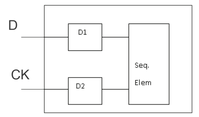
Conditional Delays for Negative Limit Timing Checks in Event Driven Simulation

N. Kalil and D. Roberts, Cadence Design Systems

NTC….Where do they come from?

- Originated from combining multiple cells e.g. Scan flops.
- D1 and D2 are internal delays: logic and/or wires.
- Pre characterized sequential element timing check is the same but input pins and associated delays are different.
- What if D1 > D2 (or D2 > D1) and (D2 – D1) or (D1 – D2) > limit?
- Nowadays NTC has no physical basis.
- Limit values are generated by mathematical interpolation by STA tool.



Problem Statement

- Timing checks with negative limits need to be transformed to restore normal sequence of data and reference events (simulator cannot predict the future!)
- NTC algorithm: Find the input signal delays that needs to be inserted so that all NTC negative limits are transformed into positive limits.
- Non-convergence issues with newer technology cells due to timing checks conditions.
- Typically conditions on input signals are:

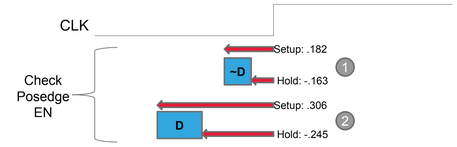
  - Mutually exclusive
  - Active in different regions of cell operation.

- Example checks that can cause problems:

```
(SETUPHOLD (posedge EN)   (COND  D  (posedge CK)) (0.306::0.308) (-0.245::-0.243))
(SETUPHOLD (posedge EN)   (COND ~D  (posedge CK)) (0.182::0.184) (-0.163::-0.162))
```


Problem Example Illustration

(Not to scale)



1. SETUPHOLD (posedge EN) (COND ~D (posedge CK)) (0.182::0.184) (-0.163::-0.162))
2. SETUPHOLD (posedge EN) (COND D (posedge CK)) (0.306::0.308) (-0.245::-0.243))

- Conditions D and ~D are mutually exclusive
- One single delay cannot be found to transform all limits to positive values
- Conditional delays are needed!

NTC Convergence Using Linear Programming

- Formulate NTC convergence as an LP maximization:

  maximize $C^T X$

  subject to $AX \leq B$ and $X \geq 0$

- Every 2 limits timing check is equivalent to two linear constraints:

  D2 - D1 < limit1

  D1 – D2 > -limit2 (limit2 is the negative limit)

- The objective function is: $\sum_{k=0}^{n} -Dk$
- Objective is equivalent to minimizing the inserted delays.
- Solve by Simplex method.

NTC Convergence Using Linear Programming (Continued)

- The objective function is extended with a penalty function that weighs the difference between the rising and falling delays of the same net : Reducing the penalty is equivalent to minimizing the difference.
- Conditions can be added by a simple extension: An LP variable (object) can thus be:

  - Signal e.g. CLK
  - Signal with an edge e.g. (posedge CLK)
  - Signal with an edge and an associated condition e.g. (posedge CLK &&& cond4).

- Simplex solution determines all LP variables.
- A single input delayed net can potentially have different delays depending on conditions.

Conditional Delay Construct

- For input signal D:

```
if (cond1) delay = D + d1
if (cond2) delay = D + d2
…
if (condn) delay = D + dn
ifnone     delay = D + d0
```

- d0, d1…dn are determined by the LP solution.
- Delays with zero value result in a wire short.
- Inherent assumption: Conditions are mutually exclusive.
- If multiple conditions are simultaneously true: Warn and select most pessimistic delay.
- Pessimistic delay depends on timing check type and signal role in check (data or reference).

Simplex Solution

- Two phases Simplex algorithm:

  - Phase 1: Feasibility
  - Phase 2: Optimization.

- Resulting linear system is sparse: Used sparse compressed storage matrix representation.
- Adapted algorithm and search to specific problem at hand: Specialized solution techniques e.g. limit modification.
- Resultant solution is fast and space efficient.
- New solution is integrated into IUS.
- Tested and recently released for customer usage

Sample Results

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

Present Methodology

- STA tool generates SDF annotation values for timing check limits.
- Annotation values are calculated using mathematical procedure e.g. table interpolation, with no physical basis.
- Simulator, by an inverse procedure, tries to recover the input signal delay values that will result in the given timing check limits.
- Game of *hide and seek* between the STA tool and the simulator
- Should we continue the game?

New Proposal

- What if the STA tool is used to generate the internal delays directly?
- Delays will better reflect physical reality inside the cell.
- Timing check limit values will be adjusted consistently with the NTC delay values
- NO more negative limits!
- New SDF construct is needed to annotate internal delays e.g.:

  (NTC (D Dint) COND (riseDelay fallDelay))

- Simulator performance improvement: No need for NTC convergence and limit modifications.

Conclusion

- An innovative solution to the NTC convergence problem is presented.
- The solution is based on linear programming and the Simplex solution algorithm.
- A novel conditional delay construct was described together with its inherent usage assumptions.
- Sample results were given.
- A new methodology was proposed that will generate the input signal internal delays directly by the STA tool.
- The new methodology will bypass the need for the NTC convergence solution in this paper.