数字集成电路静态时序分析基础

邸志雄 博士, zxdi@home.swjtu.edu.cn

西南交通大学信息科学与技术学院

CONTENT

Part 3: Standard Cell Library

01 › **Overview of Synopsys Timing Lib**

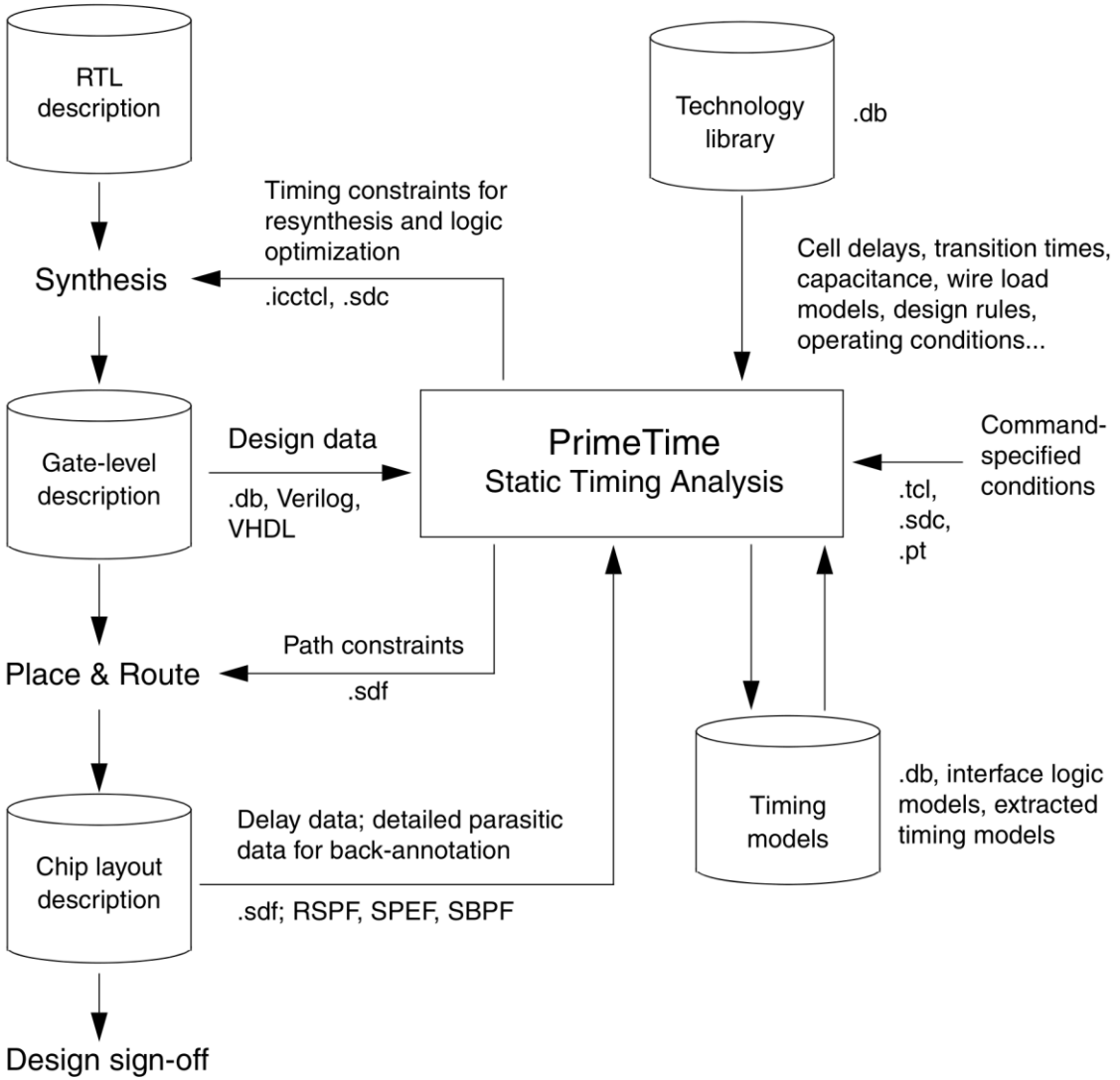
02 › Non-Linear Delay Model

03 › Threshold Specifications and Slew Derating

04 › Timing Models

05 › Wire Delay Models

Introduction

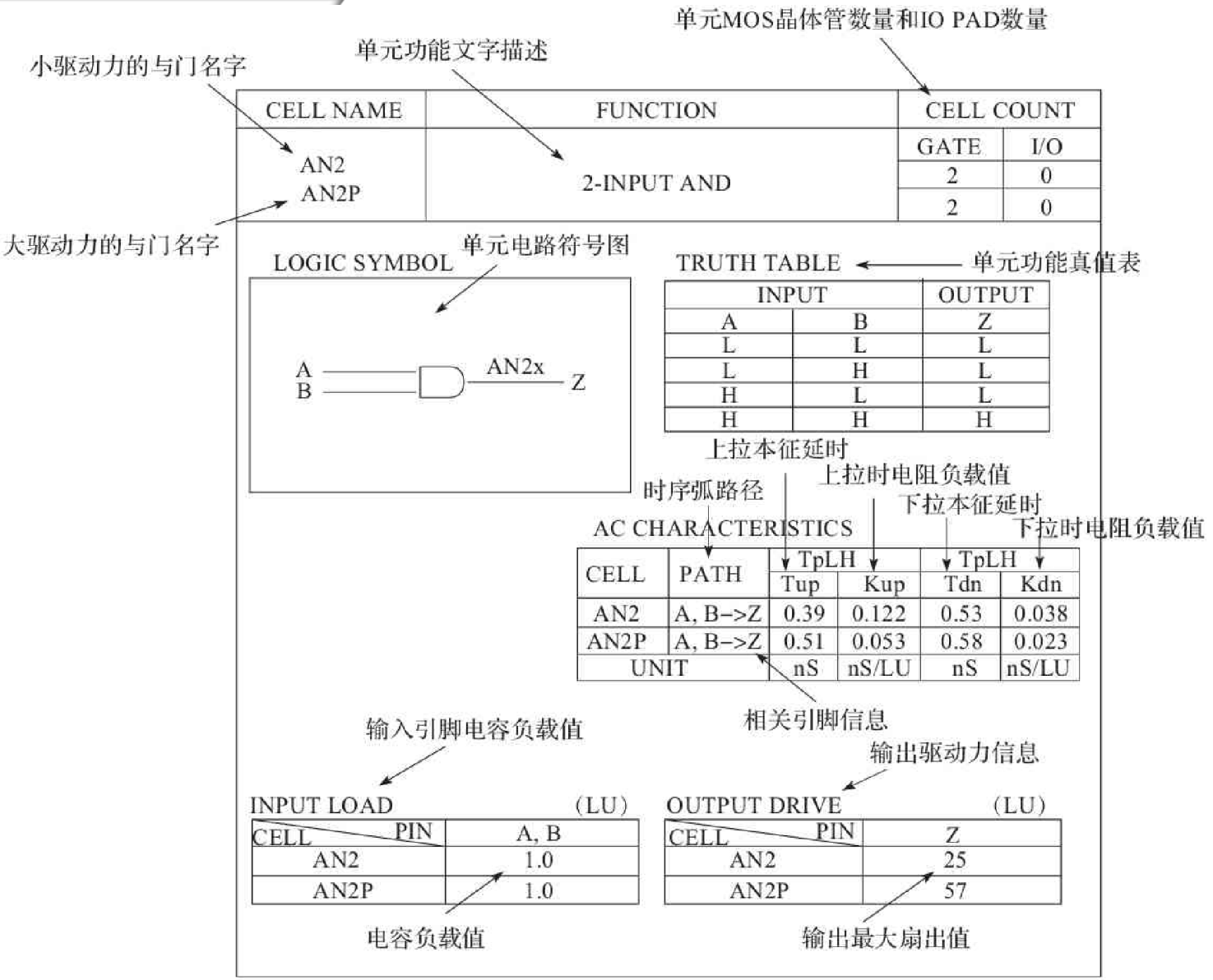


Overview of Synopsys Timing Lib

```
library ( smic13_tt ) {
    delay_model                     : table_lookup ;
    in_place_swap_mode              : match_footprint
    time_unit                       : "1ns" ;
    voltage_unit                    : "1V" ;
    current_unit                    : "1uA" ;
    pulling_resistance_unit         : "1kohm" ;
    leakage_power_unit              : "1nW" ;
    capacitive_load_unit            ( 1,pf ) ;
    slew_upper_threshold_pct_rise   : 90.00 ;
    slew_lower_threshold_pct_rise   : 10.00 ;
    slew_upper_threshold_pct_fall   : 90.00 ;
    slew_lower_threshold_pct_fall   : 10.00 ;
    input_threshold_pct_rise        : 50.00 ;
    input_threshold_pct_fall        : 50.00 ;
    output_threshold_pct_rise       : 50.00 ;
    output_threshold_pct_fall       : 50.00 ;
    nom_process                     : 1 ;
    nom_voltage                     : 1.2 ;
    nom_temperature                 : 25 ;
    revision                        : 0.1 ;
```

Overview of Synopsys Timing Lib

```
cell ( AND2CLKHD1X ) {
    area                 : 7.564 ;
    cell_leakage_power   : 0.344886 ;
    cell_footprint       : and2clk ;
    pin ( A ) {
        direction        : input ;
        capacitance      : 0.00226414 ;
    }
    pin ( B ) {
        direction        : input ;
        capacitance      : 0.00221157 ;
    }
    pin ( Z ) {
        direction        : output ;
        capacitance      : 0 ;
        max_capacitance  : 0.423245 ;
        function         : "(A B)" ;
        timing ( ) {
            related_pin  : "A" ;
            timing_sense : positive_unate ;
            cell_rise ( delay_template_6x6 ) {
                index_1 ( "0.0001, 0.02, 0.1, 0.2
                index_2 ( "0.033344, 0.093963, 0.
                values ( \
                    "0.067736, 0.074117, 0.09274,
                    "0.125151, 0.133677, 0.152551
                    "0.3343, 0.337365, 0.355432,
                    "0.5683, 0.578238, 0.598328,
                    "0.807801, 0.819009, 0.867235
```



CONTENT

Part 3: Standard Cell Library

01 › Overview of Synopsys Timing Lib

02 › **Non-Linear Delay Model**

03 › Threshold Specifications and Slew Derating

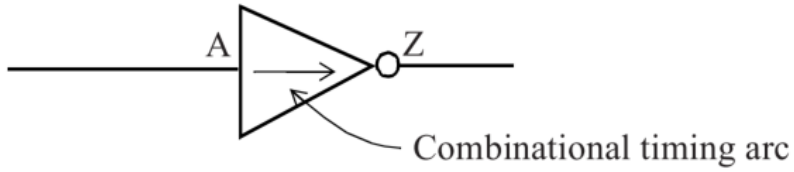
04 › Timing Models

05 › Wire Delay Models

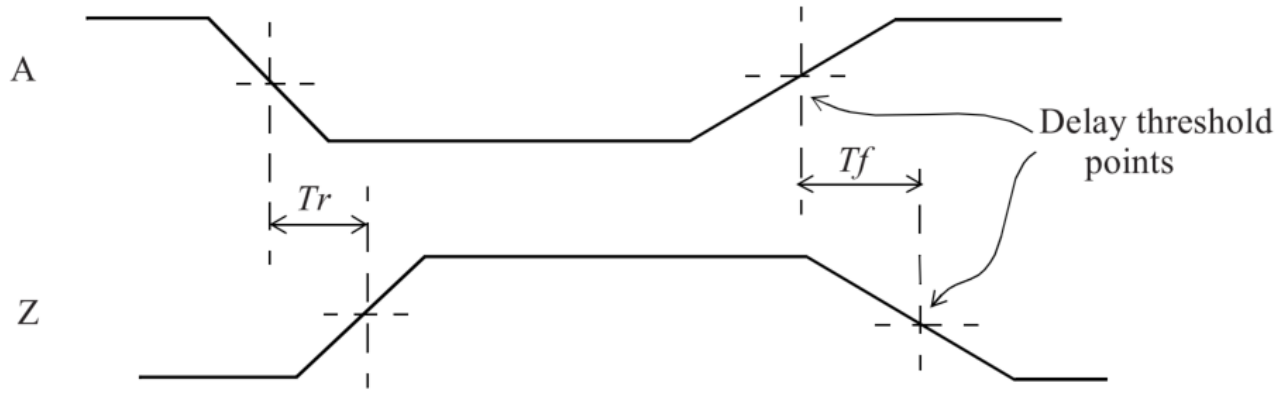
Timing Modeling

Let us first consider timing arcs for a simple inverter logic. Since it is an inverter, a rising (falling) transition at the input causes a falling (rising) transition at the output.

The two kinds of delay characterized for the cell are:



- Tr : Output rise delay
- Tf : Output fall delay



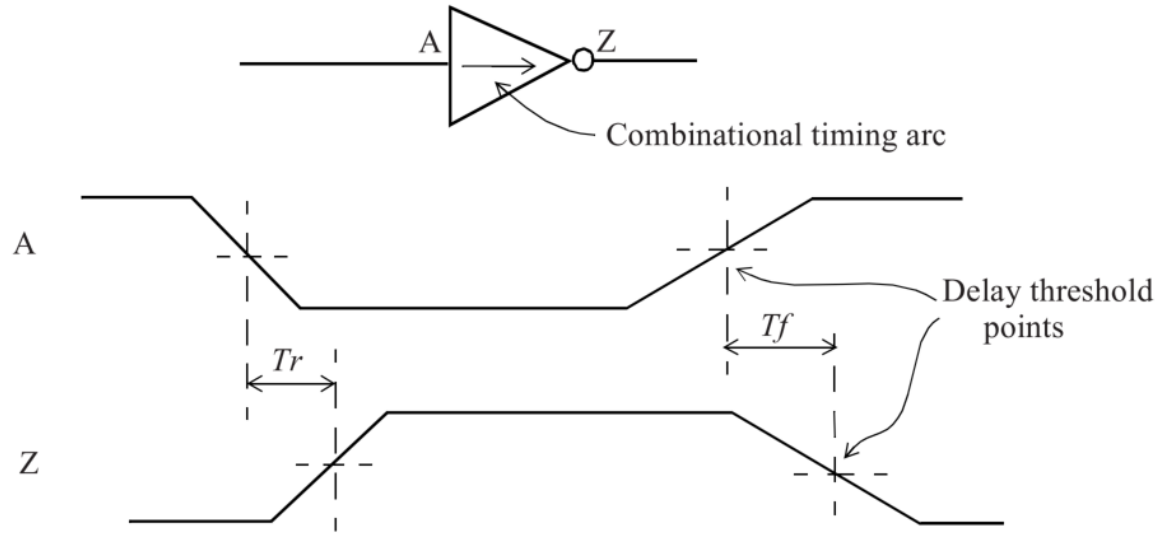
Notice that the delays are measured based upon the threshold points defined in a cell ibrary, which is typically 50% Vdd.

Timing Modeling

The delay for the timing arc through the inverter cell is dependent on two factors:

i. the **output load**, that is, the capacitance load at the output pin of the inverter, and

ii. the **transition time** of the signal at the input.



- The delay values have a direct correlation with the load capacitance - the larger the load capacitance, the larger the delay.
- In most cases, the delay increases with increasing input transition time.

Linear Timing Model

A simple timing model is a linear delay model, where the delay and the output transition time of the cell are represented as linear functions of the two parameters: input transition time and the output load capacitance.

The general form of the linear model for the delay, D, through the cell is illustrated below.

$$D = D0 + D1 * S + D2 * C$$

where D0, D1, D2 are constants, S is the input transition time, and C is the output load capacitance.

The linear delay models are not accurate over the range of input transition time and output capacitance for submicron tech nologies, and thus most cell libraries presently use the more complex models such as the non-linear delay model.

Non-Linear Delay Model

Most of the cell libraries include **table models** to specify the delays and timing checks for various timing arcs of the cell.

- The table models are referred to as NLDM (Non-Linear Delay Model) and are used for delay, output slew, or other timing checks.
- The table models capture the delay through the cell for various combinations of input transition time at the cell input pin and total output capacitance at the cell output.

Non-Linear Delay Model

An NLDM model for delay is presented in a **two-dimensional form**.

The two independent variables being the **input transition time** and the **output load capacitance**, and the entries in the table denoting the **delay**.

Here is an example of such a table for a typical inverter cell:

```
pin (OUT) {
  max_transition : 1.0;
  timing() {
    related_pin : "INP1";
    timing_sense : negative_unate;
    cell_rise(delay_template_3x3) {
      index_1 ("0.1, 0.3, 0.7"); /* Input transition */
      index_2 ("0.16, 0.35, 1.43"); /* Output capacitance */
      values (  /*   0.16      0.35      1.43 */ \
        /* 0.1 */   "0.0513,  0.1537,  0.5280", \
        /* 0.3 */   "0.1018,  0.2327,  0.6476", \
        /* 0.7 */   "0.1334,  0.2973,  0.7252");
      }
    cell_fall(delay_template_3x3) {
      index_1 ("0.1, 0.3, 0.7"); /* Input transition */
      index_2 ("0.16, 0.35, 1.43"); /* Output capacitance */
      values (  /*   0.16      0.35      1.43 */ \
        /* 0.1 */   "0.0617,  0.1537,   0.5280", \
        /* 0.3 */   "0.0918,  0.2027,   0.5676", \
        /* 0.7 */   "0.1034,  0.2273,   0.6452");
    }
```

Non-Linear Delay Model

The delays of the output pin OUT are described.

```
pin (OUT) {
 max_transition : 1.0;
 timing() {
  related_pin : "INP1";
  timing_sense : negative_unate;
  cell_rise(delay_template_3x3) {
   index_1 ("0.1, 0.3, 0.7"); /* Input transition */
   index_2 ("0.16, 0.35, 1.43"); /* Output capacitance */
   values (  /*  0.16      0.35      1.43 */ \
     /* 0.1 */   "0.0513,  0.1537,  0.5280", \
     /* 0.3 */   "0.1018,  0.2327,  0.6476", \
     /* 0.7 */   "0.1334,  0.2973,  0.7252");
   }
  cell_fall(delay_template_3x3) {
   index_1 ("0.1, 0.3, 0.7"); /* Input transition */
   index_2 ("0.16, 0.35, 1.43"); /* Output capacitance */
   values (  /*  0.16      0.35      1.43 */ \
     /* 0.1 */   "0.0617,  0.1537,   0.5280", \
     /* 0.3 */   "0.0918,  0.2027,   0.5676", \
     /* 0.7 */   "0.1034,  0.2273,   0.6452");
  }
```

Non-Linear Delay Model

There are **separate** models for the rise and fall delays (for the output pin) and these are labeled as cell_rise and cell_fall respectively.

The type of indices and the order of table lookup indices are described in the lookup table template delay_template_3x3.

```
pin (OUT) {
 max_transition : 1.0;
 timing() {
  related_pin : "INP1";
  timing_sense : negative_unate;
  cell_rise(delay_template_3x3) {
   index_1 ("0.1, 0.3, 0.7"); /* Input transition */
   index_2 ("0.16, 0.35, 1.43"); /* Output capacitance */
   values (  /*  0.16      0.35      1.43 */ \
     /* 0.1 */   "0.0513,  0.1537,  0.5280", \
     /* 0.3 */   "0.1018,  0.2327,  0.6476", \
     /* 0.7 */   "0.1334,  0.2973,  0.7252");
   }
  cell_fall(delay_template_3x3) {
   index_1 ("0.1, 0.3, 0.7"); /* Input transition */
   index_2 ("0.16, 0.35, 1.43"); /* Output capacitance */
   values (  /*  0.16      0.35      1.43 */ \
     /* 0.1 */   "0.0617,  0.1537,  0.5280", \
     /* 0.3 */   "0.0918,  0.2027,  0.5676", \
     /* 0.7 */   "0.1034,  0.2273,  0.6452");
  }
```

Non-Linear Delay Model

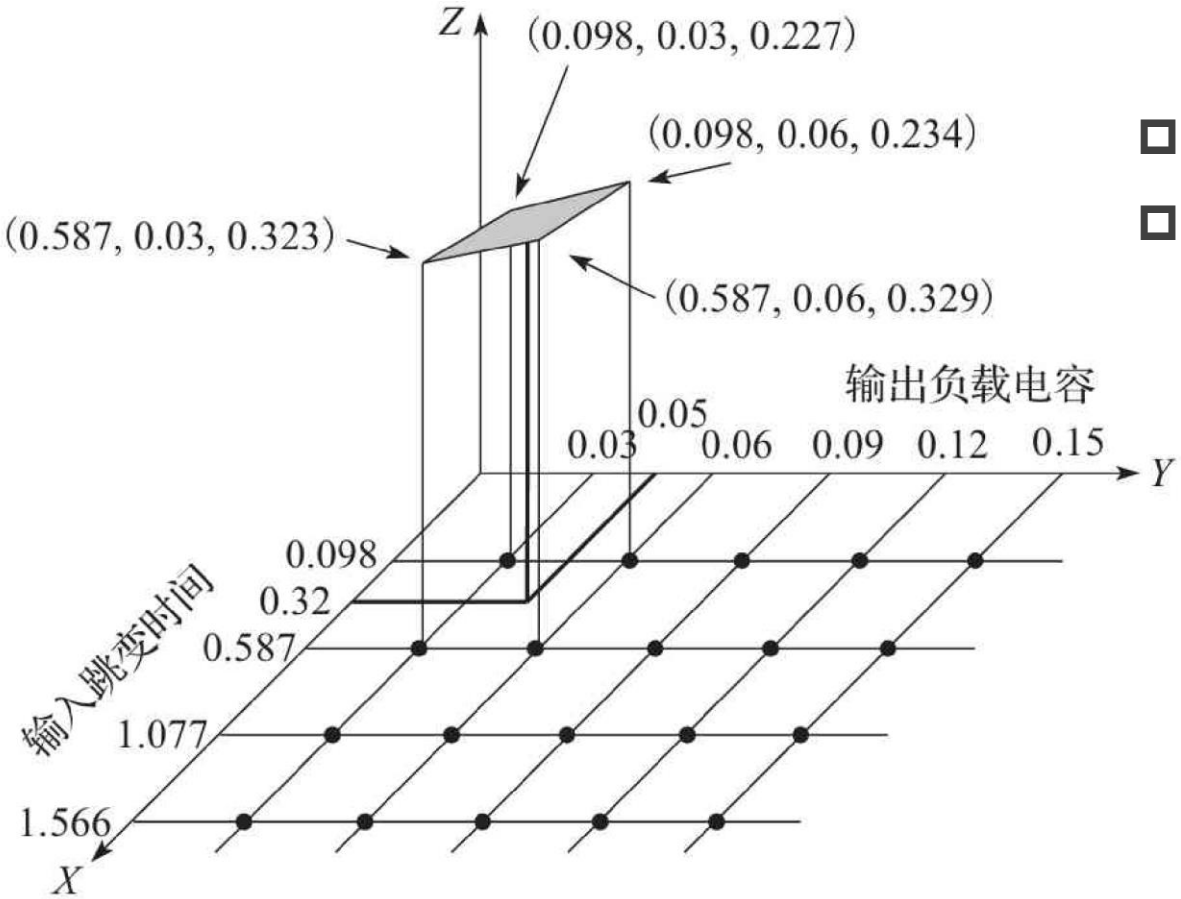
The example below corresponds to a general case where the lookup does not correspond to any of the entries available in the table.

```
fall_transition(delay_template_3x3) {
  index_1 ("0.1, 0.3 . . .");
  index_2 (". . . 0.35, 1.43");
  values ( \
    ". . . 0.1937, 0.7280", \
    ". . . 0.2327, 0.7676"
```

In such cases, **two-dimensional interpolation is utilized to provide the resulting timing value**. The two **nearest** table indices in each dimension are chosen for the table interpolation.

Non-Linear Delay Model

The example below corresponds to a general case where the lookup does not correspond to any of the entries available in the table.



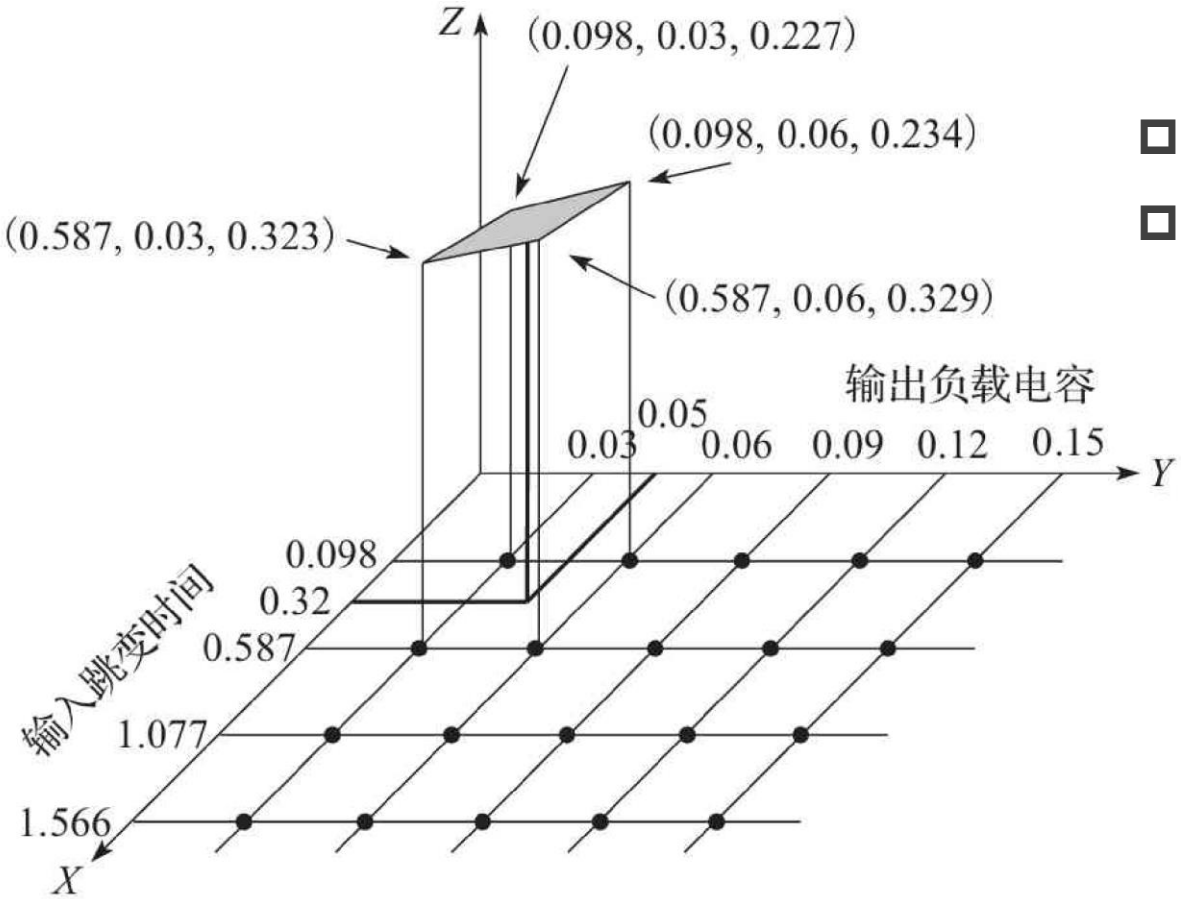
- X: Transition time（ns）;Y: CLoad（pf）
- Z: Tdelay（ns）

X=0.32,Y =0.05

Z=?

Non-Linear Delay Model

The example below corresponds to a general case where the lookup does not correspond to any of the entries available in the table.

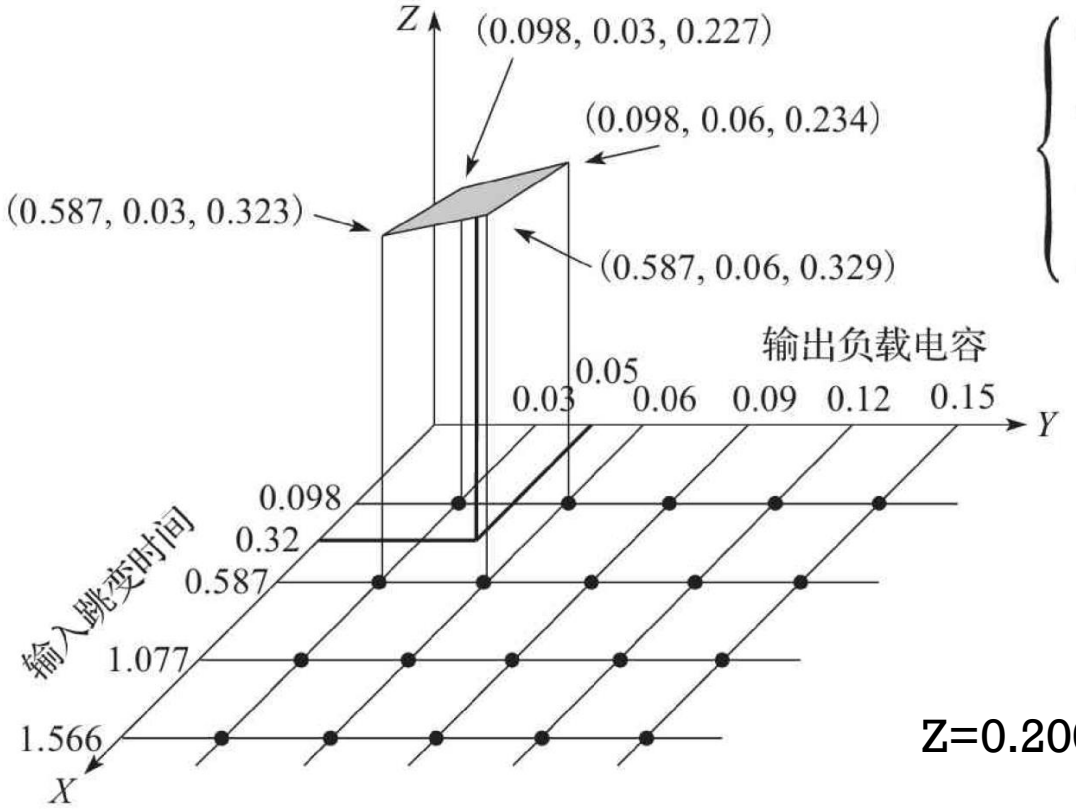


- X: Transition time（ns）;Y: CLoad（pf）
- Z: Tdelay（ns）

X=0.32,Y =0.05

Z=?

Non-Linear Delay Model



Gaussian Elimination

$$\begin{cases} 0.227 = A + B * 0.098 + C * 0.03 + D * 0.098 * 0.03 \\ 0.234 = A + B * 0.098 + C * 0.06 + D * 0.098 * 0.06 \\ 0.323 = A + B * 0.587 + C * 0.03 + D * 0.587 * 0.03 \\ 0.329 = A + B * 0.587 + C * 0.06 + D * 0.587 * 0.06 \end{cases}$$

⇩

A=0.2006，B=0.1983，C=0.2399，D=0.0677

⇩

Z=0.2006+0.1983×0.32+0.2399×0.05+0.0677×0.32×0.05=0.275

CONTENT

Part 3: Standard Cell Library

01 › Overview of Synopsys Timing Lib

02 › Non-Linear Delay Model

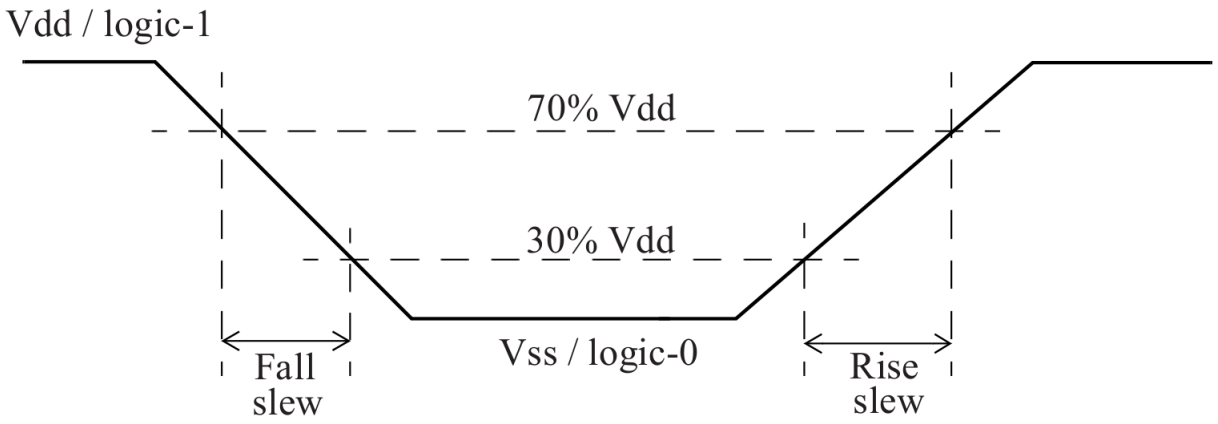
03 › **Threshold Specifications and Slew Derating**

04 › Timing Models

05 › Wire Delay Models

Threshold Specifications and Slew Derating

The slew 1 values are based upon the measurement thresholds specified in the library. Most of the previous generation libraries (0.25?m or older) used 10% and 90% as measurement thresholds for slew or transition time.

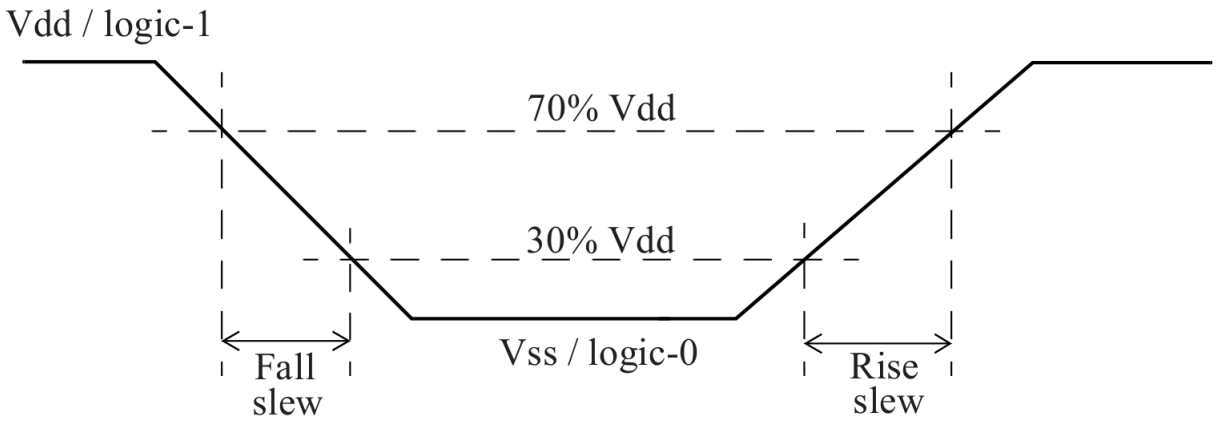



```
/* Threshold definitions */
slew_lower_threshold_pct_fall : 30.0;
slew_upper_threshold_pct_fall : 70.0;
slew_lower_threshold_pct_rise : 30.0;
slew_upper_threshold_pct_rise : 70.0;
input_threshold_pct_fall : 50.0;
input_threshold_pct_rise : 50.0;
output_threshold_pct_fall : 50.0;
output_threshold_pct_rise : 50.0;
slew_derate_from_library : 0.5;
```

Threshold Specifications and Slew Derating

The slew thresholds are chosen to correspond to the linear portion of the waveform. As technology becomes finer, the portion where the actual waveform is most linear is typically between 30% and 70% points.

Thus, most of the newer generation timing libraries specify slew measurement points as 30% and 70% of Vdd.

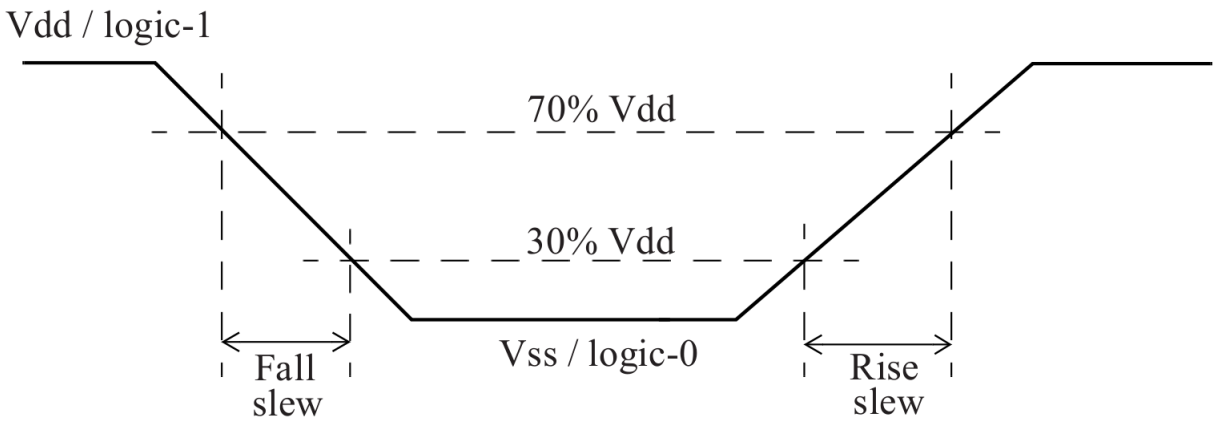



```
/* Threshold definitions */
slew_lower_threshold_pct_fall : 30.0;
slew_upper_threshold_pct_fall : 70.0;
slew_lower_threshold_pct_rise : 30.0;
slew_upper_threshold_pct_rise : 70.0;
input_threshold_pct_fall : 50.0;
input_threshold_pct_rise : 50.0;
output_threshold_pct_fall : 50.0;
output_threshold_pct_rise : 50.0;
slew_derate_from_library : 0.5;
```

Threshold Specifications and Slew Derating

However, because the transition times were previously measured between 10% and 90%, the transition times measured between 30% and 70% are usually doubled for populating the library. This is specified by the slew derate factor which is typically specified as 0.5.

The slew thresholds of 30% and 70% **with slew derate as 0.5** results in equivalent measurement points of 10% and 90%. An example settings of threshold is illustrated below.




```
/* Threshold definitions */
slew_lower_threshold_pct_fall : 30.0;
slew_upper_threshold_pct_fall : 70.0;
slew_lower_threshold_pct_rise : 30.0;
slew_upper_threshold_pct_rise : 70.0;
input_threshold_pct_fall : 50.0;
input_threshold_pct_rise : 50.0;
output_threshold_pct_fall : 50.0;
output_threshold_pct_rise : 50.0;
slew_derate_from_library : 0.5;
```

Threshold Specifications and Slew Derating

The above settings specify that the transition times in the library tables have to be multiplied by 0.5 to obtain the transition times which correspond to the slew threshold (30-70) settings.

This means that the values in the transition tables (as well as corresponding index values) are effectively 10-90 values.

```
/* Threshold definitions */
slew_lower_threshold_pct_fall : 30.0;
slew_upper_threshold_pct_fall : 70.0;
slew_lower_threshold_pct_rise : 30.0;
slew_upper_threshold_pct_rise : 70.0;
input_threshold_pct_fall : 50.0;
input_threshold_pct_rise : 50.0;
output_threshold_pct_fall : 50.0;
output_threshold_pct_rise : 50.0;
slew_derate_from_library : 0.5;
```

During characterization, the transition is measured at 30-70 and the transition data in the library corresponds to extrapolation of measured values to 10% to 90% ((70 - 30)/(90 - 10) = 0.5).

Threshold Specifications and Slew Derating

Another example with a different set of slew threshold settings may contain:

```
/* Threshold definitions 20/80/1 */
slew_lower_threshold_pct_fall : 20.0;
slew_upper_threshold_pct_fall : 80.0;
slew_lower_threshold_pct_rise : 20.0;
slew_upper_threshold_pct_rise : 80.0;
/* slew_derate_from_library not specified */
```

- In this example of 20-80 slew threshold settings, there is no slew_derate_from_library specified (implies a default of 1.0), which means that the transition time data in the library is not derated.
- The values in the transition tables correspond directly to the 20-80 characterized slew values.

Threshold Specifications and Slew Derating

Here is another example of slew threshold settings in a cell library.

```
slew_lower_threshold_pct_rise : 20.00;
slew_upper_threshold_pct_rise : 80.00;
slew_lower_threshold_pct_fall : 20.00;
slew_upper_threshold_pct_fall : 80.00;
slew_derate_from_library : 0.6;
```

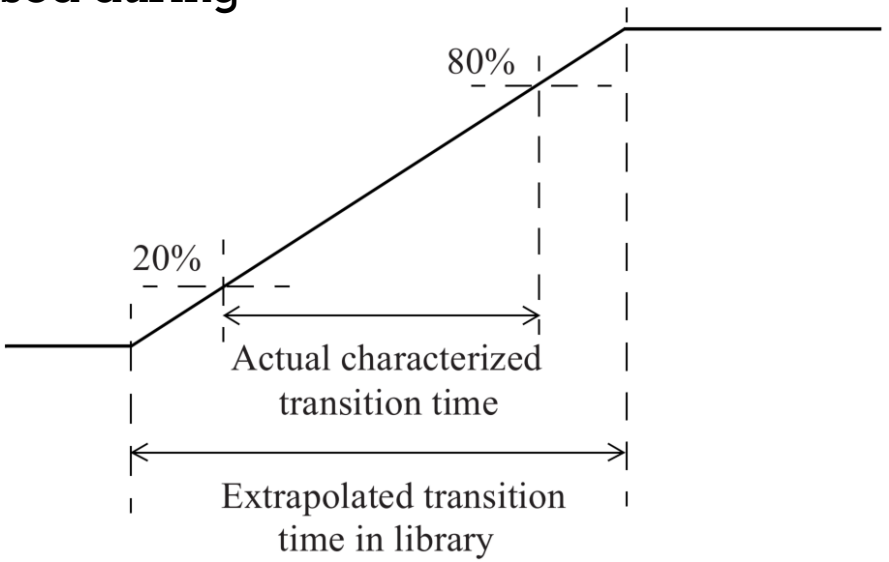
In this case, the slew_derate_from_library is set to 0.6 and characterization slew trip points are specified as 20% and 80%.

This implies that transition table data in the library corresponds to 0% to 100% ((80 - 20)/(100 - 0) = 0.6) extrapolated values.

Threshold Specifications and Slew Derating

When slew derating is specified, the slew value internally used during delay calculation is:

library_transition_time_value * slew_derate



This is the slew used internally by the delay calculation tool and corresponds to the characterized slew threshold measurement points.

CONTENT

Part 3: Standard Cell Library

01 › Overview of Synopsys Timing Lib

02 › Non-Linear Delay Model

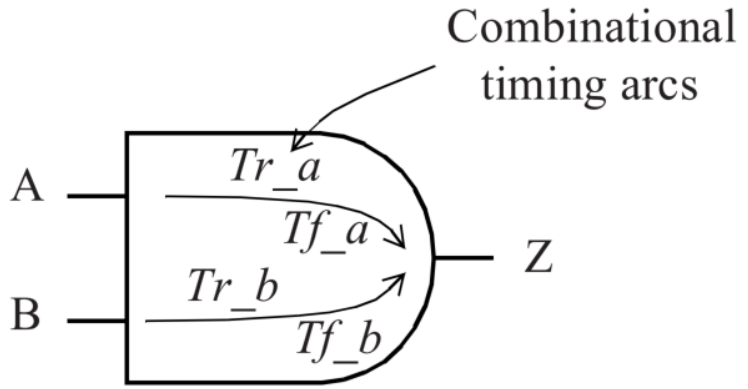
03 › Threshold Specifications and Slew Derating

04 › **Timing Models**

05 › Wire Delay Models

Timing Models - Combinational Cells

Let us consider the timing arcs for a two-input and cell. Both the timing arcs for this cell are positive_unate; therefore an input pin rise corresponds to an output rise and vice versa.

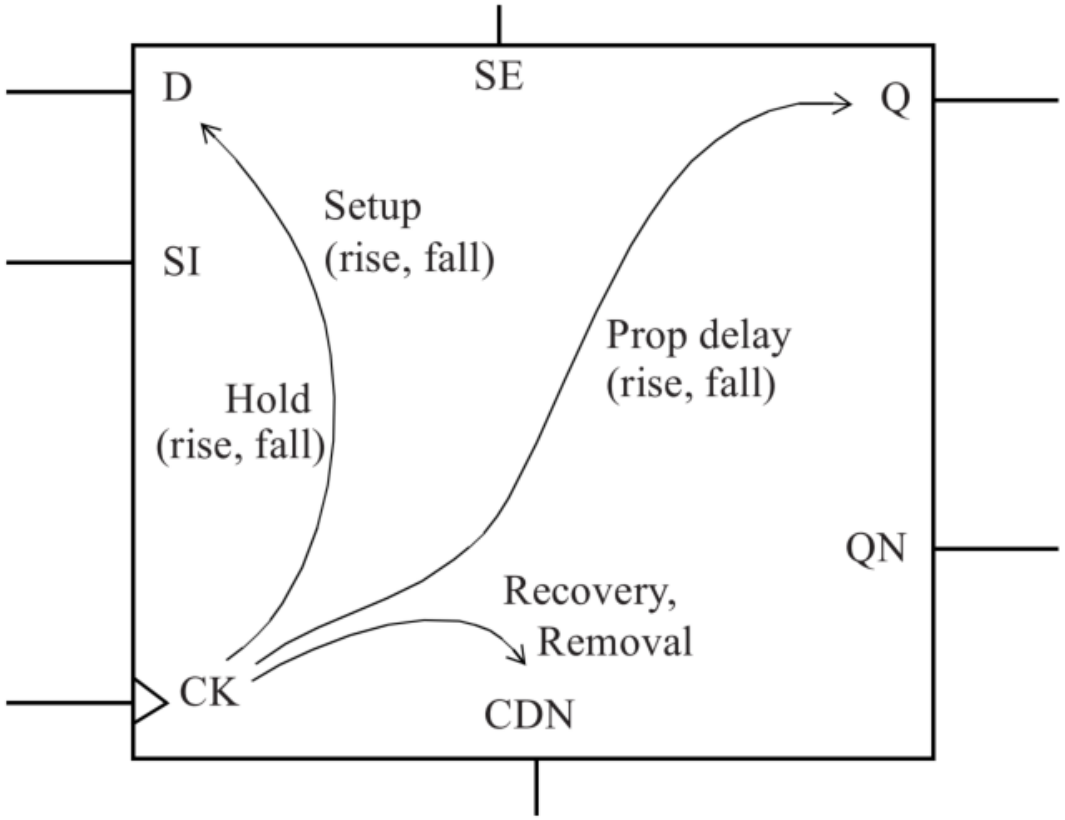


This implies that for the NLDM model, there would be four table models for specifying delays. Similarly, there would be four such table models for specifying the output transition times as well.

Timing Models - Combinational Cells

```
pin (OUT) {
  max_transition : 1.0;
  timing() {
    related_pin : "INP1";
    timing_sense : negative_unate;
    cell_rise(delay_template_3x3) {
      index_1 ("0.1, 0.3, 0.7");
      index_2 ("0.16, 0.35, 1.43");
      values ( \
        "0.0513, 0.1537, 0.5280", \
        "0.1018, 0.2327, 0.6476", \
        "0.1334, 0.2973, 0.7252");
      }
    rise_transition(delay_template_3x3) {
      index_1 ("0.1, 0.3, 0.7");
      index_2 ("0.16, 0.35, 1.43");
      values ( \
        "0.0417, 0.1337, 0.4680", \
        "0.0718, 0.1827, 0.5676", \
        "0.1034, 0.2173, 0.6452");
    }
    cell_fall(delay_template_3x3) {
      index_1 ("0.1, 0.3, 0.7");
      index_2 ("0.16, 0.35, 1.43");
      values ( \
        "0.0617, 0.1537, 0.5280", \
        "0.0918, 0.2027, 0.5676", \
        "0.1034, 0.2273, 0.6452");
    }
    fall_transition(delay_template_3x3) {
      index_1 ("0.1, 0.3, 0.7");
      index_2 ("0.16, 0.35, 1.43");
      values ( \
        "0.0817, 0.1937, 0.7280", \
        "0.1018, 0.2327, 0.7676", \
        "0.1334, 0.2973, 0.8452");
```

Timing Models - Sequential Cells



Timing Models - Sequential Cells

```
pin (D) {
  direction : input;
  . . .
  timing () {
    related_pin : "CK";
    timing_type : "setup_rising";
    rise_constraint ("setuphold_template_3x3") {
      index_1("0.4, 0.57, 0.84"); /* Data transition */
      index_2("0.4, 0.57, 0.84"); /* Clock transition */
      values( /*    0.4      0.57     0.84 */ \
        /* 0.4 */   "0.063,  0.093,   0.112", \
        /* 0.57 */  "0.526,  0.644,   0.824", \
        /* 0.84 */  "0.720,  0.839,   0.930");
    }
    fall_constraint ("setuphold_template_3x3") {
      index_1("0.4, 0.57, 0.84"); /* Data transition */
      index_2("0.4, 0.57, 0.84"); /* Clock transition */
      values( /*    0.4      0.57     0.84 */ \
        /* 0.4 */   "0.762,  0.895,   0.969", \
        /* 0.57 */  "0.804,  0.952,   0.166", \
        /* 0.84 */  "0.159,  0.170,   0.245");
    }
  }
}
```

Timing Models - Sequential Cells

```
timing () {
  related_pin : "CK";
  timing_type : "hold_rising";
  rise_constraint ("setuphold_template_3x3") {
    index_1("0.4, 0.57, 0.84"); /* Data transition */
    index_2("0.4, 0.57, 0.84"); /* Clock transition */
    values(   /*    0.4      0.57     0.84 */ \
      /* 0.4 */   "-0.220,  -0.339,  -0.584", \
      /* 0.57 */  "-0.247,  -0.381,  -0.729", \
      /* 0.84 */  "-0.398,  -0.516,  -0.864");
  }
```

```
  fall_constraint ("setuphold_template_3x3") {
    index_1("0.4, 0.57, 0.84"); /* Data transition */
    index_2("0.4, 0.57, 0.84");/* Clock transition */
    values(   /*    0.4          0.57     0.84 */ \
      /* 0.4 */     "-0.028,    -0.397,  -0.489", \
      /* 0.57 */    "-0.408,    -0.527,  -0.649", \
      /* 0.84 */    "-0.705,    -0.839,  -0.580");
  }
}
```

Timing Models - Sequential Cells

- The propagation delay of a sequential cell is from the active edge of the clock to a rising or falling edge on the output.
- Here is an example of a propagation delay arc for a negative edge-triggered flip-flop, from clock pin CKN to output Q.
- This is a non-unate timing arc as the active edge of the clock can cause either a rising or a falling edge on the output Q.

```
timing() {
  related_pin : "CKN";
  timing_type : falling_edge;
  timing_sense : non_unate;
  cell_rise(delay_template_3x3) {
    index_1 ("0.1, 0.3, 0.7"); /* Clock transition */
    index_2 ("0.16, 0.35, 1.43"); /* Output capacitance */
    values ( /*    0.16    0.35    1.43 */ \
      /* 0.1 */   "0.0513,  0.1537,  0.5280", \
      /* 0.3 */   "0.1018,  0.2327,  0.6476", \
      /* 0.7 */   "0.1334,  0.2973,  0.7252");
  }
```

```
  rise_transition(delay_template_3x3) {
    index_1 ("0.1, 0.3, 0.7");
    index_2 ("0.16, 0.35, 1.43");
    values ( \
      "0.0417, 0.1337, 0.4680", \
      "0.0718, 0.1827, 0.5676", \
      "0.1034, 0.2173, 0.6452");
  }
  cell_fall(delay_template_3x3) {
    index_1 ("0.1, 0.3, 0.7");
    index_2 ("0.16, 0.35, 1.43");
    values ( \
      "0.0617, 0.1537, 0.5280", \
      "0.0918, 0.2027, 0.5676", \
      "0.1034, 0.2273, 0.6452");
  }
  fall_transition(delay_template_3x3) {
    index_1 ("0.1, 0.3, 0.7");
    index_2 ("0.16, 0.35, 1.43");
    values ( \
      "0.0817, 0.1937, 0.7280", \
      "0.1018, 0.2327, 0.7676", \
      "0.1334, 0.2973, 0.8452");
  }
}
```

CONTENT

Part 3: Standard Cell Library

01 › Overview of Synopsys Timing Lib

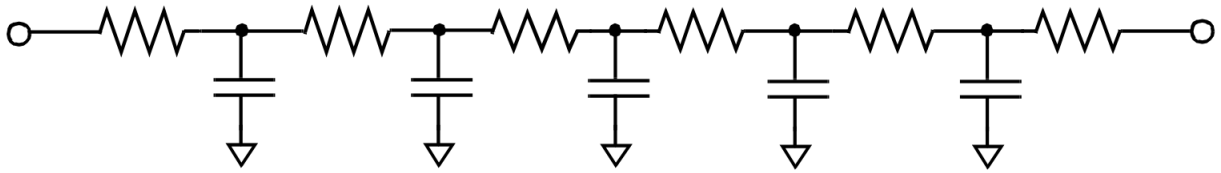
02 › Non-Linear Delay Model

03 › Threshold Specifications and Slew Derating

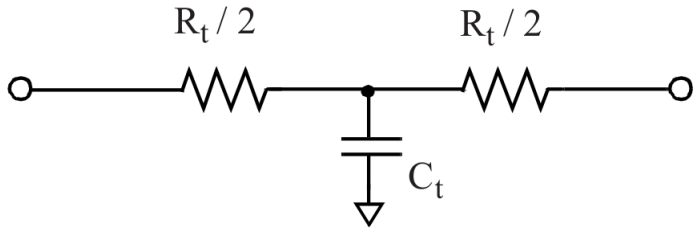
04 › Timing Models

05 › **Wireload Models**

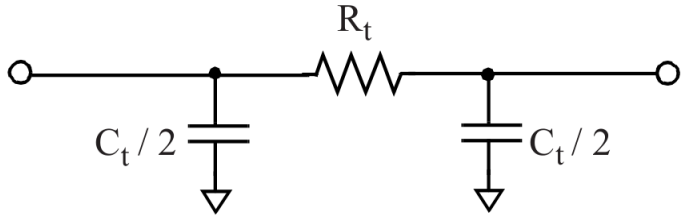
Wireload Models



Distributed RC tree



T-model representation

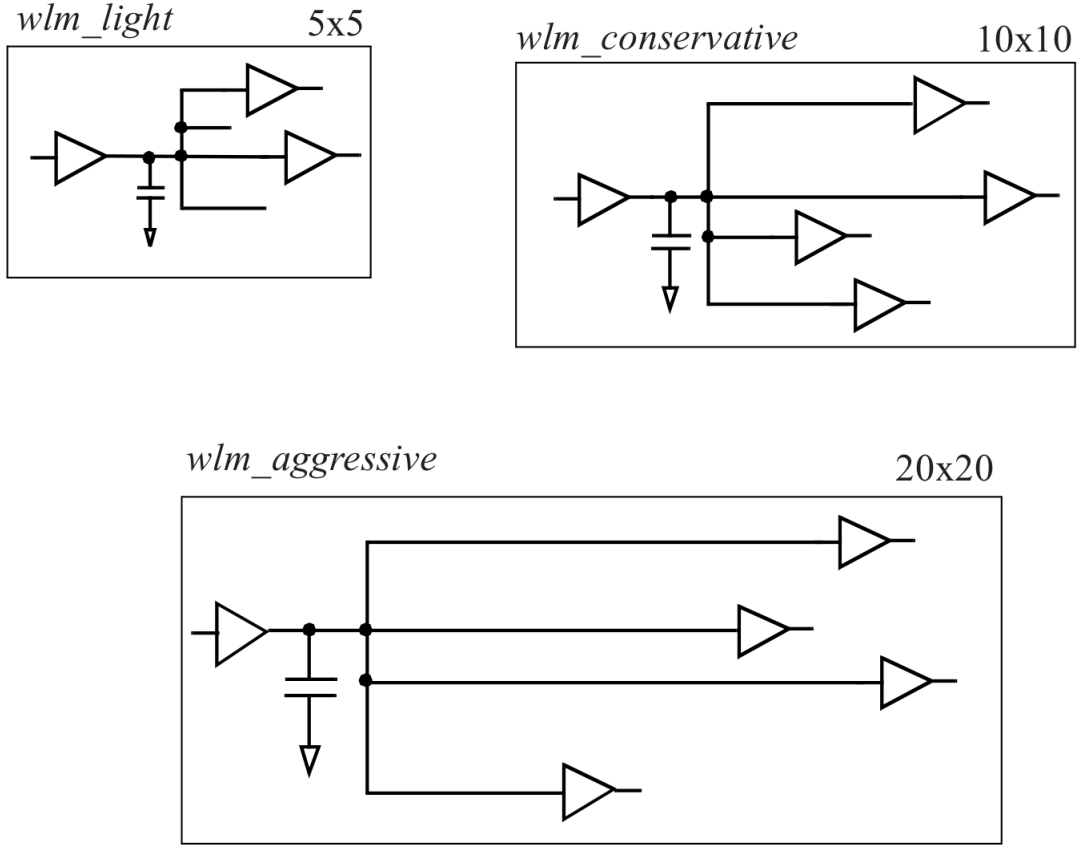


Pi-model representation

Wireload Models

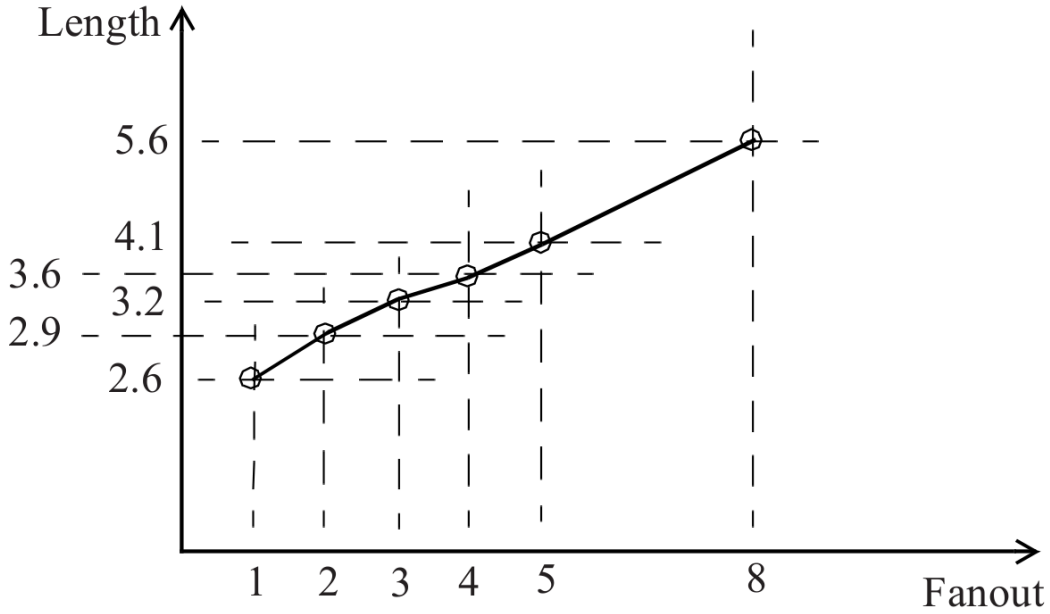
Here is an example of a wireload model

```
wire_load ("wlm_conservative")
  resistance : 5.0;
  capacitance : 1.1;
  area : 0.05;
  slope : 0.5;
  fanout_length (1, 2.6);
  fanout_length (2, 2.9);
  fanout_length (3, 3.2);
  fanout_length (4, 3.6);
  fanout_length (5, 4.1);
}
```



Different wireload models for different areas

Wireload Models



Fanout vs wire length

```
wire_load ("wlm_conservative")
  resistance : 5.0;
  capacitance : 1.1;
  area : 0.05;
  slope : 0.5;
  fanout_length (1, 2.6);
  fanout_length (2, 2.9);
  fanout_length (3, 3.2);
  fanout_length (4, 3.6);
  fanout_length (5, 4.1);
}
```

- Length = 4.1 + (8 - 5) * 0.5 = 5.6 units
- Capacitance = Length * cap_coeff(1.1) = 6.16 units
- Resistance = Length * res_coeff(5.0) = 28.0 units
- Area overhead due to interconnect = Length * area_coeff(0.05)= 0.28 area units

参考书目

- Static Timing Analysis for Nanometer Designs: A Practical Approach. J. Bhasker, Rakesh Chadha. Springer Science Business Media, LLC 2009. Chaper 3 & 4.
- 集成电路静态时序分析与建模. 刘峰，机械工业出版社.出版时间：2016-07-01. 第四章.

谢谢聆听！

个人教学工作主页https://customizablecomputinglab.github.io/