TAU'2013: Variation Aware Timing Analysis Contest

v1.0 - October 12, 2012

1 Introduction

This document describes the timing models and file formats that will be used in the TAU'2013 Variation Aware Timing Analysis Contest. The latest news, contacts and other information should be obtained from the contest web page at the following site.

`https://sites.google.com/site/taucontest2013/`

2 Timing Analysis

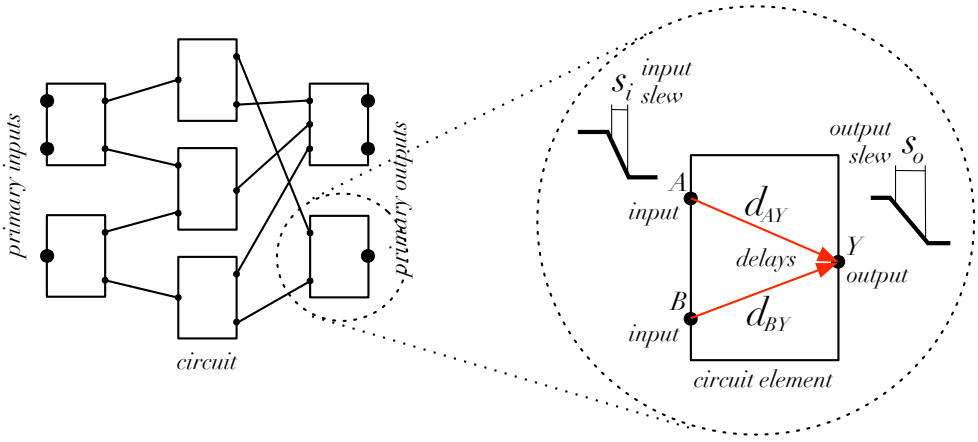

Figure 1: Circuit and circuit element characterization.

Timing analysis, in the context of Electronic Design Automation (EDA), amounts to computing timing information as signal transitions propagate from the inputs to the outputs of a digital circuit, usually described by a netlist of circuit elements. This is achieved by estimating signal propagation through the circuit elements, from the netlist inputs to outputs. Signal transitions arriving at the input of an element will be available at its outputs some time later. Each element therefore introduces a delay on signal transition propagation. Furthermore, we will assume that signal transitions are characterized by a *slew*. Circuit elements affect the signal transitions at their inputs by modifying their slew when shown at the outputs. The general model is illustrated in Figure 1, where delay is designated by d, input slew by s_i and output slew by s_o.

Arrival times, that we will designate by at, quantify the earliest or the latest time instant that a signal transition can reach the corresponding circuit node, when traveling from a circuit input. The meaning of the arrival time values depends on whether we assume the early or the late mode of operation. In early mode, we are concerned with computing the earliest time instant that a signal transition can reach any given circuit node. Conversely, in late mode we are concerned with computing the latest time instant that a signal transition can reach any given circuit node.

Therefore, arrival times are computed by adding edge delays across a path and computing the min or max (assuming either early or late mode) of such delays when they converge at a given circuit node. For example, assuming at_A^{early} and at_B^{early} to be the early arrival times at pins A and B of the circuit element represented in Figure 1, then the early arrival time at the output pin Y will be

$$at_Y^{early} = \min(\ at_A^{early} + d_{AY},\ at_B^{early} + d_{BY}\ ). \tag{1}$$

Conversely, the late arrival time at the output pin Y will be

$$at_Y^{late} = \max(\ at_A^{late} + d_{AY},\ at_B^{late} + d_{BY}\ ). \tag{2}$$

Required arrival times, that we will designate by rat, are limits imposed on the arrival times, in particular nodes of the circuit. Such limits are usually necessary to ensure proper circuit operation. Assuming either early or late mode, when a required arrival time is defined for a particular circuit node, the following conditions must hold.

$$at^{early} \geq rat^{early} \tag{3}$$

$$at^{late} \leq rat^{late} \tag{4}$$

Slacks, that we will designate by $slack$, are the difference between arrival times and required arrival times, and measure how well the constraints of (3) and (4) are met.

$$slack^{early} = at^{early} - rat^{early} \tag{5}$$

$$slack^{late} = rat^{late} - at^{late} \tag{6}$$

Slacks are positive when the required arrival time constraints are met, and negative otherwise.

Slew propagation is also an essential task of timing analysis, since cell and interconnect delays are a function of the input slew. We will assume worst-slew propagation, meaning that we propagate either the smallest or the largest slew, when we consider either early or late mode, respectively.

$$s_{oY}^{early} = \min(\ s_{oAY}^{early}(\ s_{iA}^{early}\ ),\ s_{oBY}^{early}(\ s_{iB}^{early}\ )\ ) \tag{7}$$

$$s_{oY}^{late} = \max(\ s_{oAY}^{late}(\ s_{iA}^{late}\ ),\ s_{oBY}^{late}(\ s_{iB}^{late}\ )\ ) \tag{8}$$

Slew propagation is irrespective of delay propagation: for the example circuit element of Figure 1, we can propagate the delay from input A and propagate the slew from input B.

2.1 Variation Aware Timing Analysis

With semiconductor technology scaling to sub 65 nano-meters, an increased significance of variability in a VLSI chip manufacturing process necessitates variation aware timing analysis. Manufacturing sources of variability include device front-end variability (e.g. variations in channel length, oxide thickness, dopant concentration, etc.) and back-end-of-line variability (metal variability). In addition, environmental sources of variation like voltage and temperature strongly impact circuit timing. Variability may be classified into different categories like intra-chip variability and inter-chip variability. Each of these may further be sub-classified as systematic variability and random variability. Sources of variation that impact circuit timing are termed *parameters*.

Variation aware timing analysis can be performed in different ways. One approach is to perform multiple deterministic timing runs at various corners in the parameter space, wherein a corner indicates a condition for each parameter (e.g. at some corner C_i, parameter voltage is at 1.05 volts that denotes a 3 sigma corner for parameter voltage, temperature is at 50 degrees Celsius that denotes a 2 sigma corner for parameter temperature, parameter metal-layer 4 is at the thick condition that denotes a -3 sigma corner for parameter metal-layer 4, ...). This approach requires a large number of runs to cover the parameter space. In some cases, more than two hundred corners need to be analyzed [5]. As a result, multi-threading or parallelization techniques are often employed for multi-corner timing. An alternate approach is to perform a single statistical timing analysis, wherein each timing quantity (e.g. delay, slew, arrival time) is denoted using a random variable with a known distribution to represent uncertainty due to variability. While the primary advantage of this approach is the ability to cover the parametric variation space in a single run, analytic linear modeling assumptions may lead to some inaccuracies in the final result. Monte Carlo based statistical timing analysis overcomes the accuracy problem, but is too run-time expensive for modern large designs. Several approaches to variation aware timing analysis have been proposed. A broad overview of this topic is presented in [1].

2.2 Parametric Timing Analysis Using a Linear Gaussian Model

A linear parametric timing model will be used for timing analysis in this contest. Similar to the work in [11], timing quantities are represented by a linear function of the parameters. In addition, all parameters are assumed to be independent. Each parameter is modeled as a random Gaussian variable. Any timing quantity (e.g. delay) is thus a weighted sum of Gaussians, and expressed as

$$a_0 + \sum_{i=1}^{n} a_i \Delta X_i + a_{n+1} \Delta R_a, \tag{9}$$

where, a_0 is the mean or nominal value of the delay, ΔX_i ($\forall i = 1, 2, \ldots, n$) represents the variation of n global parameters (X_i, $\forall i = 1, 2, \ldots, n$) from their nominal values, and a_i ($\forall i = 1, 2, \ldots, n$) are the per sigma sensitivities to their corresponding sources of variation. ΔR_a denotes the variation from the nominal of an independent random variable R_a associated with each circuit element (gate or wire). a_{n+1} represents the per sigma sensitivity to ΔR_a.

Variation aware parametric timing analysis involves propagating timing information across the parameter space through the circuit. This could be done using multiple techniques including multiple corner analysis, Monte Carlo analysis, or analytical statistical timing analysis. **The contest does not require adoption of any particular technique; and novel techniques are encouraged**. The results obtained via Monte Carlo based timing analysis will be used as the golden reference; however given that performance is one of the most important aspects of the contest, this technique is not recommended for use by contestants.

For analytical statistical timing analysis using timing quantities as a first order sum of random variables, addition and subtraction operations are performed easily. All terms are linearly added, while the random terms are root-sum-squared [11]. Maximum and minimum operations, on the other hand are complicated, and involve approximations. Clark's method [3] to obtain the moments of the maximum of two Gaussian distributions may be applied and a Gaussian with these two moments may be constructed to denote the approximate result of the maximum operation. Relevant mathematical details have been presented in [11, 2, 3]. For multiple Gaussians, the final result is obtained using iterative *pair-wise* maximum operations. It should be noted that approximation errors in the final

result depend on the order of the pair-wise operations, and techniques may be adopted to reduce error in the ordering process [10]. The same idea applies to statistical minimum operations as well. While performing maximum/minimum operations using Monte Carlo simulations may be most accurate, they are run-time expensive and should be used cautiously. The contest does not advocate any particular approach for these operations.

2.3 Sources of Variability and Sensitivity

Each timing quantity may be sensitive to the following global inter-chip sources of variability.

- environmental: **voltage (V)**, **temperature (T)**
- front end of line process: **channel length (L)**, **device width (W)**, **voltage threshold (H)**
- back end of line: **metal (M)**

For simplicity, only a single parameter M is assumed for all metal layers in this contest. These parameters denote systematic chip-to-chip (or inter-chip) sources of variation. As an example, variations in parameter temperature imply that the chips would be subject to different ambient temperature conditions. This parametric variation does not imply within chip (or intra-chip) systematic temperature gradients or differences. For simplicity in this contest, systematic intra-chip variability from all parameters are ignored. In addition to the above global inter-chip parameters, each timing quantity may contain an **independent random source of variability (R)** that denotes random inter-chip as well as random intra-chip variation. Any timing quantity may thus be denoted in the following notation, where μ denotes the nominal value of the quantity (value in absence of variability).

$$\mu + a_v\Delta V + a_t\Delta T + a_l\Delta L + a_w\Delta W + a_h\Delta H + a_m\Delta M + a_r\Delta R$$

Each parameter may vary between -3 to $+3$ sigmas. Parameter sensitivities are denoted as time units per sigma values (e.g. $a_v = 5$ pico seconds per sigma), and are obtained either as *asserted* values, or via *finite differencing*. In the former case, the sensitivity of a timing quantity to a parameter is available directly as an input (e.g. voltage sensitivity for cells are available as asserted values in the cell library). Finite differencing in context of any parameter X implies that the value of the timing quantity Q is available (or can be computed) for at least two (sigma) corners of X. The first order sensitivity is then computed as the ratio of the change in Q between the two corners of X and the difference in the two corners of X. Assuming two sigma corners of X as $+3$ and -3 sigma, the finite differenced sensitivity a_X is computed as

$$a_X = \frac{Q_{|X=+3\sigma} - Q_{|X=-3\sigma}}{3-(-3)}.$$

If Q is truly a linear function of X, the choice of sigma corners does not matter during finite differencing. For models where Q is not a linear function of X, finite differenced sensitivity calculation is ideally sigma corner dependent but allows the creation of an approximate linear model. In this contest, all parameters except metal (M) should be finite differenced (if required) between $+3$ and -3 sigma values. The metal parameter should be finite differenced between $+3$ and 0 sigma values (aids faster analysis, details in following section). All timing quantities mentioned in this section should be assumed to be a function of variational parameters.

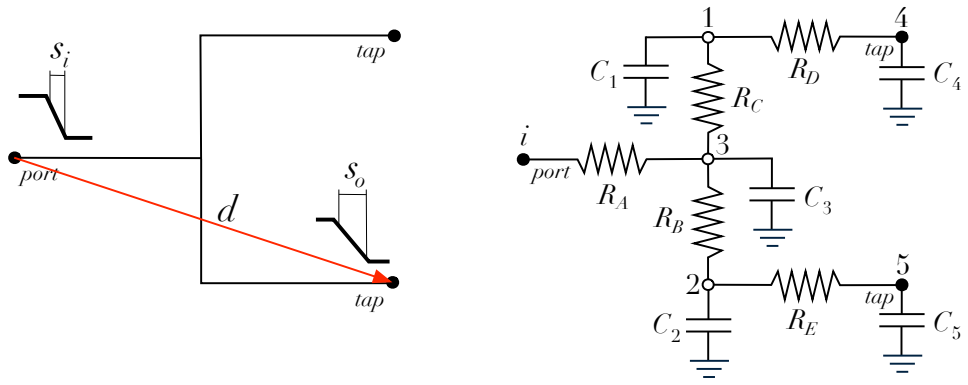

Figure 2: Interconnect characterization.

3 Models

For the purpose of TAU'2013 Timing Analysis Contest, we will assume that for each benchmark circuit two files are available: a netlist file and a library file. The netlist file contains circuit information, topology and other circuit related data, that will be modeled as discussed in Section 3. The netlist is composed of a set of interconnected elements, namely cell instances and interconnecting circuitry. The library file contains timing information regarding the available cell elements as well as variability information. The syntax of such files will be described in Section 6. The models to be encountered during timing analysis, are of two types: interconnect and cell circuit elements.

3.1 Interconnect

The basic instance of interconnect (wire) is a *net*, which is assumed to have an input pin, designated by *port*, and one or many output pins, designated by *taps*, as illustrated in Figure 2 (left). For each net, the netlist of its parasitic RC tree is provided in the netlist file. An example of a parasitic RC tree is presented in Figure 2 (right). Parasitic RC trees only contain grounded capacitors and resistors between nodes in the tree (there are no grounded resistors or coupling capacitors).

The computation of port-to-tap delays can be accurately performed through electrical simulation. However, and for the sake of simplicity, we will assume a simpler delay model, namely the Elmore delay model [4], whereby the delay is approximated by the value of the first moment of the impulse response. For RC tree networks, [7] provides a simple topological method for computing such value, that we summarize here.

Consider any two given nodes e and k, where the lumped capacitance in node k is known to be C_k. The resistance R_{ke} is the resistance of the common subpath between the paths from the port to k and e, respectively. Moreover, R_{ee} is the resistance between the port and node e. For the example net illustrated in Figure 2 (right), we have $R_{15} = R_A$, since the common subpath between nodes 1 and 5 only comprises resistor R_A. The Elmore delay, for a given node e, is given by the sum,

$$d_e = \sum_k R_{ke}\, C_k, \tag{10}$$

where the summation extends over all nodes in the network. This value can be easily computed by an appropriate traversal of the netlist of the parasitic RC tree. For the example net illustrated in

Figure 2 (right), we have,

$$d_5 = R_A(C_1 + C_3 + C_4) + (R_A + R_B)\, C_2 + (R_A + R_B + R_E)\, C_5. \tag{11}$$

This value provides the nominal or mean wire delay between the port and the tap. For variation aware parametric delay computation, finite differencing is performed to compute the sensitivity of delay to the metal parameter M. Provided corner specific metal resistance m_R^σ and metal capacitance m_C^σ scalar values (from the cell library) are used to obtain the updated resistance and capacitance values of the interconnect network when the metal parameter is set to a given corner ($\Delta M = \sigma$). Each interconnect resistance and capacitance is scaled by the provided scalar, and another deterministic delay computation is performed to compute the delay when ΔM is at the σ corner. It should be noted that the component of tap capacitance that comes from the cell pin capacitance should not be scaled.

For the example above, assuming that the tap capacitance C_5 comes partly from the cell pin capacitance $C_{p,5}$ connected at node 5, the remaining capacitance $(C_5 - C_{p,5})$ is part of the interconnect network, using similar notation for tap capacitance C_4, the delay at this corner is given by

$$\begin{aligned} d_{5|\Delta M=\sigma} \;=\; & m_R^\sigma R_A(m_C^\sigma[C_1 + C_3 + C_4 - C_{p,4}] + C_{p,4}) + (m_R^\sigma[R_A + R_B])\, m_C^\sigma C_2 + \\ & (m_R^\sigma[R_A + R_B + R_E])\, (C_{p,5} + m_C^\sigma(C_5 - C_{p,5})). \end{aligned} \tag{12}$$

The above computation is performed for the provided sigma corner of the metal parameter (typically +3 sigma). The sensitivity to parameter ΔM is now computed using finite differencing between this corner and the nominal corner as follows.

$$\alpha_{m,5}^D = \frac{d_{5|\Delta M=\sigma} - d_{5|\Delta M=0}}{\sigma - 0} = \frac{d_{5|\Delta M=\sigma} - d_5}{\sigma}. \tag{13}$$

The parametric delay model of the interconnect from the port to tap node 5 thus involves two deterministic delay calculations and is finally approximated to a linear model as

$$d_5 + \alpha_{m,5}^D \Delta M.$$

Wire delays will consequently not contain any sensitivity to other parameters (including random variation). The above parametric wire delay model as a function of the normal Gaussian parameter ΔM **will be assumed as the golden model** even for evaluation using a Monte Carlo simulator (that is, non-linearities will be ignored).

The value of the nominal output slew on any given tap node o can be approximately computed by a two-step procedure. First, one computes the nominal output slew of the impulse response on o, which was observed [4, 6] to be well approximated by the following expression,

$$\hat{s}_o \approx \sqrt{2\beta_o - d_o^2}, \tag{14}$$

where β_o is the second moment of the impulse response at node o, and d_o is the corresponding Elmore delay computed from (10), for node o. The value of β_o can be computed through the efficient path-tracing algorithm for moment computation proposed in [9], which is a generalization of the algorithm proposed in [7] and described earlier.

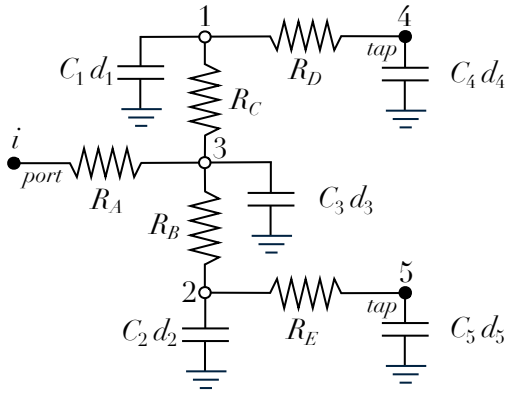

Figure 3: Modified RC tree for computing the second moment of impulse response.

For computing β_o, we start by replacing all capacitance values C_k by $C_k d_k$, where d_k is the Elmore delay computed from (10). Figure 3 illustrates the modified parasitic RC tree for the example of Figure 2. Next, we follow the same procedure as before for computing β_e as

$$\beta_e = \sum_k R_{ke}\, C_k\, d_k. \tag{15}$$

Therefore, for the example parasitic RC tree illustrated in Figure 3, we obtain,

$$\beta_5 = R_A\,(C_1 d_1 + C_3 d_3 + C_4 d_4) + (R_A + R_B)\,C_2 d_2 + (R_A + R_B + R_E)\,C_5 d_5. \tag{16}$$

After computing $\hat{s}_o$ from (14), we proceed to compute the mean slew of the response to the input ramp, s_o, for which a good approximation is proposed in [8], given by the simple expression,

$$s_o \approx \sqrt{s_i^2 + \hat{s}_o^2}, \tag{17}$$

where s_i is the nominal or mean input slew.

Parametric output slew calculation involves a bit of complex finite differencing. Under metal variability, $\hat{s}_o$ is a function of parameter ΔM only since both β_o and d_o are dependent on metal scalars. For a metal sigma corner σ,

$$\hat{s}_{o|\Delta M=\sigma} \approx \sqrt{2\beta_{o|\Delta M=\sigma} - d_{o|\Delta M=\sigma}^2}, \tag{18}$$

where, $d_{o|\Delta M=\sigma}$ is the Elmore delay value to the tap at node o at the given metal corner[1] and $\beta_{o|\Delta M=\sigma}$ may be computed by scaling the interconnect resistance and capacitance values similar to (12). The sensitivity to the metal parameter may now be computed via finite differencing the value of $\hat{s}_o$ between the two sigma corners for ΔM, as

$$\alpha_m^{\hat{S}_o} = \frac{\hat{s}_{o|\Delta M=\sigma} - \hat{s}_{o|\Delta M=0}}{\sigma - 0} = \frac{\hat{s}_{o|\Delta M=\sigma} - \hat{s}_o}{\sigma} \tag{19}$$

It follows that an additional deterministic computation for β_o at the metal corner σ is required to compute the parametric output slew $\hat{S}_o$ to an impulse response. We thus have

$$\hat{S}_o = \hat{s}_o + \alpha_m^{\hat{S}_o} \Delta M. \tag{20}$$

[1] already calculated during parametric wire delay calculation in (13)

It should be noted that this **model will be treated as the golden model** during evaluation. Given parametric models for S_i (should be available from upstream calculations) and $\hat{S}_o$, it is not straightforward to compute a linear parametric output slew S_o given the non-linearity of (17). **Novel approaches to modeling the parametric output slew (possibly as a linear model) are encouraged**. For evaluation using a Monte Carlo simulator, samples for S_i and $\hat{S}_o$ would be applied to (17) to obtain a sample for the output slew.

A naive finite differencing based approach to model S_o in a linear parametric form is as follows (it is not required to follow this approach in the contest). It is assumed that S_i is available in the following linear parametric form

$$S_i = s_i + s_m \Delta M + \ldots + s_x \Delta X + \ldots,$$

where, X denotes any non metal parameter (e.g. $V, T, \ldots$). For any non-metal parameter (including random) variation ΔX, the sensitivity of output slew to ΔX is computed by perturbing that parameter between its extreme sigma corners and observing the change in the projected value of S_i when all other parameters are at their nominal conditions (0 sigma). Note that $\hat{S}_o$ is not a function of ΔX. Using finite differencing, the sensitivity of the parametric output slew to ΔX is thus given by

$$\begin{aligned} \alpha_x^S &= \frac{s_{o|\Delta X=+3} - s_{o|\Delta X=-3}}{+3-(-3)} = \frac{\sqrt{s_{i|\Delta X=+3}^2 + \hat{s}_o^2} - \sqrt{s_{i|\Delta X=-3}^2 + \hat{s}_o^2}}{6} \\ &= \frac{\sqrt{(s_i + 3s_x)^2 + \hat{s}_o^2} - \sqrt{(s_i - 3s_x)^2 + \hat{s}_o^2}}{6}. \end{aligned} \tag{21}$$

For computing the sensitivity to the metal parameter, both S_i and $\hat{S}_o$ need to be perturbed during finite differencing, since both are functions of ΔM. Mathematically,

$$\begin{aligned} \alpha_m^S &= \frac{s_{o|\Delta M=+3} - s_{o|\Delta M=0}}{+3-0} = \frac{\sqrt{s_{i|\Delta M=+3}^2 + \hat{s}_{o|\Delta M=+3}^2} - \sqrt{s_{i|\Delta M=0}^2 + \hat{s}_{o|\Delta M=0}^2}}{3} \\ &= \frac{\sqrt{(s_i + 3s_m)^2 + (\hat{s}_o + 3\alpha_m^{\hat{S}_o})^2} - \sqrt{s_i^2 + \hat{s}_o^2}}{3} = \frac{\sqrt{(s_i + 3s_m)^2 + (\hat{s}_o + 3\alpha_m^{\hat{S}_o})^2} - s_o^2}{3}. \end{aligned} \tag{22}$$

The output slew is thus expressed in the following parametric linear form.

$$S_o = s_o + \alpha_m^S \Delta M + \ldots + \alpha_x^S \Delta X + \ldots,$$

where, X denotes any non metal parameter (e.g. V, $T, \ldots$).

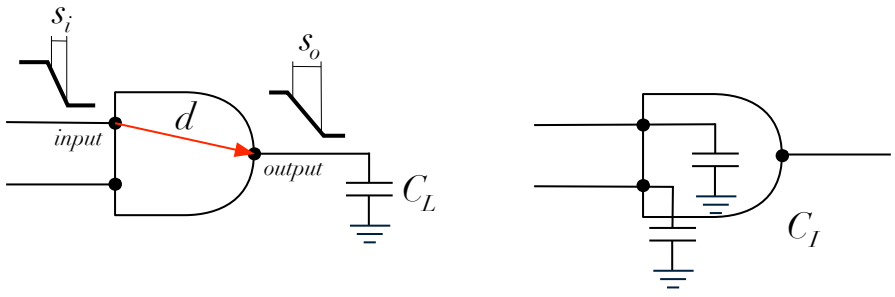

Figure 4: Combinational cell characterization.

3.2 Combinational Cells

We will assume here that cell delay, D, and output slew, S_o, can be approximated, for a given combinational cell input/output pin pair, by the following formulas.

$$D = a\,(1 + k_{d,v}\,\Delta V + k_{d,t}\,\Delta T + k_{d,l}\,\Delta L + k_{d,w}\,\Delta W + k_{d,h}\,\Delta H + k_{d,r}\,\Delta R) + b\,C_L + c\,S_i \quad (23)$$

$$S_o = x\,(1 + k_{s,v}\,\Delta V + k_{s,t}\,\Delta T + k_{s,l}\,\Delta L + k_{s,w}\,\Delta W + k_{s,h}\,\Delta H + k_{s,r}\,\Delta R) + y\,C_L + z\,S_i \quad (24)$$

where a, b, c, x, y, z and each k_i term are cell-dependent constants; C_L and S_i are the parametric output load and parametric input slew, respectively. The cell dependent constants are provided for each cell and for rise/fall transition, in the cell library file. Note that the k_i terms represent sensitivity to front end parameters as a fractional value of a or x. As an example, the sensitivity of delay to parameter ΔV would be given by $(a * k_{d,v} + c * s_v)$, where s_v denotes the sensitivity of input slew S_i to ΔV. However, the delay sensitivity to random variation ΔR would be given by $\sqrt{(a * k_{d,r})^2 + (c * s_r)^2}$.

C_L denotes the equivalent downstream capacitance seen from the output pin of the cell. Several sophisticated models have been proposed for computing C_L. For simplicity, the application of such models is considered to be out of the scope of the present contest, and a simple model is adopted. C_L is assumed to be the sum of all the capacitances in the parasitic RC tree including cell pin capacitances at the taps of the interconnect network.

$$C_L = \sum_k C_k \quad (25)$$

For the example net illustrated in Figure 2 (right), at the nominal metal corner, we trivially have

$$C_{L|\Delta M=0} = C_1 + C_2 + C_3 + C_4 + C_5. \quad (26)$$

Since the interconnect capacitances are dependent on ΔM, C_L is also a function of ΔM. In the above example, at a given sigma corner σ for ΔM (from the cell library),

$$C_{L|\Delta M=\sigma} = m_C^\sigma[C_1 + C_2 + C_3 + (1 - C_{p,4}) + (C_5 - C_{p,5})] + C_{p,4} + C_{p,5}, \quad (27)$$

where, $C_{p,4}$ and $C_{p,5}$ denote the cell pin capacitance contributions in the tap capacitances C_4 and C_5, respectively, and m_C^σ is the interconnect capacitance scalar value obtained from the cell library to denote impact of metal variation at corner σ. Cell input pin capacitances are provided as fixed values for each pin and for each rise/fall transition, in the cell library file.

The effective load's sensitivity to ΔM is computed via finite differencing as

$$l_m = \frac{C_{L|\Delta M=\sigma} - C_{L|\Delta M=0}}{\sigma - 0} = \frac{C_{L|\Delta M=\sigma} - C_{L|\Delta M=0}}{\sigma}. \quad (28)$$

The effective load C_L is then expressed in a linear parametric form as

$$C_{L|\Delta M=0} + l_m \Delta M,$$

and plugged into the cell delay and slew models in (23) and (24).

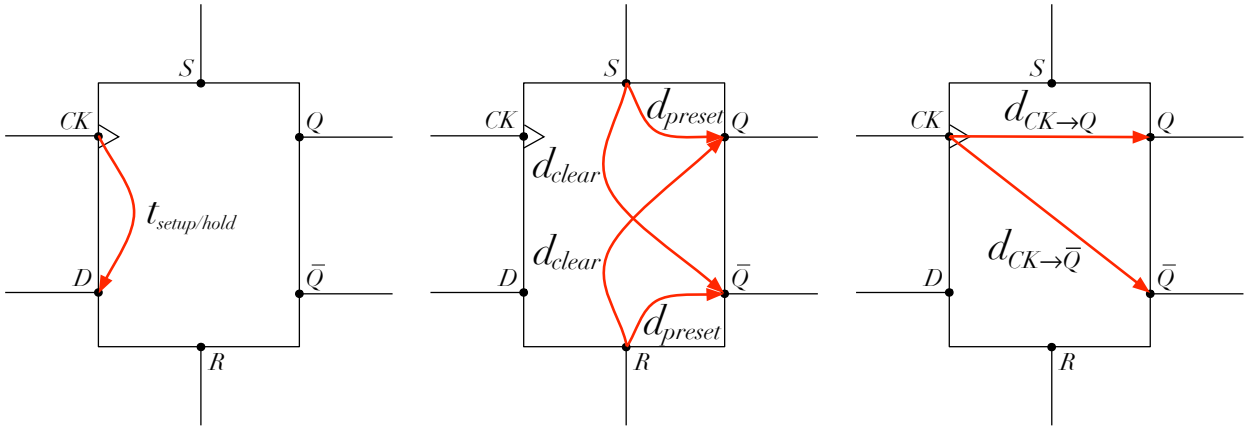

Figure 5: Flip-flop characterization.

3.3 Flip-Flops

Sequential circuits consist of combinational blocks interleaved by registers, usually implemented with flip-flops. Typically they are composed of several stages, where a register captures data from the outputs of a combinational block and injects it into the inputs of the combinational block in the next stage. Register operation is synchronized by clock signals generated by one or multiple clock sources. Clock signals that reach distinct flip-flops (sinks in the clock tree) are delayed from the clock source by a given *clock latency*, that we will designated by l.

A flip-flop (D flip-flop, more specifically) is a storage element that captures a given logic value at its input data pin D, when a given clock edge is detected at its clock pin CK, and subsequently presents the captured value and its complement at the output pins Q and $\overline{Q}$. The flip-flop also enables asynchronous preset (set) and clear (reset) of the output pins, through the S and R input pins.

Proper operation of a flip-flop requires the logic value of the input data pin to be stable for a specific period of time *before* the capturing clock edge. This period of time is designated by *setup time*, and we will represent it by t_{setup}. Additionally, the logic value of the input data pin must also be stable for a specific period of time *after* the capturing clock edge. This period of time is designated by *hold time*, and we will represent it by t_{hold}. Setup/hold times are one of the standard performance figures provided in cell specification libraries for storage elements. Other figures include delay from clock to output, $d_{CK\to Q}/d_{CK\to\overline{Q}}$ and asynchronous preset and clear delays, d_{preset} and d_{clear}. All flip-flop standard timing figures are illustrated in Figure 5.

Setup and hold constraints are modeled as functions of the input slews at both the clock pin, CK, and the data input pin, D, respectively, as follows.

$$t_{setup} = g + h\ S_i^{CK} + j\ S_i^D \tag{29}$$

$$t_{hold} = m + n\ S_i^{CK} + p\ S_i^D \tag{30}$$

In the above equations, the input slews are parametric, and consequently, the setup and hold times are parametric as well. Let us consider the usual case of signal propagation between two flip-flops, as illustrated in Figure 6. Assuming that the clock edge is generated in the clock source at time 0, then it will reach the injecting flip-flop at time l_i, making the data available at the input of the

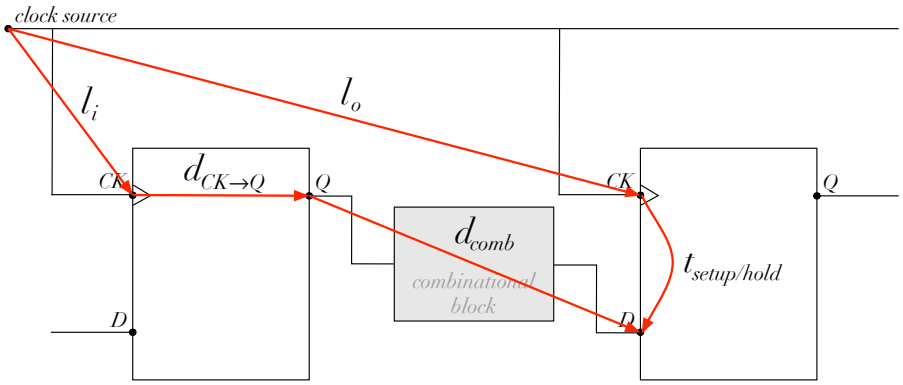

Figure 6: Propagation between two flip-flops.

combinational block $d_{CK\to Q}$ time later. If the propagation delay in the combinational block is d_{comb}, then the data will be available at the input of the capturing flip-flop at time $l_i + d_{CK\to Q} + d_{comb}$. Assuming the clock period to be a deterministic constant T, then the next clock edge will reach the capturing flip-flop at time $T + l_o$. For correct operation, the data must be be available at the input of the capturing flip-flop t_{setup} before the next clock edge reaches the capturing flip-flop. Therefore, at the data input pin, D, we have the following.

$$at_D^{late} = l_i + d_{CK\to Q} + d_{comb}^{late} \tag{31}$$

$$rat_{setup} = rat_D^{late} = T + l_o - t_{setup} \tag{32}$$

A similar condition can be derived for ensuring that the hold time is respected. The data input of the capturing flip-flop must remain stable for at least t_{hold} after the clock edge reaches the corresponding CK pin. Consequently, at the data input pin, D, we have the following.

$$at_D^{early} = l_i + d_{CK\to Q} + d_{comb}^{early} \tag{33}$$

$$rat_{hold} = rat_D^{early} = l_o + t_{hold} \tag{34}$$

The previous arrival times and required arrival times induce setup and hold slacks, which can be computed from (5) and (6). Please notice that the preset and clear values are presented for completeness, and should be ignored for the timing analysis task proposed in this contest.

4 Output

4.1 Projected Statistical Timing Quantities

Various projection techniques are possible for any parametric timing quantity T expressed as

$$T = \mu + a_v\Delta V + a_t\Delta T + a_l\Delta L + a_w\Delta W + a_h\Delta H + a_m\Delta M + a_r\Delta R.$$

Three projection techniques are required in this contest and are as follows.

1. **MEAN_ONLY**: In this projection approach, only the nominal or mean value μ of T is desired.
2. **SIGMA_ONLY**: In this projection approach, the standard deviation σ_T of T is desired. σ_T is computed as $\sqrt{a_v^2 + a_t^2 + a_l^2 + a_w^2 + a_h^2 + a_m^2 + a_r^2}$.
3. **WORST_CASE**: In this projection approach, all device parameter sensitivities are root sum squared to a *device* sensitivity a_d, and then this device sensitivity along with random and metal sensitivities are each projected to a +3 or −3 sigma corner to reflect a worst case projection of T. Mathematically, $a_d = \sqrt{a_v^2 + a_t^2 + a_l^2 + a_w^2 + a_h^2}$. For all late mode timing quantities like late mode arrival time or slew, the returned value should be $(\mu + 3|a_d| + 3|a_m| + 3|a_r|)$, while for all early mode quantities like early mode arrival time or slew and all slacks (both early and late mode), the returned value should be $(\mu - 3|a_d| - 3|a_m| - 3|a_r|)$.

4.2 Tool Input

The tool should expect 3 filenames as arguments, corresponding to the library file, netlist file, and the output file, in that order, respectively.

4.3 Tool Output

The result of the timing analysis of a circuit, which should be produced in the specified output file, will consist of two consecutive sets of lines, which will list projected values of parametric timing quantities like arrival times, slews and slacks.

The first set of lines will consist of a list of lines, starting with the `at` keyword, one per primary output node, containing the node name followed by the corresponding early and late projected arrival times, as well as the early and late projected slews. This list of nodes should be ordered lexicographically in ascending order (using the ASCII code ordering sequence).

The second set of lines will consist of a list of lines, starting with the `slack` keyword, one per required arrival time constraint, followed by the node name and either the `early` or `late` keyword, indicating either early or late mode, respectively. Finally, the value of the projected slack should be printed. This list of nodes should also be ordered lexicographically in ascending order (using the ASCII code ordering sequence). The `early` slack will appear before the `late` slack, for nodes where both exist. Slacks are induced either by explicit required arrival time constraints defined in the netlist file, or by implicit setup and hold constraints, that must be considered for every flip-flop in the circuit. The slacks (early, late, or both) should therefore be reported for all the nodes in the fanin cone of any given node for which an explicit or implicit required arrival time constraint must be considered.

The projection technique used in the output should be based on the value of the shell environmental variable `$TAU_PROJECTION` which could be any of `MEAN_ONLY`, `SIGMA_ONLY`, or `WORST_CASE`. If

the variable is not set, the default projection should be `WORST_CASE`. While the two former projection techniques will be primarily used for testing and debugging, final evaluations would be based on `WORST_CASE` projection.

All numerical results will be given in seconds and printed in scientific notation, with 5 decimal places (e.g. `1.23456e-10`). All keywords and variable fields should be separated by a white space.

```
at <node> <at early fall> <at early rise> <at late fall> <at late rise>
<slew early fall> <slew early rise> <slew late fall> <slew late rise>
...
slack <node> early <slack early fall> <slack early rise>
slack <node> late <slack late fall> <slack late rise>
...
```

---

5 Evaluation

5.1 Computational Infrastructure

The submissions will be evaluated on a machine with the following characteristics:

- Dual Intel Xeon Quad Core Processor E5410 @ 2.33GHz, 6144Kb cache
- 24GB of RAM

The following software is installed:

- Fedora Core 17 (Beefy Miracle), 64 bit;
- GCC 4.7.2 20120921;
- OpenMP 3.1;
- POSIX Threads;

The utilization of parallelization techniques (especially multi-threaded) is strongly encouraged. The evaluation machine can execute a maximum of 8 threads concurrently. Remember that you should submit a **binary** of your tool, that is compatible with the infrastructure above.

5.2 Criteria

Entries will be judged based on a weighted combination of accuracy of results, tool run-time, and tool peak-memory consumption. The results from a Monte Carlo simulator will be used as the golden reference for accuracy. Several inaccuracy range buckets (e.g., $0-5\%$, $5-10\%$, $10-20\%$) will be assumed wherein results within any bucket would be awarded the same score, and larger inaccuracy buckets would be awarded lower scores. Results outside the largest inaccuracy bucket would be awarded 0 points irrespective of their run-time and memory consumption. Inaccuracy measurements would be based on average inaccuracy and peak inaccuracy.

At least half of the final score would based on run-time and peak memory. Elapsed time will be used instead of CPU time, thus enabling contestants to take advantage of parallelization techniques. A large number of test runs will be executed, so as to enable accurate elapsed time measurements. Since netlist and library reading/parsing, as well as output generation times could be a non-negligible fraction of the overall run time, contestants are encouraged to apply parallelization/multi-threaded techniques for these steps as well.

Further details on the evaluation policy will be published shortly.

6 File Formats

6.1 Netlist

The netlist file, which contains the description of the circuit topology, is formatted as follows.

```
input <node>
output <node>
instance <cell name> <pin name>:<node> ... <pin name>:<node>
wire <port node> <tap node> ... <tap node>
    res <node> <node> <resistance>
    ...
    cap <node> <capacitance>
    ...
slew <node> <slew fall> <slew rise>
clock <node> <period>
at <node> <at fall early> <at fall late> <at rise early> <at rise late>
rat <node> <mode of operation> <rat fall> <rat rise>
```

Keywords:

- `input`, primary input node;
- `output`, primary output node;
- `instance`, cell instance;
- `wire`, interconnect net;
- `res`, `cap`, resistor and capacitor of a parasitic RC tree (can appear in any order);
- `slew`, input slew at the primary inputs;
- `clock`, clock input information (node and period);
- `at`, arrival time constraint (only used for primary input nodes);
- `rat`, required arrival time constraint.

Variable fields:

- `<node>`, `<port node>` and `<tap node>` are node names, of up to 64 characters in length, which can contain alphanumeric characters, the underscore or the dash (the first character must be a letter);
- `<cell name>` is the name of the library cell (exactly as it will appear in the cell library file), of up to 32 characters in length, which can contain only alphanumeric characters (the first character must be a letter);
- `<pin name>` is the name of a pin of the cell (exactly as it will appear in the cell library file), of up to 32 characters in length, which can contain only alphanumeric characters;

- `<resistance>` is the value of the resistance in Ohm, represented in scientific notation;
- `<capacitance>` is the value of the capacitance in Farad, represented in scientific notation;
- `<slew fall>` and `<slew rise>` are the fall and rise slews for the corresponding primary input, in seconds, represented in scientific notation. Early and late slews at the inputs are assumed to be identical;
- `<period>` is the clock period in seconds, represented in scientific notation;
- `<at fall early>`, `<at fall late>`, `<at rise early>` and `<at rise late>` are real numbers, represented in scientific notation, which represent arrival time constraints for fall/rise transitions in early/late mode, at the primary inputs, in seconds;
- `<mode of operation>`, is the mode of operation and can be either `early` or `late`;
- `<rat fall>` and `<rat rise>` are real numbers, represented in scientific notation, which represent required arrival time constraints for rise/fall transitions and early/late mode, in seconds.

If no input slew is defined for any given primary input, it should be assumed to be 1e-12, for both fall and rise transitions. The design will have one clock input pin (one clock domain) at most. The netlist file does not contain any parametric values, and a deterministic timing quantity (e.g. arrival time) may be considered a parametric quantity with 0 sensitivity to all parameters.

6.2 Cell Library

The cell library file, that contains the timing information of each cell, is formatted as follows.

```
metal <sigma corner> <resistance scale factor> <capacitance scale factor>
  ...
cell <cell name>
  pin <pin name> input <fall capacitance> <rise capacitance>
  pin <pin name> output
  pin <pin name> clock
  ...
  timing <input pin name> <output pin name> <timing sense>
<fall slew> <rise slew> <fall delay> <rise delay>
  setup <clock pin name> <input pin name> <edge type>
<fall constraint> <rise constraint>
  hold <clock pin name> <input pin name> <edge type>
<fall constraint> <rise constraint>
  preset <input pin name> <output pin name> <edge type> <slew> <delay>
  clear <input pin name> <output pin name> <edge type> <slew> <delay>
```

Keywords:

- `metal`, metal parameter scalars at given sigma corner;
- `cell`, start of cell definition;

- `pin`, start of pin definition;
- `input`, `output` and `clock`, pin type;
- `timing`, delay;
- `setup`, setup time;
- `hold`, hold time;
- `preset`, preset time (output node will be set to *high*);
- `clear`, clear time (output node will be set to low *low*).

Variable fields:

- `<sigma corner>` is the sigma corner value (σ) of metal parameter ΔM for which resistance and capacitance scale factors are provided.
- `<resistance scale factor>` is the value (m_R^σ) by which the nominal interconnect resistance provided in the netlist should be scaled at the given metal sigma corner;
- `<capacitance scale factor>` is the value (m_C^σ) by which the nominal interconnect capacitance provided in the netlist should be scaled at the given metal sigma corner;
- `<cell name>` is the name of the cell, of up to 32 characters in length, which can contain only alphanumeric characters (the first character must be a letter);
- `<pin name>` is the name of a pin of the cell, of up to 32 characters in length, which can contain only alphanumeric characters (the first character must be a letter);
- `<fall capacitance>` and `<rise capacitance>` are values of the pin's input capacitances in Farad, for rise/fall transitions, represented in scientific notation;
- `<input pin name>`, `<output pin name>` and `<clock pin name>` are the names of the input, output and clock pins of a given delay or constraint specification, of up to 32 characters in length, which can contain only alphanumeric characters (the first character must be a letter);
- `<timing sense>`, can be one of:
  - `positive_unate`, transition direction is maintained from input to output (rise→rise, fall→fall);
  - `negative_unate`, transition direction is reversed from input to output (rise→fall, fall→rise);
  - `non_unate`, transition direction cannot be inferred from a single input (take the worst, among rise/fall);
- `<slew>`, `<fall slew>`, `<rise slew>`, are each given by 9 real numbers separated by white spaces, which correspond to the parameters x, y, z, $k_{s,v}$, $k_{s,t}$, $k_{s,l}$, $k_{s,w}$, $k_{s,h}$, and $k_{s,r}$ of Eqn. (24) (fall/rise refers to the transition direction in the output pin);
- `<delay>`, `<fall delay>` and `<rise delay>`, are each given by 9 real numbers separated by white spaces, which correspond to the parameters a, b, c, $k_{d,v}$, $k_{d,t}$, $k_{d,l}$, $k_{d,w}$, $k_{d,h}$, and $k_{d,r}$ of Eqn. (23) (fall/rise refers to the transition direction in the output pin);

- `<edge type>`, can be one of:
  - `falling`, constraint applies to the falling clock edge;
  - `rising`, constraint applies to the rising clock edge;
- `<fall constraint>` and `<rise constraint>`, are each given by 3 real numbers separated by white spaces, which correspond to the parameters g, h and j of (29), or m, n and p of (30), if we are dealing with setup constraints or hold constraints, respectively;

The preset and clear values are presented for completeness, and should be ignored for the timing analysis task proposed in this contest.

7 Example

Consider the small example circuit illustrated in Figure 7, and the corresponding library cells, represented on the right side of the figure.
Netlist file:

```
input in_1
input in_2
input in_3
input in_4
output out
instance AND2X1 A:in_1 B:in_2 Y:w
instance XOR2X1 A:u B:in_4 Y:v
instance NOR2X1 A:k B:h Y:out
wire w k
    res w r 0.355
    cap r 1.23423e-13
    res r k 0.7884
    cap k 0.8e-14
wire v h
    res v h 0.5
    cap h 1.37e-13
wire in_3 u
    res in_3 h 0.75
    cap u 1.44e-13
at in_1 0 0 0 0
at in_2 0 0 0 0
at in_3 0 0 0 0
at in_4 0 0 0 0
rat k late 1e-13 2e-13
```

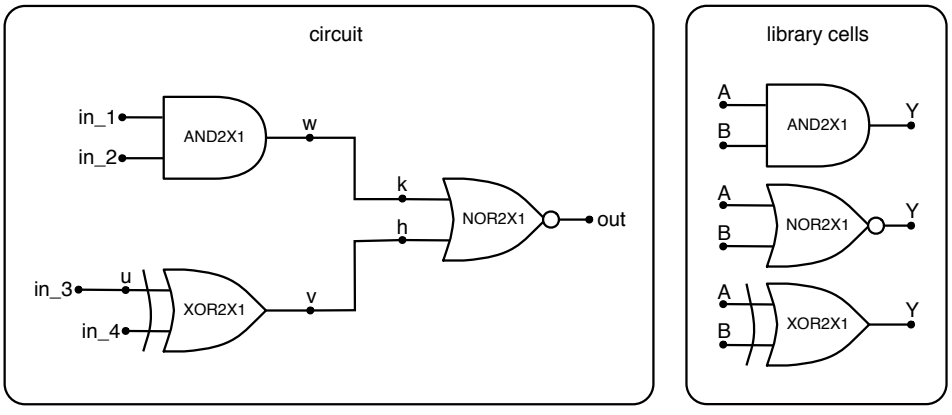

Figure 7: Small example circuit.

Cell library file:

```
metal 0 1.0 1.0
metal 3 0.85 1.24
cell AND2X1
  pin A input 5.14e-16 5.34e-16
  pin B input 5.35e-16 5.7e-16
  pin Y output
  timing A Y positive_unate 4.78193e-12 4807.29 0.000196751 -0.00602524 0.00585616
    0.00021047 -0.0000985957 0.000195775 0.00019961 6.88656e-12 11138.6 0.0000843378
    -0.012127 0.037587 0.000227912 -0.0000493878 0.0000838057 0.00019257 6.25087e-11
    6032.36 0.00247343 -0.0026644 0.00159043 0.0000694437 -0.000138492 0.000119349
    0.000157218 5.56981e-11 11789 -0.000183877 -0.003468 0.00371931 0.0000911961
    -0.00012835 0.00023857 0.00022949
  timing B Y positive_unate 5.98666e-12 4868.63 0.000112529 -0.0055235 0.00511724
    0.000158928 -0.000179324 0.0000354006 0.000151742 6.69766e-12 11142.7
    0.0000819104 -1.23305 0.03842 0.0000047514 -0.0000607217 0.0000343079 0.00020144
    7.51493e-11 6005.01 0.00261237 -0.00222054 0.00135711 0.0000391698 -0.000100236
    0.0000324476 0.0000272022 5.93502e-11 11798.7 -0.000224446 -0.0039269 0.00354033
    0.000249731 -0.0000545642 0.000128233 0.000209778
cell XOR2X1
  pin A input 4.15e-16 4.35e-16
  pin B input 4.34e-16 4.7e-16
  pin Y output
  timing A Y non_unate 5.78193e-12 5807.29 0.000196741 -0.00602425 0.00484616
    0.00021057 -0.0000984947 0.000194774 0.00019961 6.88646e-12 11138.6 0.0000853378
    -0.012127 0.0374874 0.000227912 -0.0000593878 0.0000838047 0.00019247 6.24087e-11
    6032.36 0.00257353 -0.0026655 0.00149053 0.0000695537 -0.000138592 0.000119359
    0.000147218 4.46981e-11 11789 -0.000183877 -0.003568 0.00371931 0.0000911961
    -0.00012834 0.00023847 0.00022959
  timing B Y non_unate 4.98666e-12 5868.63 0.000112429 -0.0044234 0.00411725
    0.000148928 -0.000179325 0.0000345006 0.000141752 6.69766e-12 11152.7
    0.0000819105 -0.0123304 0.03852 0.0000057415 -0.0000607217 0.0000353079
    0.00020155 7.41593e-11 6004.01 0.00261237 -0.00222045 0.00134711 0.0000391698
    -0.000100236 0.0000325576 0.0000272022 4.93402e-11 11798.7 -0.000225556
    -0.0039269 0.00345033 0.000259731 -0.0000454652 0.000128233 0.000209778
cell NOR2X1
  pin A input 5.13e-16 5.33e-16
  pin B input 5.35e-16 5.7e-16
  pin Y output
  timing A Y negative_unate 3.78193e-12 3807.29 0.000196761 -0.00602623 0.00686616
    0.00021037 -0.0000986967 0.000196776 0.00019961 6.88666e-12 11138.6 0.0000833378
    -0.012127 0.0376876 0.000227912 -0.0000393878 0.0000838067 0.00019267 6.26087e-11
    6032.36 0.00237333 -0.0026633 0.00169033 0.0000693337 -0.000138392 0.000119339
    0.000167218 6.66981e-11 11789 -0.000183877 -0.003368 0.00371931 0.0000911961
    -0.00012836 0.00023867 0.00022939
  timing B Y negative_unate 6.98666e-12 3868.63 0.000112629 -0.0066236 0.00611723
    0.000168928 -0.000179323 0.0000363006 0.000161732 6.69766e-12 11132.7
    0.0000819103 -0.012330 0.03832 0.0000037613 -0.0000607217 0.0000333079 0.00020133
    7.61393e-11 6006.01 0.00261237 -0.00222063 0.00136711 0.0000391698 -0.000100236
    0.0000323376 0.0000272022 6.93602e-11 11798.7 -0.000223336 -0.0039269 0.00363033
    0.000239731 -0.0000636632 0.000128233 0.000209778
```

8 Implementation Tips

- You should not report slacks for the internal nodes of wires. Slacks should only be reported for regular nodes if required arrival times exists at the node.
- You will need to perform two distinct wire delay computations for Fall and Rise transitions (per mode, early and late), since the tap loads may be different for each transition type (taps are usually connected to cell inputs, which have distinct Fall/Rise input capacitances). Moreover, the input slews for Fall and Rise transitions may also be distinct.
- When forward computing ATs, you should do the MAX for Late mode and the MIN for Early mode. When backward computing RATs, you should do the MIN for Late mode and the MAX for Early mode.
- When computing Setup time, you should assume Early mode clock slew and AT and Late mode data input slew and AT. When computing Hold time, you should assume Late mode clock slew and AT and Early mode data input slew and AT.
- If you want to format the output in scientific notation with 5 decimal places, in C++ you should do:

```
cout.setf(ios::scientific,ios::floatfield);
cout.precision(5);
```

References

[1] D Blaauw, K Chopra, A Srivastava, and L Scheffer. Statistical Timing Analysis: From Basic Principles to State-of-the-art. In *IEEE Transactions on Computer-Aided Design, 27(4) April 2008*, pages 589–607.

[2] H. Chang and S. S. Sapatnekar. Statistical timing analysis considering spatial correlations using a single PERT-like traversal. pages 621–625, 2003.

[3] C. E. Clark. The greatest of a finite set of random variables. In *Operations Research, Vol. 9, No. 2 (Mar - Apr)*, pages 145–162, 1961.

[4] W. C. Elmore. The Transient Response of Damped Linear Networks with Particular Regard to Wide-Band Amplifiers. *Journal of Applied Physics*, 19(1):55–63, January 1948.

[5] R. Goering. *Why Multi-Mode, Multi-Corner (MMMC) ECO Closure Requires a New Signoff Approach*. http://www.cadence.com/Community/blogs/ii/archive/2012/08/06/why-multi-mode-multi-corner-mmmc-eco-closure-requires-a-new-signoff-approach.aspx.

[6] R. Gupta, B. Tutuianu, and L. T. Pileggi. The Elmore Delay as a Bound for RC Trees with Generalized Input Signals. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(1):95–104, January 1997.

[7] P. Penfield Jr. and J. Rubinstein. Signal Delay in RC Tree Networks. In *Design Automation Conference*, pages 613–617, 1981.

[8] C. V. Kashyap, C. J. Alpert, F. Liu, and A. Devgan. Closed-Form Expressions for Extending Step Delay and Slew Metrics to Ramp Inputs for RC Trees. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(4):509–516, April 2004.

[9] C. L. Ratzlaff and L. T. Pillage. RICE: Rapid Interconnect Circuit Evaluation Using AWE. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(6):763–776, June 1994.

[10] D. Sinha, H. Zhou, and N. V. Shenoy. Advances in computation of the maximum of a set of Gaussian random variables. In *IEEE Transactions on Computer-Aided Design, 26(8) August 2007*, pages 1522–1533.

[11] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan. First-order incremental block-based statistical timing analysis. pages 331–336, 2004.