Conditional Delays for Negative Limit Timing Checks in Event Driven Simulation

Nadeem Kalil and David Roberts
Cadence Design System, Inc.
270 Billerica Rd,
Chelmsford, MA 01824

***Abstract*- In modern VLSI technology, reduced timing margins and timing closure pressure result in an increase in the number and complexity of Negative Timing Checks (NTC). To enable event driven simulation, timing checks with negative limit values are typically transformed by delaying the corresponding input signals so as to restore the normal sequence of the data and reference events. Concurrently the NTC limit values are adjusted to compensate for the added delays. We present a solution methodology based on linear programming and the well-established Simplex method to determine the input signal delay values. We also introduce conditional delays, a new construct that associates different delays with the same input signal depending on the timing check condition truth values. Finally, a new methodology to generate NTC delays directly by the static timing tools is proposed.**

I. Introduction

When first introduced, negative limits resulted from combining pre-characterized cells into a larger one. NTC negative limits and the associated delays were physically correlated to the actual delays on the path from the input ports to the internal sequential element within the enclosing ASIC cell [1]. They actually represented the cumulative wiring and cell delays on the path. On the simulation side, a simple algorithm enabled the determination of a single set of input signal delay values when needed. The delayed signals were subsequently created automatically by the simulator.

The relentless push for higher design performance and associated reduced timing margins coupled with the timing closure pressure resulted in an increase in the number and complexity of NTC. Nowadays, The NTC limit values are calculated by pure mathematical table interpolation by the static timing tool and are not directly correlated to the cell physical delays. Consequently, the simple explanation of the previous paragraph and the corresponding convergence procedure fail severely for newer technologies. Furthermore, it is becoming impossible to find a single set of delays that satisfies the ever increasing number of NTC with different conditions. Indeed various input signal conditions are used to relate and combine timing checks that are active for different regions of a cell operation. Though each set of timing checks associated with the same condition, can typically be solved, finding a common convergence solution for all conditions is becoming practically impossible.

In this paper, we first provide a brief overview of the evolution of NTC delay characteristics as evidenced by the delay model features from its initial introduction to the current time. After presenting various drawbacks and pitfalls that may result from an invalid or inaccurate determination of NTC delays, we describe our solution procedure and algorithm which is based on linear programming and the Simplex solution procedure [2, 3]. We also describe the conditional delay construct that is needed to accurately model the NTC delayed net behavior together with the associated assumptions that need to hold for correct operation. In Section VI, we present examples and results that illustrate the applicability of the algorithm to nano-scale designs. Finally we conclude by outlining a novel approach and methodology to generate NTC delays directly by the static timing tools as part of the SDF annotation process.

II NTC DELAY CHARACTERISTICS EVOLUTION

Early NTC models required a single delay value for each input signal. The delay value represented the same rising and falling input signal transition. The single value was used to adjust the limits of the corresponding timing checks so as to consistently reflect the original condition between the input signal transitions.

As timing accuracy requirements increased, more edge sensitive NTC were used and single delay values were not sufficient to achieve convergence of all NTC i.e. a single delay value could not be found that will ensure the proper sequence for both rising and falling edge NTC. To resolve the situation the use of two-valued delay was introduced in the solution. Delayed nets could then have different rising and falling delays. This added extra complexity to the solution and introduced the potential for glitch filtering and suppression.

With the ever increasing complexity and requirement of nanometer technology devices, further timing accuracy improvement was needed and the use of conditional NTC increased substantially. Indeed the usage of specific sets of timing checks for different regions of cell operation became very common. Furthermore, the NTC limits were now calculated by pure mathematical procedures such as table interpolation without any regard to physical reality. Consequently finding a common set of delayed signals to satisfy simultaneously all NTC inequalities was frequently impossible and further refinement of the NTC solution was needed. In the following section we will expand on this as well as other drawbacks of inaccurate NTC delays.

III DRAWBACKS OF INACCURATE NTC DELAYS

The widening of a timing check violation window is the most common drawback associated with the NTC non-convergence. When convergence cannot be achieved, the simulators typically issue a non-convergence warning during elaboration and sets the corresponding NTC limits to zero. As a result, one of the two timing checks will be disabled; In addition the timing check violation window becomes wider resulting in more simulation pessimism that could lead to false timing violations. This is illustrated in Figure 1 for a negative hold time in a $setuphold() timing check.

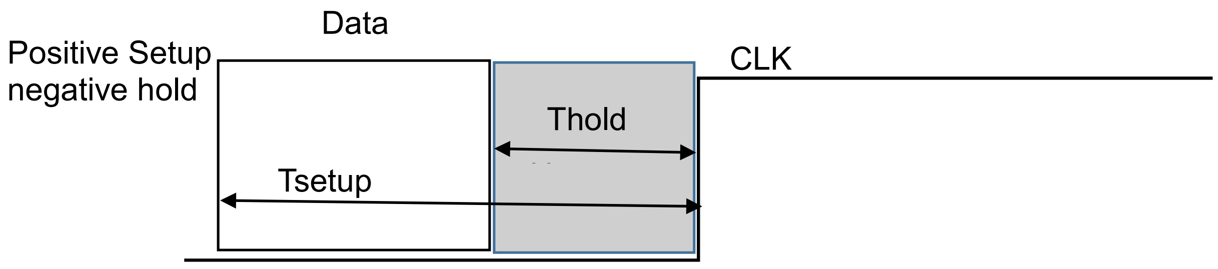


Figure 1: Timing check violation window changes when NTC limits are set to 0.

In Figure 1, the shaded area represent the additional violation window that results from setting the hold time to zero. In addition to the widening of the timing check violation window, other problems can be encountered and are more subtle to detect. The cause behind the generation of invalid delays is the non-uniqueness of the NTC solution which is the result of the mathematical procedure used by the STA tool to generate the limits coupled with the NTC convergence solution algorithm used to calculate the input signal internal delays from the timing check limits. Indeed the constraint imposed from the timing check limit is on the difference between the input signal transition times not the actual transition time. As a result, the following are common problems that could occur due to inaccurate NTC delays:

- Glitch suppression: When different rising and falling delays are calculated, the potential for glitch filtering exist. As a result, it is recommended to minimize the difference between the rising and falling delays of the delayed signals.

- Delay filtering: Sometimes the calculated NTC internal delays can be larger than the enclosing cell input-output path delays. This is an unexpected case and can lead to erroneous simulation results.

- Functional verification errors such as the capture of data at the wrong clock edge could also result from invalid NTC delay values. A timing check limit enforces a difference between two transitions, so satisfying the constraint with a large NTC delay on a clock input signal could cause this functional error.

In the following section we present a procedure for the determination of the NTC delay values based on linear programming and the simplex solution. The procedure is adapted in such a way as to avoid the above drawbacks.

IV SIMPLEX PROCEDURE TO DETERMINE NTC DELAYS

The transformation of the NTC delay convergence problem into a linear programming problem is presented. The transformation enables a solution based on the Simplex method which bypasses existing iterative solution that is heuristic and local by nature.

A linear programming problem can be formulated in the following normal form:

maximize $\mathbf{C}^T\mathbf{X}$
subject to $\mathbf{AX} \leq \mathbf{B}$ **and** $\mathbf{X} \geq \mathbf{0}$

where **X** is a vector of n independent positive variables, **A** is an mxn linear matrix representing m linear constraints between the independent variables.

To provide an insight into the transformation, we recall that timing checks capture the conditions that the input pin value change must satisfy for correct operation. Figure 2 illustrates the two signal of a timing checks with their corresponding internal NTC delays DEL_1, and DEL_2. D and CLK are the cell input signals. DEL_1 and DEL_2 are the NTC delays that need to be inserted to guarantee proper sequence of the data and reference signals. Assume the following timing check is present:

$setuphold(posedge CLK, posedge D, Tsetup, Thold);

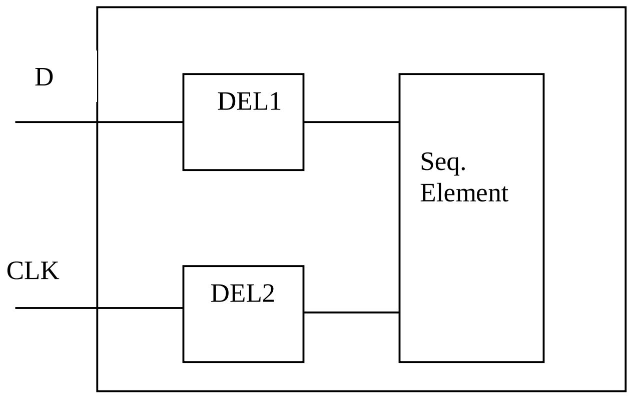


Figure 2: NTC internal delay illustration

The $setuphold timing check is equivalent to two timing checks: a $setup() and a $hold(). Note that only one of the limits can be negative and without loss of generality we will assume that the hold limit is negative in the following. The values of the input signal delays DEL_1 and DEL_3 have to satisfy the following constraints:

- The negative hold limit needs to become positive:
  $DEL_1 - DEL_2 > -Thold$

- The positive setup limit should remain positive:
  $DEL_2 – DEL_1 < Tsetup$

The above inequalities are linear constraints that need to hold. By similarly transforming all the timing check the matrix A from above can be constructed.

In addition to the linear constraints, we also define the objective function to maximize as the following:
Objective = $-\sum DELi$

Maximizing the above objective results in minimizing all the individual input signal delays thus avoiding the delay filtering drawback from Section III.

The linear programming system resulting from the objective function together with all the inequality constraints corresponding to all NTC in a cell can be solved using the Simplex method. We use a two phase solution that first determine a feasible solution and then maximizes the objective function while maintaining feasibility.

We note that for when presented with a non-feasible problem the Simplex procedure provides a straightforward method to adjust the timing check limits to achieve convergence. Substituting the non-feasible solution in the matrix A, the limit of the invalid constraints can be adjusted individually to satisfy the inequalities. The limits will thus be adjusted so as to achieve convergence without undue pessimism.

A. Signals with different rising and falling delays

When different edge sensitive transitions are present in the timing checks we extend the transformation described in the previous section to use different Linear Programming (LP) variables for the rising and falling internal delays of NTC input signals. In this case, the LP Simplex solution could potentially calculate different rise and fall delays which could result in simulation glitches filtering as described in Section III. In order to minimize the difference between rising and falling delays, and thus the potential for simulation glitches, we perform another iteration after regenerating the inequality constraints for signals with different rise and fall delays using the following additional transformation:

```
if (rise > fall) then rise = fall + diff
if (fall  > rise) then fall = rise + diff
```

In the objective function, the appropriate variable is replaced by the corresponding difference value (diff). The LP solution will then result in a minimum difference between the rise and fall delay.

B. Signals with conditional delays

A similar procedure can be applied to get input signal NTC delays that depend on timing check conditions. In this case an LP variable will represent the delay of a signal in combination with a timing check condition. And the resulting linear programming system will also be solved by the same Simplex method. So an LP variable can represent either of the following:

- An input signal delay
- An input signal edge sensitive delay i.e. rise/fall delay.
- An input signal edge sensitive delay combined with a timing check condition i.e. a rise/fall delay when a certain condition is true.

In all cases the resulting linear programming problem solution is the same, namely the implex algorithm. Note that maximizing the objective function, will always ensure that any delay that is not needed to ensure feasibility will be set to zero. When conditional delays are used, the NTC delayed signal can have different delays depending on the timing check conditions truth value. This is described in details in the following section.

V CONDITIONAL DELAY CONSTRUCT

In the presence of conditional delays, the NTC signal delay values will depend on the truth value of the timing check conditions. If Di_int is the internal delayed signal corresponding to the input signal Di then the delay behavior can be captured by the pseudo code listing in Figure 3.

```
if ($cond_1$)  ($D1 => D1_{int}$) = ($rise_1$, $fall_1$);
if ($cond_2$)  ($D2 => D2_{int}$) = ($rise_2$, $fall_2$);
.
.
.
if ($cond_n$)  ($Dn => Dn_{int}$) = ($rise_n$, $fall_n$)
```

Figure 3: Verilog specify block code listing equivalent to conditional delay construct.

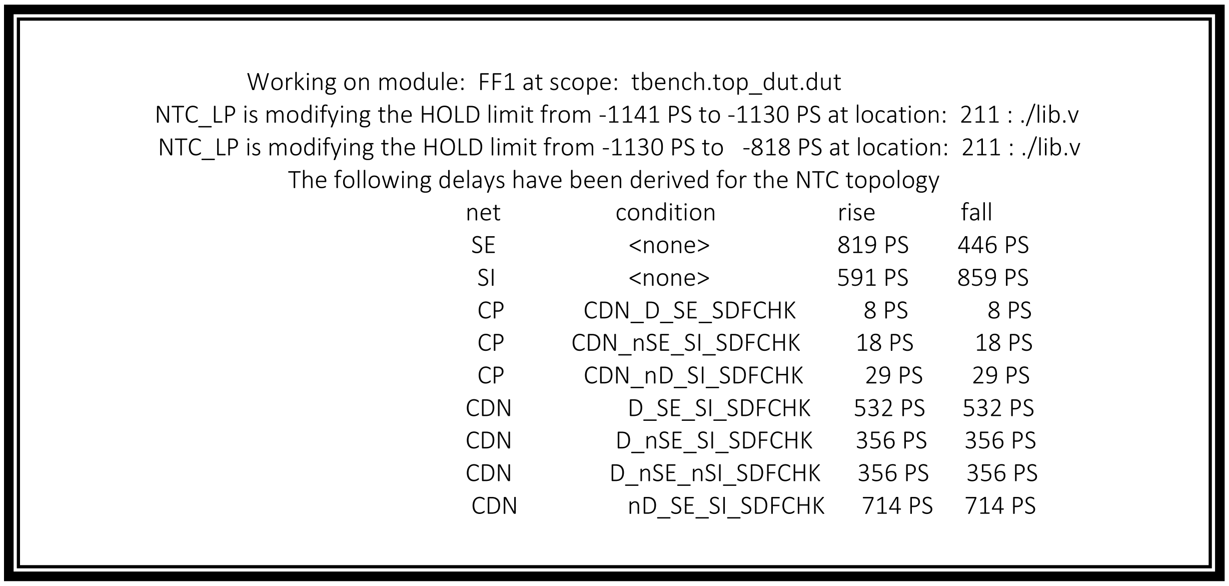
Note that in the pseudo code listing in Figure 3, the rise and fall delay values can be the same which corresponds to the case of a single delay value. Additionally, when none of the conditions is true, the delay will default to zero.

An inherent assumption in the NTC linear programming formulation and solution is that the timing check conditions are mutually exclusive. This is typically the case. During simulation however invalid input bias conditions can result in more than one condition being true simultaneously. In this case a warning is issued and the delay selected will be the more pessimistic delay. We note that the pessimistic delay selection could be either the minimum or maximum active delay. The selection will depend on the corresponding timing check type (e.g. is it a setup or a hold) and how the signal is connected to the internal sequential element (i.e. is it a reference or a data signal).

VI EXAMPLES AND RESULTS

We have implemented the linear programming Simplex solution for NTC and the new NTC conditional delay construct in the latest IUS release. The new solution consistently converges in all cases where the old algorithm failed to converge. In addition and due to the layered abstraction levels of the LP variables, the results match the old algorithm results in all cases where both converge. When limit adjustments was needed, the new method adjusted limits were always less than the existing coarser adjustments.

Figure 4 provides an example of the results in case of a cell where conditional delays were needed.



```
              Working on module:  FF1 at scope:  tbench.top_dut.dut
    NTC_LP is modifying the HOLD limit from -1141 PS to -1130 PS at location:  211 : ./lib.v
    NTC_LP is modifying the HOLD limit from -1130 PS to   -818 PS at location:  211 : ./lib.v
                The following delays have been derived for the NTC topology
                               net           condition              rise        fall
                               SE              <none>               819 PS      446 PS
                               SI              <none>               591 PS      859 PS
                               CP          CDN_D_SE_SDFCHK            8 PS        8 PS
                               CP         CDN_nSE_SI_SDFCHK          18 PS       18 PS
                               CP          CDN_nD_SI_SDFCHK           29 PS       29 PS
                               CDN            D_SE_SI_SDFCHK        532 PS      532 PS
                               CDN           D_nSE_SI_SDFCHK        356 PS      356 PS
                               CDN          D_nSE_nSI_SDFCHK        356 PS      356 PS
                               CDN            nD_SE_SI_SDFCHK       714 PS      714 PS
```

Figure 4: Example results showing calculated NTC delays.

VII A NEW METHODOLOGY PROPOSAL

The current NTC methodology is the result of historical conditions that are irrelevant to the current technology reality. As stated earlier, combining pre-characterized ASIC cells into a new cell was probably the reason for the initial NTC usage. The advent of VLSI technology caused this practice to disappear. In this section we first highlight the main limitation of the current NTC methodology and propose a new approach for generating the NTC delays directly by the Static Timing Analysis (STA) tool.

In general, the timing check limits are generated by means of mathematical table interpolation by STA tools. The interpolation is typically done with respect to the cell input transition time and output load. The NTC negative limits are then manipulated by the simulator to recover the presumed cell internal delays by various heuristics and/or iterative algorithm or more formally using the linear programming approach and the Simplex solution as advocated in this paper. The two step process loses any connection to the physical reality of the actual cell delays and can obfuscate and complicate the problem with no in-built benefit. Furthermore it is prone to the risk of generating invalid delays that could result in simulation errors. The process is indeed superfluous: First hide the input signal delays by capturing their effect as negative NTC limits; then during simulation try to recover the same delays.

We propose a new methodology and flow whereby the STA tool will generate the negative delays directly instead of via the negative NTC limits. The inherent assumption is that a delayed net can potentially exists for all cell input signals. The simulator will automatically generate a delayed net corresponding to each input signal where an NTC delay was specified. Note that if no internal delay annotation is given then the delay is assumed to be zero i.e. the input signal and corresponding delayed net are shorted. This will require the following modifications to the current tool flow:

- SDF: For cell instances with NTC, SDF should support the annotation of internal delays associated with each cell input signal. This might require the addition of a new syntax for SDF delays such as the following:
  (NTC (D Dint) COND (rise fall))
  We note that the signals in the SDF annotations have to be consistent with the cell HDL especially the naming of the delayed signals needs to be consistent.

- STA: While generating the timing check limit annotation, the STA tool will calculate the limits on the internal sequential element (See Figure 2) instead of the cell input signals. The STA tool will assume the presence of internal delays associated with each cell input signal and will calculate the appropriate delay values. Note that in this case all the limit annotations will be positive.

- On the simulator side, there will be no need to calculate any NTC delays to recover the proper signal sequence and no corresponding NTC limit adjustments.

We are currently investigating this approach with the Cadence timing team and hope to report on the results of a prototype implementation at the conference. Aside from assessing the basic feasibility of the proposed approach, we would like to evaluate the following:

- The changes needed to generate the internal delays directly by the STA tool and whether a separate parameter extraction step is required.

- Whether the new conditional delay construct is needed to model internal delays.

- The soundness of the STA calculated delay values. We expect the delays to better reflect the physical reality of the cell. This will provide a mean to automatically adjust the delay and limits to better reflect manufacturing variances.

VIII CONCLUSION

In this paper, we have presented a new approach based on linear programming and the Simplex algorithm to solve the NTC convergence problem in event driven simulation. We also introduced the conditional delay construct and highlighted the associated assumption inherent in its use. Finally, we propose a new methodology and tool flow that will bypass the need for the work described in this paper and results in simulation performance improvement as well as a more seamless tool flow.

ACKNOWLEDGMENT

We would like to acknowledge the contribution of the following colleagues to the work presented in this paper: Manoj Kumar, Ashish Narang, and Gagandeep Singh.

REFERENCES

[1] "1364-2001 IEEE Standard Verilog Hardware Description Language", IEEE, Pascataway, New Jersey. Copyright 2001. ISBN: 0-7381-2826-0.
[2] Donald A. Pierre, *"Optimization Theory and Applications*", Dover Publications Inc, New York.
[3] William H. Press et al., "*Numerical Recipes in C, The Art of Scientific Computing*", Second Edition, Cambridge University Press.