime. In practice, designers rely on experience to choose a small subset of all available corners which they believe will bound circuit slacks at most endpoints. This set of corners is then used for non-final STA runs throughout the design flow. Then, at sign-off, a number of additional corners are added to this set to make sure that no violations were overlooked in this process. Although it is possible for a pre-determined small set of corners to account for worst-case slacks over all corners for setup analysis, this is not always the case for hold analysis. In fact, such a method can lead to overlooking a large percentage of hold timing violations. Therefore, it is important for designers to have an alternative method of running multi-corner STA that not only reduces runtimes but also provides accurate worst-case slack values.

In this work, we present such an approach for achieving fast and accurate timing verification covering all PVT corners. For setup analysis, we show that the industry practice of choosing a small set of corners can indeed capture worst-case slacks over all corners. On the other hand, we show that it is not possible to choose such a set for hold analysis. Instead, we present a method where full timing runs, i.e. covering the entire circuit, are performed for only a small number of corners. Then, much cheaper partial timing runs that cover only the clock network are used to estimate slacks at other corners. Our approach is primarily intented for non-final STA runs where it provides large runtime gains while maintaining a high degree of accuracy. As a result it can lead to significantly reduced turnaround times for multi-mode timing runs and circuit optimization iterations.

2. BACKGROUND

2.1 Static Timing Analysis

Consider the general logic circuit shown in Fig. 1, where R_{in} is the set of register input data pins, R_{out} is the set of register output data pins, P_{in} is the set of circuit input ports, and P_{out} is its set of output ports. We call $E = R_{in} \cup P_{out}$, the set of endpoints of this circuit, and $S = R_{out} \cup P_{in}$ its set of startpoints. At an endpoint $e \in E$, two types of timing constraints must be satisfied for every startpoint $s \in S$. The first type is the setup timing constraint which can be written, in its simplest form, as follows:

$$\max_{s \in S} (a_s + D_{s,e}) - a_e + t_{setup} < T \tag{1}$$

where a_s is the clock signal arrival time at the startpoint s, $D_{s,e}$ is the largest datapath delay between s and e, a_e is the clock signal arrival time at the endpoint e, and t_{setup} is the required setup time at e. The slack σ_{setup} of this constraint is the timing quantity by which this constraint is met, i.e.

$$\sigma_{setup} = T - \max_{s \in S} (a_s + D_{s,e}) + a_e - t_{setup} \tag{2}$$

Thus a positive slack means that the endpoint meets the constraint while a negative slack means that it does not. The second type of timing constraints is the hold timing constraint, written as follows:

$$\min_{s \in S} (a_s + d_{s,e}) - a_e > t_{hold} \tag{3}$$

where t_{hold} is the required hold time at e, and $d_{s,e}$ is the smallest datapath delay between s and e. The slack σ_{hold} for this constraint is written as:

$$\sigma_{hold} = \min_{s \in S} (a_s + d_{s,e}) - a_e - t_{hold} \tag{4}$$

As in the case of setup constraints, a positive slack for a hold constraint indicates that the constraint is met. A circuit is said to satisfy its timing requirements, if, and only if, the setup and hold timing constraints at each endpoint $e \in E$ are met. This verification is performed using STA, where the circuit is modeled as a directed graph with nodes representing the nets of the circuit and edges representing its timing arcs. Maximum and minimum delays are computed and propagated through this graph to find the maximum and minimum signal arrival times as well as the clock signal arrival times at all the endpoints of the circuit. These values are then used to verify that the timing constraints are met at all these endpoints. In corner analysis, this process of delay computation, delay propagation, and the evaluation of timing constraints is repeated for each PVT corner. An endpoint is deemed to have satisfied its timing constraints only if the slacks at this endpoint are positive for all corners.

2.2 Corner Dominance

Let the set of PVT corners of interest be $C = \{c_1, c_2, \ldots, c_n\}$ and the set of endpoints of the circuit under consideration be $E = \{e_1, e_2, \ldots, e_m\}$. A corner $c_i \in C$ is said to be dominant at an endpoint $e_j \in E$ if it is the corner resulting in the worst-case slack at e_j. For an arbitrary set of corners $C_A \subset C$, we call the percentage of endpoints in E where one of the corners in C_A is dominant the *endpoint coverage*, or simply *coverage*, of this set. A set $C_D \subset C$ is said to be a *dominant set* if it has a coverage of 100%, i.e. if it includes the dominant corners for all $e \in E$. If one were to know beforehand of such a set that is smaller than C, then runtime gains could be achieved by restricting corner analysis to C_D instead of C. In fact, for non-final STA runs it is acceptable to run a set $C_H \subset C$ which has high coverage even if it is not strictly dominant.

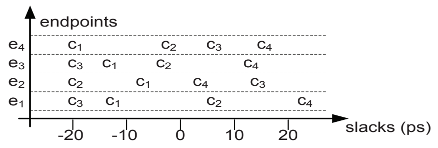

Figure 2: Endpoints with corresponding corner slacks.

Table 1: Test Circuits and Instances

Circuit	T1	T2	T3	T4	T5
# Inst	916K	933K	1,080K	1,145K	1,419K

The notion of dominance as defined above requires that a dominant set captures the exact worst-case slack at all endpoints. However, any industrial timing analyzer will have some margin of error, and allowing deviations within this margin will not introduce any additional inaccuracies. This concept can be be used to reduce the size of dominant sets by allowing a *dominance margin*, where for a given endpoint, any corner whose slack is within some acceptable margin from the worst-case slack can be considered dominant. Consider the example shown in Fig. 2, which shows the slacks at 4 endpoints of a circuit for the set $\{c_1, \ldots, c_4\}$. In order to capture the exact worst-case slacks at each of these endpoints, we would need to run corners c_1, c_2, and c_3, i.e. the smallest dominant set would be $C_D = \{c_1, c_2, c_3\}$. Now, if we allow a dominance margin of 10ps, our dominant set can be reduced to $\{c_1, c_2\}$. This is because the dominance margin allows c_1 to be dominant at e_1 and e_3, thereby increasing the coverage of the set $\{c_1, c_2\}$ to 100%.

3. EXPERIMENTAL SETUP

Due to the complexity involved in performing STA on industrial designs, a rigorous experimental setup is required to evaluate any proposed technique in this area. In what follows, we present the details of the setup used in this work.

3.1 Test Circuits and PVT Corners

Our timing runs were performed on a set of 15 industrial circuits mapped to a 45nm standard library. However, for brevity, we only show results for a reduced set of 5 circuits that capture the trends. These circuits range in size from 916,000 to 1.4 million instances. Table 1 shows our test circuits and their respective sizes.

In our tests, we use the standard set of PVT corners that is used to run corner analysis in industry. This set covers 3 device settings ("Best Case", "Typical Case", and "Worst Case") and 5 interconnect settings ("Min C", "Min RC", "Typical", "Max RC", and "Max C"), which are crossed to give a set of 15 PVT corners [1]. Let us first consider the device settings, shown in Table 2. The "Best Case" setting (fast process, high voltage, and low temperature) is the one which results in the fastest device performance, whereas the "Worst Case" setting (slow process, low voltage, and high temperature) results in the slowest performance. The "Typical Case", as the name suggests, gives the nominal performance. In the interest of clarity, our discussion does not consider the effect of temperature inversion, where it becomes possible for the slowest device performance to occur at a low temperature. However,

Table 2: Device Settings

Setting	Process	Supply Voltage	Temp.
Best Case (bc)	Fast	0.99V	0^oC
Typical Case (tc)	Typical	0.90V	25^oC
Worst Case (wc)	Slow	0.81V	125^oC

Table 3: Interconnect Settings

Setting	Capacitance	Resistance
Min C	Low	> Nominal
Min RC	> Nominal	Low
Typical	Nominal	Nominal
Max RC	< Nominal	High
Max C	High	< Nominal

Table 4: PVT Corners

	Min C	Min RC	Typical	Max RC	Max C
bc	c_1	c_2	c_3	c_4	c_5
tc	c_6	c_7	c_8	c_9	c_{10}
wc	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}

our work can be easily extended to handle this effect by adding an extra device setting with slow process, low voltage, and low temperature to the list in Table 2 [1].

Now let us look at the interconnect settings, which are shown in Table 3. The settings "Min RC" and "Max RC" result in the lowest and highest interconnect *RC* products respectively, whereas "Min C" and "Max C" result in the lowest and highest interconnect capacitance values [1]. Therefore, for paths with long interconnects, corners "Min RC" and "Max RC" will give the smallest and largest path delays respectively. On the other hand, for paths with short nets, it is corners "Min C" and "Max C" that cause the extreme path delays [1].

The device and interconnect settings described above are combined to get the set C of 15 PVT corners shown in Table 4. In this set, we define the three subsets $C_b = \{c_1, c_2, c_3, c_4, c_5\}$, $C_t = \{c_6, c_7, c_8, c_9, c_{10}\}$, and $C_w = \{c_{11}, c_{12}, c_{13}, c_{14}, c_{15}\}$. The set C includes the corners commonly available for performing timing verification in industry, and typically designers have to make some difficult choices on how to reduce it to a more manageable set as described earlier. Note that identifying the corners in this set which result in extreme *signal arrival times* is not very challenging. That is, it would be enough to consider corners c_1 and c_2 to find minimum signal arrival times, and corners c_{14} and c_{15} to find maximum arrival times. Therefore, we call $C_m = \{c_1, c_2\}$ the set of "fast" corners and $C_M = \{c_{14}, c_{15}\}$ the set of "slow" corners. Although these two sets can be used to find extreme signal arrival times, things are not necessarily this simple when one is looking for the worst-case *slacks*.

3.2 Practical Issues

In our timing runs margining is used to account for on-chip variations. In the case of setup analysis, this is done by adding a margin of 5% to maximum signal arrival times, and a margin of −5% to capture-clock signal arrival times. On the other hand, in the case of hold analysis, this is done by using a margin of −5% with minimum signal arrival times, and 5% with capture-clock signal arrival times. These margins are commonly referred to as *derates*. In cases where the clock launch and capture paths at a given endpoint share common logic cells and interconnect, derating the delays of common elements differently can result in pessimistic slack values. Therefore, a clock reconvergence pessimism removal (CRPR) technique was used to reduce this pessimism as is customary in industry. In the interest of brevity, a lot of the discussion and arguments in this work assume flop-to-flop paths, i.e. paths that don't start or terminate at circuit ports. Also, unless stated otherwise, our timing runs exclude paths that start at circuit input ports or terminate at circuit output ports. The reason for this is that in the case of logic blocks these paths are typically handled at a higher level, i.e. when the block timing is analyzed in its context. As for a complete circuit, these will be handled separately due to the extra complexity introduced by external factors such as pin capacitances and connections with other circuits. Another practical issue of importance in our results is that of failing and non-failing endpoints. Recall that a dominant corner set is one which can account for worst-case slacks at all the endpoints of a circuit. However, designers are only interested in a set which accounts for worst-case slacks at endpoints that are failing or "close" to failing. Therefore it is acceptable to cover only such endpoints. Because timing failures depend on external parameters such as clock periods and path derates, different circuits will have different percentages of failing endpoints. Therefore, looking at failing endpoints only will lead to skewed results if some circuits have very low or high percentages of failing endpoints. On the other hand, looking at all the endpoints will not address the real issue of timing violations, and might miss runtime improvement opportunities by attempting to capture the behavior of safe endpoints that are of no interest to designers. In our experiments, we accommodate both of these concerns by collecting data from the top-25% of endpoints with the worst slacks, regardless of whether they fail any timing constraints or not. This relatively high percentage of endpoints allows us to focus on the behavior of the endpoints with the worst slacks without the risk of skewed results from circuits with few timing violations. Therefore, in all our tests, we consider these to be the set of failing endpoints and we only collect data from these endpoints.

Table 5: Margins vs. minimum clock periods

Circuit	T1	T2	T3	T4	T5
T_{min}	4ns	3.9ns	2.2ns	3ns	1ns
$Margin/T_{min}$ (setup)	0.8%	0.8%	1.4%	1.0%	3.0%
$Margin/T_{min}$ (hold)	0.3%	0.3%	0.5%	0.3%	1.0%

Table 6: Maximum arrival-time dominance

	Circuits				
Margins	T1	T2	T3	T4	T5
0ps	c_{15}	c_{14}, c_{15}	c_{14}, c_{15}	c_{14}, c_{15}	c_{14}, c_{15}
10ps	c_{15}	c_{14}, c_{15}	c_{14}, c_{15}	c_{14}, c_{15}	c_{14}, c_{15}
20ps	c_{15}	c_{14}, c_{15}	c_{14}, c_{15}	c_{14}, c_{15}	c_{14}, c_{15}
30ps	c_{15}	c_{14}, c_{15}	c_{14}, c_{15}	c_{14}, c_{15}	c_{14}, c_{15}

Recall that in our analysis, we allow a dominance margin that is acceptable to designers. In this work, results are presented for multiple dominance margin values, where for setup analysis the largest dominance margin allowed is 30ps, and in the case of hold analysis it is 10ps. In order to put these numbers in perspective, Table 5 shows the smallest clock period of each of our test circuits, and how the maximum setup and hold dominance margins compare to these clock periods. This table shows that our margins represent a very small percentage of clock periods, which would be acceptable in an industrial setting.

4. ARRIVAL-TIME DOMINANCE

For an endpoint $e \in E$, the maximum signal arrival time is written as $W_e = \max_{s \in S} (a_s + D_{s,e})$ and the minimum signal arrival time is written as $w_e = \min_{s \in S} (a_s + d_{s,e})$. These arrival times are used to compute setup and hold slacks respectively at e as shown in (2) and (4). In Section 3.1, we claim that it is enough to look at corners c_{14} or c_{15} to find the largest value of W_e, whereas corners c_1 and c_2 can be used to get the smallest w_e. In order to check that, we ran each of our test circuits at all PVT corners in set C. Then, for each circuit we collected the maximum and minimum arrival times at all endpoints for all corners. Table 6 and Table 7 show corner dominance for maximum arrival times (W_e) and minimum arrival times (w_e), respectively. As expected, we see that the corners in set $C_M = \{c_{14}, c_{15}\}$ are the main corners which cause the largest signal arrival times in all of our circuits. We also see that set $C_m = \{c_1, c_2\}$ results in the smallest signal arrival times at the endpoints of these circuits.

5. SLACK DOMINANCE

So far, we have seen that signal arrival time dominance is predictable. This is mainly because signal arrival times are path delays found by adding logic cell and interconnect delays, which have known dependencies on PVT corners. However, slack is not computed in the same way, but rather by finding the difference between launch and capture path delays. Therefore, one cannot assume that slack dominance will follow arrival time dominance. In this section, we look at slack dominance for both setup and hold analyses. We show that in the case of setup analysis, worst-case slacks are indeed determined by the same corners which cause maxiumum arrival times. On the other hand we show that the same cannot be said of hold slacks and applying such a technique is not possible in that case.

Table 7: Minimum arrival-time dominance

	Circuits				
Margins	T1	T2	T3	T4	T5
0ps	c_1, c_2	c_1, c_2	c_1, c_2	c_2	c_1, c_2
5ps	c_2	c_1, c_2	c_1, c_2	c_2	c_1, c_2
7.5ps	c_2	c_1, c_2	c_1, c_2	c_2	c_1, c_2
10ps	c_2	c_1, c_2	c_1, c_2	c_2	c_1, c_2

Table 8: Setup slack coverage of C_M

	Circuits				
Margins	T1	T2	T3	T4	T5
0ps	99.8%	99.0%	99.9%	87.6%	100%
10ps	100%	99.8%	100%	97.9%	100%
20ps	100%	100%	100%	99.8%	100%
30ps	100%	100%	100%	99.9%	100%

Table 9: Endpoints with startpoint changes

Circuit	T1	T2	T3	T4	T5
% SP Changes	2.1%	2.7%	11.4%	0.5%	13.1%

5.1 Setup Analysis Dominance

Recall that for setup analysis, the slack at an endpoint e is written as shown in (2). Changing PVT corners affects the values of the maximum arrival time $W_e = \max_{s\in S}(a_s + D_{s,e})$, the clock signal arrival time a_e, and the required setup time t_{setup}. In the case of setup analysis, the maximum datapath delay $D_{s,e}$ is expected to be relatively large, while a_s and a_e will be of the same order of magnitude for a reasonably balanced clock tree. This means that the maximum arrival time W_e will be considerably larger than $(a_e - t_{setup})$. If this is not the case, i.e. if the value of W_e is comparable to that of $a_e - t_{setup}$, then these two terms will almost cancel each other out and we would have a large slack σ_{setup}. In such a case, the endpoint will be safe and it is not of much interest to designers. Therefore, for failing or "almost failing" endpoints, $D_{s,e}$ will be large, and the signal arrival time will dominate the slack expression. As a result, the changes in maximum arrival time W_e will dominate changes in slack when we move from one PVT corner to another. Because of this, the worst-case setup slacks are expected to occur at the corners which cause the largest signal arrival times, i.e. the "slow" corners. That is why it is common practice in industry to verify setup constraints by running only these corners.

Recall that $C_M = \{c_{14}, c_{15}\}$ is the set of corners that cause the largest signal arrival times. Table 8 shows, for each circuit, the endpoint coverage of the set C_M for different dominance margins. This table shows that C_M dominates over 99.9% of all endpoints for a margin of 30ps, and over 97.9% of all endpoints for a margin of just 10ps. For non-final STA runs, the coverage provided by C_M is considered more than adequate, and it is enough to run these 2 pre-determined corners to verify circuit setup constraints. Therefore, the industry practice of using "slow" corners as a dominant set for setup slacks is accurate and justified.

5.2 Hold Analysis Dominance

Consider the expression of the hold slack at an endpoint e, shown in (4). Although it is possible for the minimum arrival time $w_e = \min_{s\in S}(a_s + d_{s,e})$ to dominate $a_e + t_{hold}$, one cannot argue that this will be the norm as in the case of setup analysis. The main reason for this is that the datapath delay $d_{s,e}$ will typically be much smaller than in the case of setup analysis, thus leading to more comparable values of w_e and $a_e + t_{hold}$. Moreover, if w_e is much larger than $a_e + t_{hold}$ then the slack will be large and the endpoint will not be of much interest. Therefore, here we see that endpoints fail when signal arrival times do **not** dominate the slack expression rather than when they do as in the case of setup analysis. This means that, unlike the setup analysis case, we should not expect the corners which dominate worst-case minimum arrival times to always dominate hold-slacks as well. In fact, slack dominance for hold analysis can vary considerably for different circuits as can be demonstrated for our test circuits.

Table 10 shows the endpoint coverage of the three sets C_m (the set of dominant corners for minimum arrival times), C_b (the set of corners with *bc* device setting), and C_w (the set of corners with *wc* device setting) defined in Section 3.1. Here the objective is to demonstrate that dominant sets for hold analysis can be quite different from one circuit to another. Note that we do not show any results for the set C_t only because it has extremely low coverage for all of our test circuits. First, let us look at the coverage of set C_m. Although this set covers a high percentage of endpoints for circuits T1 and T3 (especially for larger margins), we see that for other circuits this percentage drops to much lower levels, e.g. a coverage of 68.5% of endpoints of circuit T2 for a margin of 10ps. Therefore it is not possible to claim that this set can be used to account for worst-case hold slacks for all of our circuits. Now consider set C_b. This set improves on the coverage of C_m in many cases, namely for circuits T2-T5. This means that some corners which are in C_b but not in C_m can be dominant at many endpoints of these circuits. However, although C_b achieves very good results for some of the circuits, its coverage remains very low for T2. Therefore, this set also cannot be used to account for hold slacks for all of the circuits. Another complication in the case of hold analysis is that corners in C_w can also be dominant at many of the endpoints. This is clearly seen in the cases of T2-T5, where the percentage of endpoints dominated by this set exceeds 4% and can be up to 26%. These results clearly show that the set of dominant corners can be quite different from one circuit to another. In some cases the set C_m is enough, whereas for other circuits this is not enough and other corners have to be added from C_b and C_w. Thus, designers cannot simply restrict the analysis to two pre-chosen corners as in the case of setup analysis, because doing so would risk overlooking a lot of timing violations that can occur at other corners.

6. A METHOD FOR HOLD ANALYSIS

In the previous section, we showed that the set of dominant corners for hold slacks can vary considerably from one circuit to another. Therefore, any set that is guaranteed to bound the hold slacks of a circuit has to include a large number of corners. In this section, we present an alternative method where full timing runs are performed for only a small number of corners and then partial runs, which cover only the clock network, are used to estimate slacks at remaining corners.

6.1 Hold Slack Estimation

Consider an arbitrary endpoint e of a circuit and a PVT corner c_i. Let s' be the startpoint of the path with the worst-case hold slack at e for an STA run at c_i. Thus, the minimum signal arrival time at e for corner c_i is written as:

$$w_e^i = \min_{s\in S}\left(a_s^i + d_{s,e}^i\right) = a_{s'}^i + d_{s',e}^i \tag{5}$$

where $a_{s'}^i$ is the clock signal arrival time at s', and $d_{s',e}^i$ is the smallest datapath delay between s' and e at c_i. Therefore, the hold slack for e at corner c_i can be written as:

$$\sigma_{hold}^i = a_{s'}^i + d_{s',e}^i - a_e^i - t_{hold}^i = k_{s',e}^i + d_{s',e}^i - t_{hold}^i \tag{6}$$

where $k_{s',e}^i = a_{s'}^i - a_e^i$, is the clock skew between s' and e. Assume that for a timing run at another corner c_j, s' was also found to be the startpoint of the path causing the worst-case slack at e. As a result, the slack for e at corner c_j is written as:

$$\sigma_{hold}^j = k_{s',e}^j + d_{s',e}^j - t_{hold}^j \tag{7}$$

In this case, the change in hold slack Δ_{ij}, where $\sigma_{hold}^j = \sigma_{hold}^i + \Delta_{ij}$, can be written as follows:

$$\Delta_{ij} = \left(k_{s',e}^j - k_{s',e}^i\right) + \left(d_{s',e}^j - d_{s',e}^i\right) + \left(t_{hold}^i - t_{hold}^j\right) \tag{8}$$

Knowing the exact value of Δ_{ij} would require performing an STA run at c_j. However, consider an approximation $\tilde{\Delta}_{ij}$ of Δ_{ij} written as the skew difference between corners c_i and c_j as follows:

$$\tilde{\Delta}_{ij} = k_{s',e}^j - k_{s',e}^i \tag{9}$$

and an estimate $\tilde{\sigma}_{hold}^j$ of the slack σ_{hold}^j found by substituting $\tilde{\Delta}_{ij}$ for Δ_{ij}:

$$\tilde{\sigma}_{hold}^j = \sigma_{hold}^i + \tilde{\Delta}_{ij} \tag{10}$$

That is, the slack at corner c_i along with the clock skew difference between these two corners is used to estimate the slack at c_j. Such an estimate can be obtained by running STA on the full circuit for c_i to obtain the slacks and skews at this corner, and then performing a timing run that propagates delays only in the clock network for c_j to find only skews this time. In this operation,

Table 10: Hold slack coverage

	Circuits														
	T1			T2			T3			T4			T5		
Margins	C_m	C_b	C_w	C_m	C_b	C_w	C_m	C_b	C_w	C_m	C_b	C_w	C_m	C_b	C_w
0ps	34.4%	99.5%	0.4%	33.0%	75.5%	22.9%	63.1%	95.3%	4.2%	34.0%	95.2%	4.1%	41.7%	93.3%	6.5%
5ps	97.5%	99.7%	0.6%	59.4%	77.9%	24.7%	92.3%	95.6%	4.3%	76.3%	96.0%	4.7%	83.5%	93.7%	6.8%
7.5ps	98.7%	99.8%	0.7%	64.5%	78.5%	25.3%	93.1%	95.8%	4.4%	83.9%	96.3%	5.0%	85.9%	93.9%	7.0%
10ps	99.2%	99.8%	0.9%	68.5%	79.0%	26.0%	93.6%	95.9%	4.5%	89.0%	96.4%	5.3%	88.3%	94.0%	7.0%

Table 11: Method results

	Circuits														
	T1			T2			T3			T4			T5		
Margins	Ω_A	Ω_T	M	Ω_A	Ω_T	M	Ω_A	Ω_T	M	Ω_A	Ω_T	M	Ω_A	Ω_T	M
0ps	26.2%	100%	72.0%	11.1%	98.4%	51.8%	45.2%	99.4%	88.6%	16.2%	99.3%	60.3%	31.3%	99.7%	68.0%
5ps	85.6%	100%	99.8%	56.9%	100%	99.7%	94.7%	99.9%	99.9%	71.7%	100%	100%	76.1%	100%	98.6%
7.5ps	95.7%	100%	99.9%	65.6%	100%	99.8%	96.2%	100%	99.9%	82.5%	100%	100%	80.8%	100%	99.1%
10ps	98.7%	100%	99.9%	71.8%	100%	99.9%	96.9%	100%	99.9%	89.0%	100%	100%	85.8%	100%	99.4%

c_i is referred to as the *anchor corner* and c_j is called the *partial corner*.

The difference between σ^j_{hold} and $\tilde{\sigma}^j_{hold}$, i.e. the error of this approximation, is essentially the difference between Δ_{ij} and $\tilde{\Delta}_{ij}$ and can be written as:

$$\sigma^j_{hold} - \tilde{\sigma}^j_{hold} = \left(d^j_{s',e} - d^i_{s',e}\right) + \left(t^i_{hold} - t^j_{hold}\right) \qquad (11)$$

Typically, the differences in required hold times are very small and do not lead to significant errors. However, the term $e_d = \left(d^j_{s',e} - d^i_{s',e}\right)$ can lead to inaccuracies. If $d^j_{s',e} < d^i_{s',e}$, the estimated slack would be larger than the actual slack, and this would lead to an optimistic slack value. On the other hand, if e_d is positive and very large, the estimated slack $\tilde{\sigma}^j_{hold}$ would be much smaller than the actual slack σ^j_{hold}, and our estimate would be overly pessimistic. Therefore, in order for this slack estimate to be useful it is required that i) $e_d > 0$, i.e. $d^j_{s',e} > d^i_{s',e}$ and ii) the value of e_d should not be very large.

The first condition above requires always using anchor corners that have smaller datapath delays than those of partial corners. This means that, for example, we could use c_6 (tc, Min C) as an anchor corner to estimate slacks at the partial corner c_{11} (wc, Min C), but not at c_1 (bc, Min C). On the other hand, the second condition requires that we limit the datapath delay spread between anchor corners and partial corners. In our approach, we do this by dividing our set of corners C into the three subsets C_b, C_t, and C_w from Section 3.1 where, for each of these groups, one corner is designated as the anchor corner and is used to estimate slacks at the remaining corners in the same set. The division of C into these sets avoids potentially large spreads in datapath delays between anchor and partial corners, and prevents excessive pessimism that results from such spreads. The anchor corners chosen for these sets are c_1 (bc, Min C) for set C_b, c_6 (tc, Min C) for C_t, and c_{11} (wc, Min C) for C_w. Note that because functional logic paths generally tend to have short nets, corners with interconnect setting "Min C" rather than "Min RC" are the ones expected to result in the smallest datapath delays. Thus, these are chosen as the anchor corners for sets C_b, C_t, and C_w, and are used to estimate slacks at the partial corners in their corresponding sets.

In our discussion above, we presented the method used to estimate the worst-case slacks at the partial corner c_j, using a full timing run at the anchor corner c_i. However, in this discussion we assumed that, for an endpoint e, the worst-case startpoint s' is the same at both the anchor corner c_i and the partial corner c_j. Indeed, in practice it is mostly the case that the worst-slack startpoints for each endpoint are the same across all PVT corners. For instance, Table 9 shows, for each of our test circuits, the percentage of endpoints whose worst-slack startpoints can differ between any two corners in C. This table shows that the percentage of endpoints which change startpoints across PVT corners is typically small. Therefore, in the majority of cases our assumption of "startpoint stability" holds. When startpoint changes do occur between corners, our slack estimate becomes that of the worst-case slack from paths originating at s'. One solution to this would be to find at c_i the worst-case slacks for paths originating at a number of startpoints, and then to estimate the slacks at c_j for paths originating from each of these. However, such a solution is not necessary mainly because the estimates of worst-case slack from s' will be close enough to the actual worst-case slacks and will still provide good endpoint coverage as our results will show.

6.2 Results

6.2.1 Accuracy

The accuracy of our method can be evaluated by comparing its estimated slacks to actual worst-case slacks found by running all corners. As stated earlier, our corners were divided into the three sets C_b, C_t, and C_w. In our method, the slacks for each of these sets are found by performing a full STA run at only the anchor corner and partial runs at the remaining corners. On the other hand, in standard corner analysis, full timing runs are performed for all the corners in these sets. Now, recall that the set C_t has very low endpoint coverage for all of our test circuits. Therefore, the set $\Omega_T = C_b \cup C_w$ achieves a coverage of close to 100% for all test circuits, as we will show shortly. Because of that, our method was used to estimate the worst-case slacks at corners in C_b and C_w only, and the results were compared to standard corner analysis over Ω_T. However, note that including the set C_t in both our approach and in standard corner analysis does not affect the accuracy or the runtime improvements of our method.

Recall that our method computes a worst-case slack estimate for all the endpoints of a circuit. Here, we report, for each circuit, the endpoint coverage of our method found by comparing our estimated worst-case slacks to actual worst-case slacks found by running Ω_T. Our method is deemed to have "covered" an endpoint, if the worst-case slack estimate is found to be less than or equal to the actual worst-case slack, i.e. if the slack estimate is conservative. Moreover, we also report results where a dominance margin is used. In this case, an endpoint is also considered "covered" if the estimated worst-case slack is optimistic but within the dominance margin from the actual worst-case slack. In Table 11 we show the endpoint coverage of the set of anchor corners $\Omega_A = \{c_1, c_{11}\}$, the coverage of set Ω_T, and that of our method (M). The contribution of our method can be seen by looking at the improvement it shows over Ω_A and its accuracy can be observed by comparing its coverage to that of Ω_T. Notice that in some of our testcases the set Ω_A covers a high percentage of endpoints, especially for dominance margins of 7.5ps and 10ps (e.g. T1 and T3). However, this coverage becomes lower when the dominance margin is reduced to 5ps. In these cases, we see that our method provides coverage which is very close to that of Ω_T for a dominance margin of 5ps. Moreover, as the coverage of Ω_A becomes low for circuits such as T2, T4, and T5 our method still shows coverage of over 98.6% for a 5ps dominance margin. In general, all of the testcases show that our method gives coverage that is very close to that of Ω_T for the small margin of 5ps. Therefore, our method would be very effective at finding close to all hold timing violations with high accuracy.

6.2.2 Crosstalk

Typically, extracted interconnect capacitances are either grounded capacitances or coupling capacitances between neighboring nets. For delay calculations, the equivalent value of a coupling capacitance at a switching net (the *victim*) depends on whether the coupled net (the *aggressor*) is switching at the same time, and, if so, on whether this switching activity is in the same direction as that of the victim. More specifically, an aggressor switching

Table 12: Method results with cross-talk analysis

	Circuits														
	T1			T2			T3			T4			T5		
Margins	Ω_A	Ω_T	M	Ω_A	Ω_T	M	Ω_A	Ω_T	M	Ω_A	Ω_T	M	Ω_A	Ω_T	M
0ps	16.2%	99.6%	50.4%	6.8%	96.3%	39.3%	26.7%	99.6%	63.3%	7.7%	99.2%	47.8%	8.9%	99.7%	45.2%
5ps	87.6%	99.9%	98.9%	31.0%	99.7%	94.6%	75.0%	99.9%	97.1%	54.1%	99.9%	99.4%	44.3%	100%	86.5%
7.5ps	93.0%	100%	99.5%	41.9%	100%	97.3%	78.3%	100%	97.9%	67.0%	100%	99.7%	52.2%	100%	89.7%
10ps	94.9%	100%	99.7%	50.7%	100%	98.3%	81.5%	100%	98.4%	76.6%	100%	99.9%	59.5%	100%	92.1%

Table 13: Conservatism without and with SI

Circuit	Mean	S. Dev.	Worst
T1	0ps (−1ps)	0ps (1ps)	−9ps (−8ps)
T2	−1ps (−2ps)	2ps (2ps)	−26ps (−28ps)
T3	−2ps (−6ps)	4ps (7ps)	−26ps (−39ps)
T4	−1ps (−1ps)	1ps (3ps)	−36ps (−37ps)
T5	−2ps (−3ps)	3ps (3ps)	−38ps (−28ps)

in the opposite direction of a victim leads to a larger equivalent capacitance at the victim than a non-switching aggressor does. On the other hand an aggressor switching in the same direction of a victim results in a smaller capacitance than that caused by a non-switching aggressor. Modern timing analyzers account for this effect by integrating *crosstalk analysis*, commonly referred to as *signal integrity (SI) analysis*, into STA runs. In SI analysis, the maximum delays of certain nets and their driving cells are increased if neighboring nets can simultaneously switch in the opposite direction, whereas minimum delays are decreased if these can switch in the same direction. However, this dependency of delay values on the switching activity of neighboring nets can lead to two possible complications for our method. First, recall that in order for our technique to be conservative we must choose, for each set of corners, an anchor corner that leads to the smallest datapath delays. Under SI analysis, crosstalk that happens at a partial corner, but not at the anchor corner, can lead to some endpoints where the minimum signal arrival times occur at the partial corner. The second possible complication is that in our partial timing runs, delays are not propagated in functional logic blocks, and therefore the effects of crosstalk on the clock network from functional block aggressors cannot be accurately computed. However, in spite of these two issues, our method still works well with standard SI analysis, and this is mainly due to the following reasons. Although crosstalk analysis can have a significant effect on datapath delays, this is not very likely to happen for a partial corner without also happening for the anchor corner. Even if this does happen, datapath delays at the anchor corners are smaller than those of partial corners to start with, and any crosstalk speedup at the partial corners is offset by this difference. Moreover, designers typically take extra care to shield the clock network from the effects of crosstalk, and as a result, our partial timing runs will still lead to generally accurate skews. In Table 12 we show the coverage results of our method with SI analysis. Although we see that the coverage of our method is slightly reduced here, this table shows that its coverage is still very high. For circuits T1-T4, the coverage is still above 98.4%, whereas it drops to 92.1% for T5. Moreover, we see that, for all of the circuits, the coverage provided by our method is much larger than that provided by Ω_A and is very close to Ω_T.

6.2.3 Conservatism

As noted earlier, the slack estimates found using our method can be conservative. In what follows we present statistics of our method's conservatism, i.e. statistics of estimated worst-case slacks which are smaller than actual worst-case slacks. Table 13 shows, for all conservative estimated slacks, i) the mean value by which actual worst-case slacks are underestimated, ii) the standard deviation of this underestimation, and iii) the worst such underestimation. Results for STA runs with SI analysis are shown in brackets. Our results show that the means and standard deviations of conservative underestimation are quite low for all the circuits. The results also show that estimates can be conservative by up to −40ps (T3-T5). However, the low means and standard deviations show that these are outliers, and that the majority of conservative estimates remain small. Note that even when our estimated slacks do not add a lot to the coverage of set Ω_A (the case of T1 in Table 11), the conservatism of our results remains small.

Table 14: Method runtimes

Circuit	T1	T2	T3	T4	T5
L_{full}	490.9s	486.4s	1157.0s	2466.6s	1358.6s
$L_{partial}$	9.0s	11.9s	14.8s	10.3s	25.3s
L_{over}	153.1s	163.6s	232.0s	265.5s	288.4s
S_{simple}	2.5×	2.4×	2.9×	3.6×	2.8×
S_{shared}	4.2×	4.1×	4.4×	4.7×	4.3×

6.2.4 Runtimes

We have seen that our method is indeed accurate, and it remains to be shown that it can provide significant speedups. Table 14 gives the runtime improvements that can be achieved by running our method instead of running all corners in Ω_T. This table shows, for each circuit, the time it takes to complete a full STA run (L_{full}), the time it takes to complete a partial STA run ($L_{partial}$), and the overhead time needed to load the circuit and the parasitics (L_{over}). In addition, this table shows the speedups achieved by running our method instead of running all corners if each partial run also runs the overhead tasks (S_{simple}). We also show the speedup that is possible if the overhead tasks are shared (S_{shared}). An example of this would be if all the runs are performed in the same session.

The first observation from Table 14 is that the runtimes of partial STA runs are much smaller than the runtimes of overhead operations, and are negligible when compared to runtimes of full STA runs. This table also shows that running our method can produce runtime improvements of 2.4× to 3.6× if the overhead is included in each run. On the other hand, if the overhead is shared among all timing runs, runtime improvements of over 4.1× can be achieved. These STA runtime gains would translate to huge runtime gains in cases where timing verification has to be performed many times such as in circuit optimization, or in cases where timing has to be verified in a large number of modes.

7. CONCLUSION

In this work, we presented an approach for running STA which covers all PVT corners. Our approach is primarily intented for non-final STA runs where it provides large runtime gains while maintaining a high degree of accuracy. In the case of setup analysis, we showed that the industry practice of using a small subset of all PVT corners to find the worst-case slacks of a circuit is indeed accurate. This is not the case in hold analysis. Instead we presented an approach that uses a limited number of full STA runs, and much cheaper partial timing runs to find worst-case hold slack estimates at all corners. Our method produces large runtime gains and gives very accurate results.

8. REFERENCES

[1] J. Bhasker and R. Chadha. *Static timing analysis for nanometer designs: a practical approach.* Springer, New York, 1st edition, 2009.

[2] H. Chang and S. Sapatnekar. Statistical timing alaysis considering spatial correlations using a single PERT-line traversal. In *ICCAD*, pages 621–625, November 9-13 2003.

[3] J. A. G. Jess, K. Kalafala, W. R. Naidu, R. H. J. M. Otten, and C. Visweswariah. Statistical timing for parametric yield prediction of digital integrated circuits. In *Design Automation Conference*, pages 932–937, June 2-6 2003.

[4] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Waler, and S. Narayan. First-order incremental block-based statistical timing analysis. In *DAC*, pages 331–336, June 2004.

[5] L.G. e Silva, L.M. Silveira, and J.R. Phillips. Efficient computation of the worst-delay corner. In *DATE*, pages 1–6, 2007.

[6] J. J. Nian, S. H. Tsai, and C. Y. Huang. A unified multi-corner multi-mode static timing analysis engine. In *ASP-DAC*, pages 669–674, 2010.

[7] S. Onaissi and F. N. Najm. A linear-time approach for static timing analysis covering all process corners. *TCAD*, 27(7):1291–1304, 2008.