PATMOS'2011 Timing Analysis Contest

v1.3 - August 2, 2011

1 Introduction

This document describes the timing models and file formats that will be used in the PATMOS'2011 Timing Analysis Contest. The latest news, contacts and other information should be obtained from the contest web page at:

`http://patmos-tac.inesc-id.pt`

2 Timing Analysis

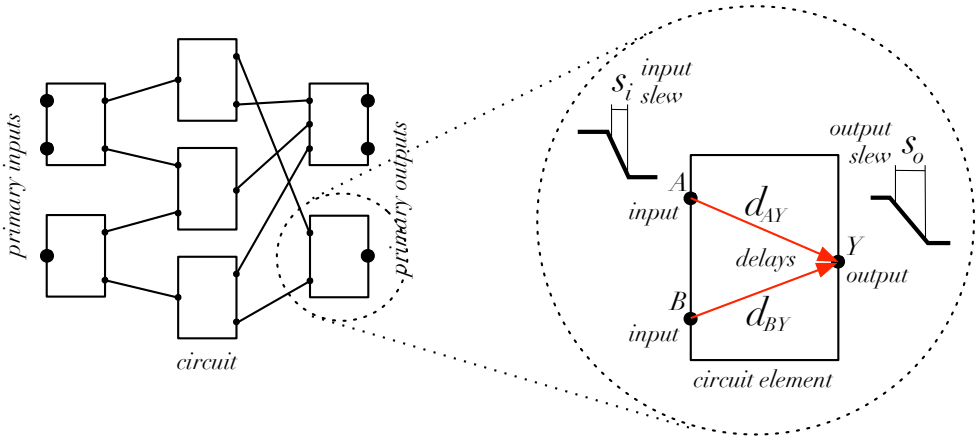

Figure 1: Circuit and circuit element characterization.

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 to 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 Eqns. (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.

For the purpose of the PATMOS'2011 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. The syntax of such files will be described in Section 6.

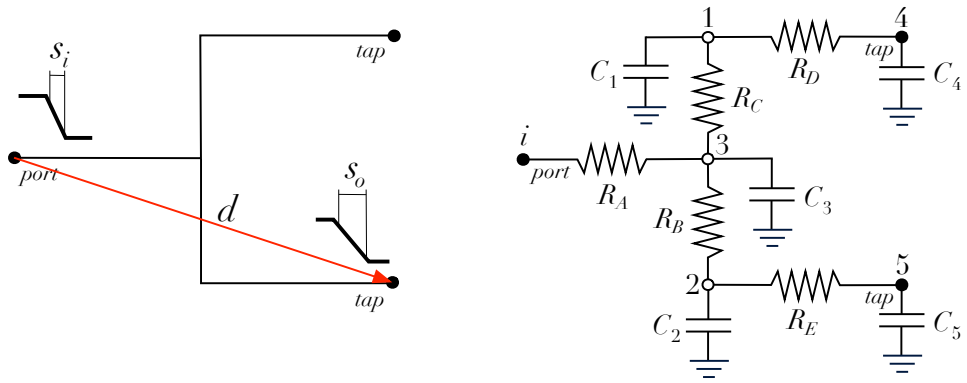

Figure 2: Interconnect characterization.

3 Models

The models to be encountered during timing analysis, are of two types: cell or interconnect 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 floating resistors.

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 [1], whereby the delay is approximated by the value of the first moment of the impulse response. For RC tree networks, [3] 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,

$$d_e = \sum_k R_{ke}\, C_k \tag{9}$$

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,

$$d_5 = R_A(C_1 + C_3 + C_4) + (R_A + R_B)\, C_2 + (R_A + R_B + R_E)\, C_5 \tag{10}$$

The value of the output slew on any given tap node o can be approximately computed by a two-step procedure. First, one computes the output slew of the impulse response on o, which was

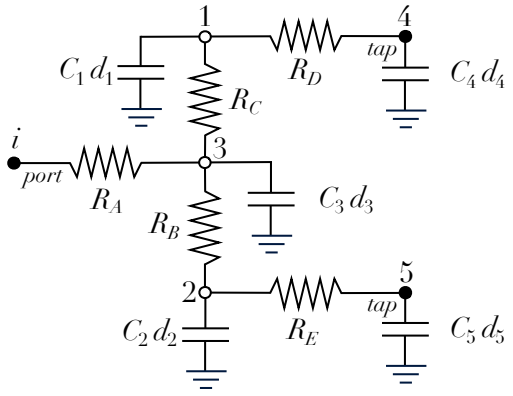

Figure 3: Modified RC tree for computing the second moment of impulse response.

observed [1, 2] to be well approximated by the following expression,

$$\hat{s}_o \approx \sqrt{2\beta_o - d_o^2} \tag{11}$$

where β_o is the second moment of the impulse response at node o, and d_o is the corresponding Elmore delay computed from Eqn. (9), for node o. The value of β_o can be computed through the efficient path-tracing algorithm for moment computation proposed in [5], which is a generalization of the algorithm proposed in [3] and described earlier.

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 Eqn. (9). 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, which will be given by,

$$\beta_e = \sum_k R_{ke}\, C_k\, d_k \tag{12}$$

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

$$\beta_5 = R_A\,(C_1 d_1 + C_3 d_3 + C_4 d_4) + (R_A + R_B)\,C_2 d_2 + (R_A + R_B + R_E)\,C_5 d_5 \tag{13}$$

After computing $\hat{s}_o$ from Eqn. (11), we proceed to compute the slew of the response to the input ramp, s_o, for which a good approximation is proposed in [4], given by the simple expression,

$$s_o \approx \sqrt{s_i^2 + \hat{s}_o^2} \tag{14}$$

As will be discussed next, the equivalent capacitance seen from the port, that we will designate by C_L, is one of the parameters necessary for computing cell delay. Several sophisticated models have been proposed for computing C_L, however, since the application of such models is out of the scope of the present contest, we will employ a much simpler model. We will consider C_L to be the sum of all the capacitances in the parasitic RC tree,

$$C_L = \sum_k C_k \tag{15}$$

For the example net illustrated in Figure 2 (right), we trivially have,

$$C_L = C_1 + C_2 + C_3 + C_4 + C_5 \tag{16}$$

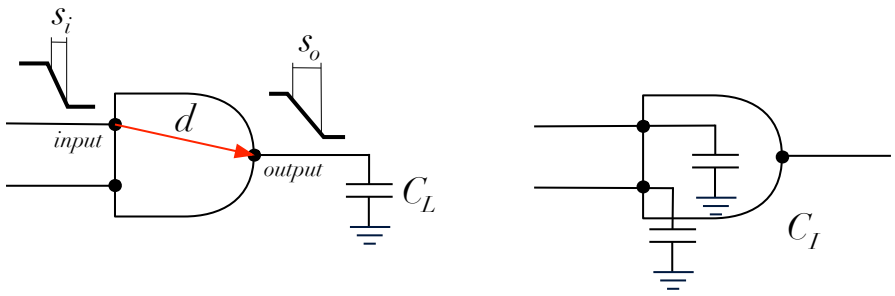

Figure 4: Combinational cell characterization.

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:

$$d = a + b\, C_L + c\, s_i \tag{17}$$

$$s_o = x + y\, C_L + z\, s_i \tag{18}$$

where a, b, c, x, y and z are cell-dependent constants and C_L and s_i are the output load and input slew, respectively. a, b, c, x, y and z are provided, for each cell and for rise/fall transition, in the cell library file.

Another relevant parameter for cell characterization is the input capacitance at each of its input pins, that we will designate by C_I. Such capacitance is a fixed value provided, for each pin and for each rise/fall transition, in the cell library file.

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 librares for storage elements. Other figures include

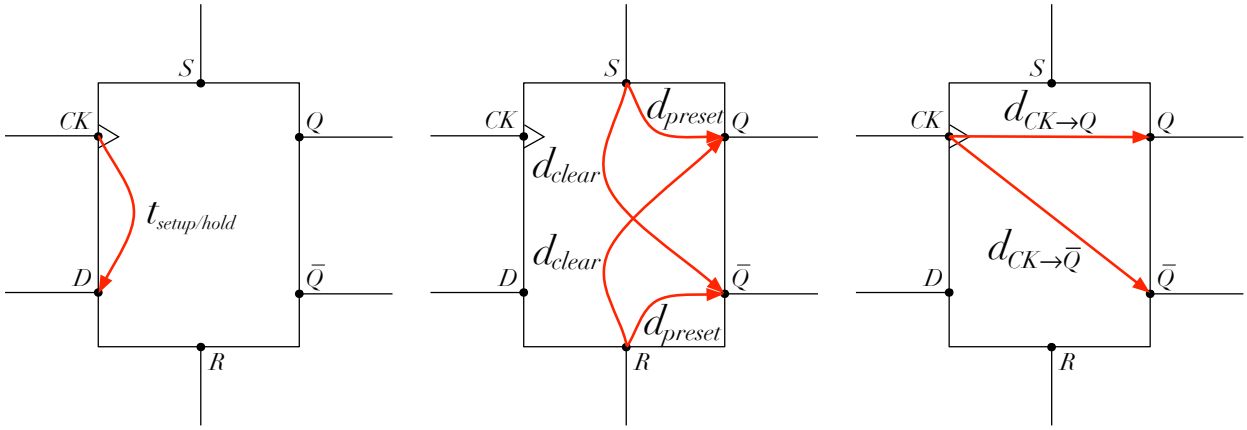

Figure 5: Flip-flop characterization.

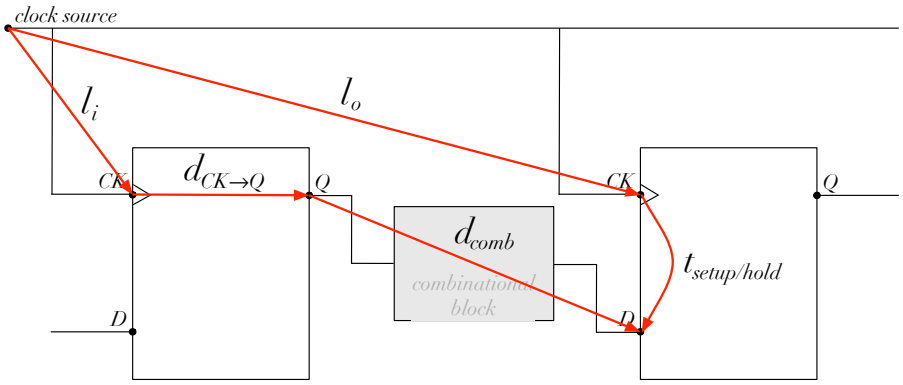

Figure 6: Propagation between two flip-flops.

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,

$$t_{setup} = g + h\ s_i^{CK} + j\ s_i^D \tag{19}$$

$$t_{hold} = m + n\ s_i^{CK} + p\ s_i^D \tag{20}$$

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 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 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:

$$at_D^{late} = l_i + d_{CK\to Q} + d_{comb}^{late} \tag{21}$$

$$rat_{setup} = rat_D^{late} = T + l_o - t_{setup} \tag{22}$$

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 que corresponding CK pin. Consequently, at the data input pin, D, we have:

$$at_D^{early} = l_i + d_{CK \to Q} + d_{comb}^{early} \tag{23}$$

$$rat_{hold} = rat_D^{early} = l_o + t_{hold} \tag{24}$$

The previous arrival times and required arrival times induce setup and hold slacks, which can be computed from Eqns. (5) and (6).

4 Output

The result of the timing analysis of a circuit, which should be produced in the standard output, will consist of two consecutive sets of lines, which will list arrival times 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 arrival times, as well as the early and late 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 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.

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.

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.

```
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 Processor E5410 (Quad-Core) @ 2.33GHz
- 24GB of RAM

The following software is installed:

- Fedora Core 13 (64-bit) - Kernel 2.6.34.8-68.fc13.x86_64;
- Gcc 4.4.5 20101112;
- OpenMP 3.0;
- POSIX Threads;

The utilization of parallelization techniques is encouraged. Remember that you should submit a **binary** of your tool, that is compatible with the infrastructure above.

5.2 Criteria

Entries will be first judged based on accuracy of results. If multiple correct entries are submitted, they will be ranked according to the elapsed 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.

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;
- `<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.

6.2 Cell Library

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

```
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:

- `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:

- `<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 3 real numbers separated by white spaces, which correspond to the parameters x, y and z of Eqn. (18) (fall/rise refers to the transition direction in the output pin);
- `<delay>`, `<fall delay>` and `<rise delay>`, are each given by 3 real numbers separated by white spaces, which correspond to the parameters a, b and c of Eqn. (17) (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 Eqn. (19), or m, n and p of Eqn. (20), 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
```

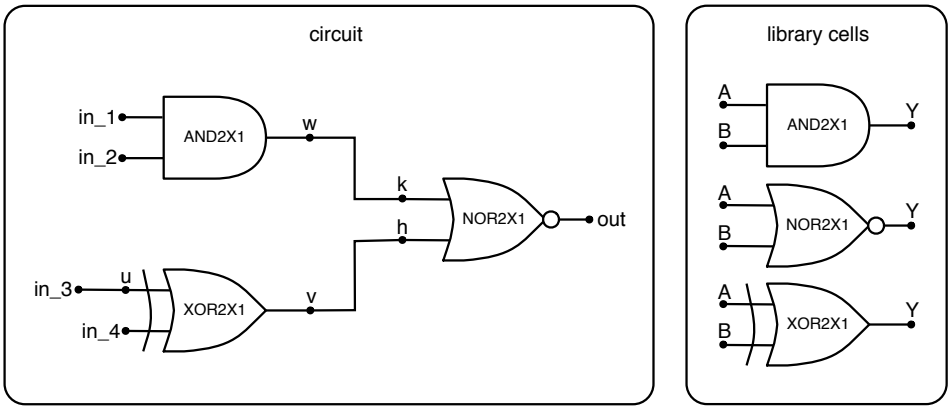

Figure 7: Small example circuit.

Cell library file:

```
cell AND2X1
  pin A input 0.00626946 0.0062621
  pin B input 0.00621027 0.00621493
  pin Y output
  timing A Y positive_unate 1.25688e-11 1755.06 0.0223926 1.06884e-11 2014.99
 0.0175593 7.89623e-11 1499.99 0.0390409 8.12886e-11 1471.14 0.0738917
  timing B Y positive_unate 8.78443e-12 1760.39 0.0242686 1.07371e-11 2048.62
 0.0191379 8.06078e-11 1488.61 0.0941661 8.05821e-11 1509.17 0.0193722
cell XOR2X1
  pin A input 0.0123581 0.0121292
  pin B input 0.0114021 0.0115424
  pin Y output
  timing A Y non_unate 2.65056e-11 2055.76 0.128813 3.29132e-11 2419.81 0.104
815 7.79759e-11 2226.06 0.0853261 7.07128e-11 2197 0.0871047
  timing B Y non_unate 2.8407e-11 1963.2 0.122103 3.24004e-11 2382.06 0.10718
 7.71562e-11 2093.48 0.0925534 6.76652e-11 2067.84 0.0689304
cell NOR2X1
  pin A input 0.00777251 0.00771846
  pin B input 0.00796173 0.00795906
  pin Y output
  timing A Y negative_unate 3.76407e-11 1267.73 0.277313 3.24898e-11 2113.92
0.1 93645 3.54943e-11 2229.71 0.0741437 2.42059e-11 2539.39 0.0998498
  timing B Y negative_unate -1.19569e-11 2133.54 0.230791 1.33793e-11 2567.5
0.1 66239 2.53227e-11 2518.47 0.00518459 2.30746e-11 2607.72 0.199106
```

8 Implementation Tips

- You should not report slacks for the internal nodes of wires. Slacks should only be reported for regular nodes.
- You will need to perform two distict wire delay computations for Fall and Rise transitions, 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] 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.

[2] Rohini Gupta, Bogdan Tutuianu, and Lawrence 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.

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

[4] Chandramouli V. Kashyap, Charles J. Alpert, Frank (Ying) Liu, and Anirudh 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.

[5] Curtis L. Ratzlaff and Lawrence 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.