芯动力——硬件加速设计方法

第四章 逻辑综合(2)

邸志雄@西南交通大学

zxdi@home.swjtu.edu.cn

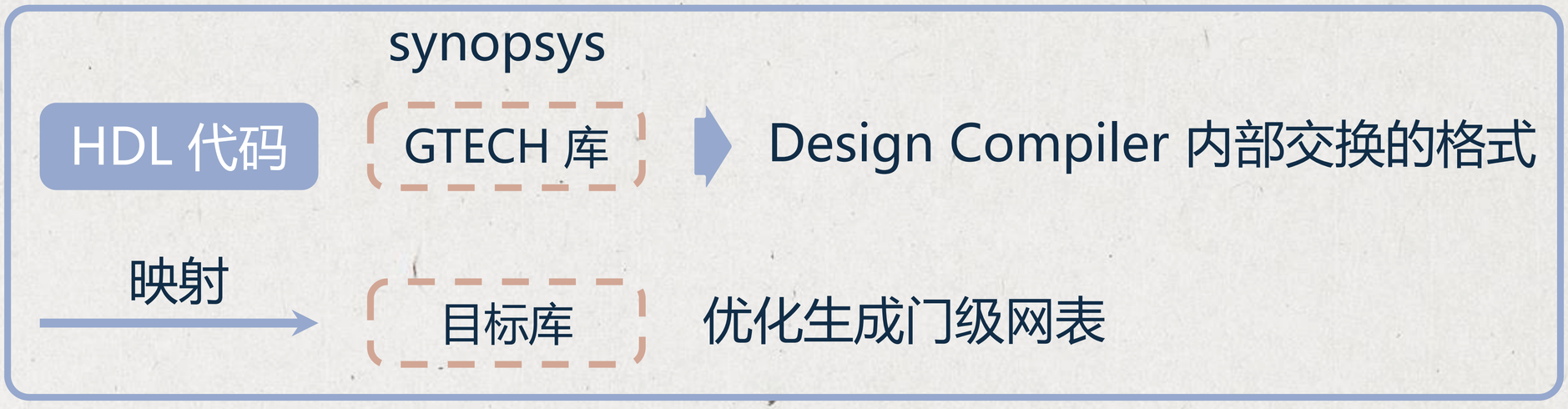
库文件的设置

- 在 Design Compiler 的运行过程中需要用到几种库文件：

目标库	target_library
链接库	link_library
符号库	symbol_library
算术运算库	synthetic_library

目标库(target_library)

- 目标库是综合后电路网表要最终映射到的库



目标库　Foundary　.db 的格式　文本格式的.lib

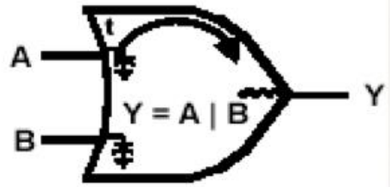
- 这种格式是 DC 认识的一种内部文件格式，不能由文本方式打开

Example of a cell description in .lib Format

```
cell ( OR2_3 ) {
    area : 8.000 ;
    pin ( Y ) {
        direction : output;
        timing ( ) {
            related_pin : "A" ;
            timing_sense : positive_unate ;
            rise_propagation (drive_3_table_1) {
                values ("0.2616, 0.2608, 0.2831,..)
            }
            rise_transition (drive_3_table_2) {
              values ("0.0223, 0.0254, ...)
                . . . .
        function : "(A | B)";
        max_capacitance :  1.14810 ;
        min_capacitance :  0.00220 ;
    }
    pin ( A ) {
        direction : input;
        capacitance : 0.012000;
                . . . .
```

Cell name

Cell Area



Pin A -> Pin Y nominal delays (look-up table)

Pin Y functionality

Design Rules for Output Pin

Electrical Characteristics of Input Pins

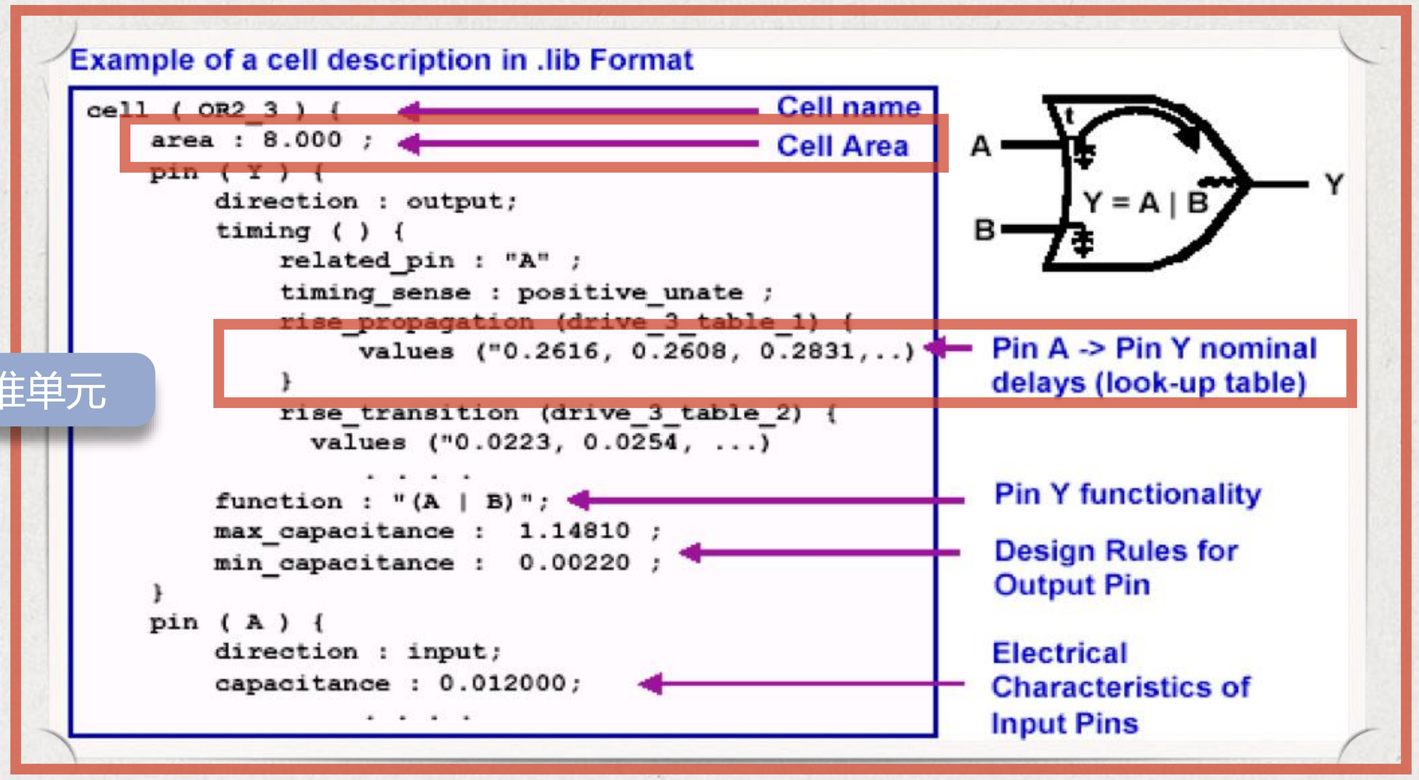
工艺库文件(.lib)

set target_library my_tech.db

目标库 各个门级单元 行为 引脚 面积 时序信息 功耗方面的参数

- DC在综合时就是根据target_library中给出的单元电路的延迟信息来计算路径的延迟。并根据各个单元延时、面积和驱动能力的不同选择合适的单元来优化电路。

取或的标准单元



工艺库文件(.lib)

set target_library my_tech.db

经常使用的方法

- 在逻辑功能相同的前提下替换单元的大小尺寸

替换的依据

cell单元 延迟 面积 功耗

链接库(link_library)

- link_library 设置模块或者单元电路的引用
- 对于所有 DC 可能用到的库，我们都需要在 link_library 中指定，其中也包括要用到的 IP

链接库和目标库区别:

- 链接库对应IP

  购买的付费IP

  存储器

  IO

  PAD

- 目标库更多的指的是标准单元

链接库(link_library)

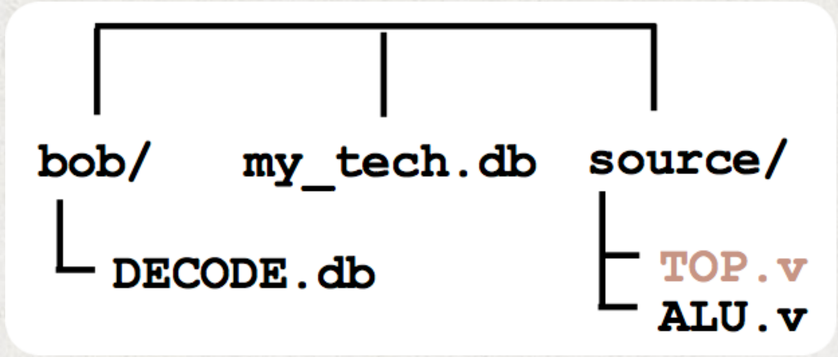
- link_library 设置模块或者单元电路的引用
- 对于所有 DC 可能用到的库，我们都需 要在 link_library 中指定，其中也包括要用到的 IP。

IP

值得注意的一点:

- 在 link_library 的设置中必须包含 “*” *
  - 表示 DC 在引用实例化模块或者单元电路时首先搜索已经调进 DC memory 的模块和单元电路
- 如果在 link library 中不包含 *
  - DC 就不会使用 DC memory 中已有的模块，因此，会出现无法匹配的模块或单元电路的警告信息(unresolved design reference)

```
set target_library  " my_tech.db"
set link_library  "* my_tech.db"
```



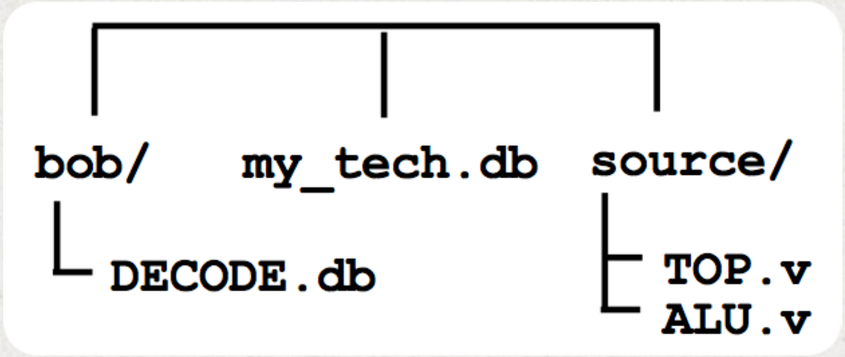
TOP.v

```
module TOP (A,B,OUT1);
input A, B;
output [1:0] OUT1;
ALU U1 (.AIN (A), . .
DECODE U2 (.A (BUS0), . .
```

```
dc_shell-t>   read_verilog source/ALU.v
dc_shell-t>   read_verilog source/TOP.v
dc_shell-t>   link
```

Unable to resolve reference ‘DECODE’ in ‘TOP’

```
set target_library  " my_tech.db"
set link_library  "* my_tech.db"
lappend search_path {bob}
```



TOP.v

```
module TOP (A,B,OUT1);
input A, B;
output [1:0] OUT1;
ALU U1 (.AIN (A), . .
DECODE U2 (.A (BUS0), . .
```

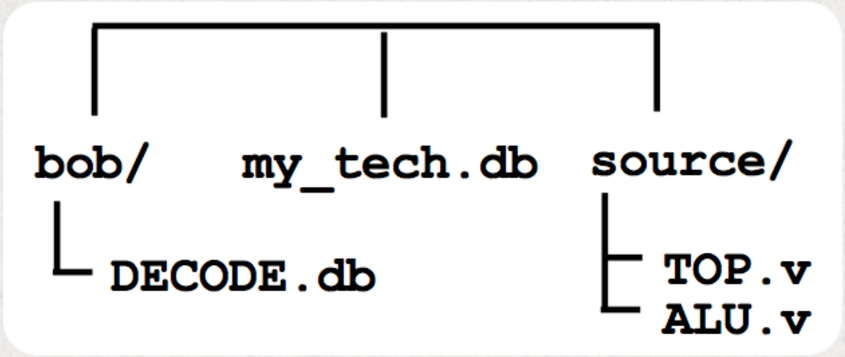
```
dc_shell-t>   read_verilog source/ALU.v
dc_shell-t>   read_verilog source/TOP.v
dc_shell-t>   link
```

Loading db file bob/DECODE.db

```
dc_shell-t>   which DECODE.db
```

/server/my_project/bob/DECODE.db

```
set target_library  " my_tech.db"
set link_library  "* my_tech.db"
lappend search_path {bob}
```



TOP.v

```
module TOP (A,B,OUT1);
input A, B;
output [1:0] OUT1;
ALU U1 (.AIN (A), . .
DECODE U2 (.A (BUS0), . .
```

```
dc_shell-t>   analyze  -f  vhdl  source/ALU.vhd
dc_shell-t>   analyze  -f  vhdl  source/TOP.vhd
dc_shell-t>   elaborate TOP
Loading db file bob/DECODE.db
 Current design is now  ' TOP'
```

符号库 (symbol_library)

- symbol_library 是定义了单元电路显示的 Schematic 的库

design_analyzer

design_vision

查看、分析电路时

symbol_library

符号库 .sdb

- 加入没有设置，DC 会用默认的符号库取代

设置符号库的命令

set symbol_library

算术运算库(synthetic_library)

在初始化 DC 的时候

- 不需要设置标准的 DesignWare 库 standard.sldb 用于实现 Verilog 描述的运算符

加法　乘法 ➡ 默认综合为一些性能相对比较差的电路结构

加法

串行进位的加法器

DesignWare 库

超前进位进位加法器

算术运算库

算术运算库(synthetic_library)

高级的license

在初始化 DC 的时候

- 不需要设置标准的 DesignWare 库 standard.sldb 用于实现 Verilog 描述的运算符

对于扩展的 DesignWare

- 需要在 synthetic_library 中设置，同时需要在 link_library 中设置相应的库以使得在链接的时候 DC 可以搜索到相应运算符的实现。

工艺库：单元时序信息

单元时序模型

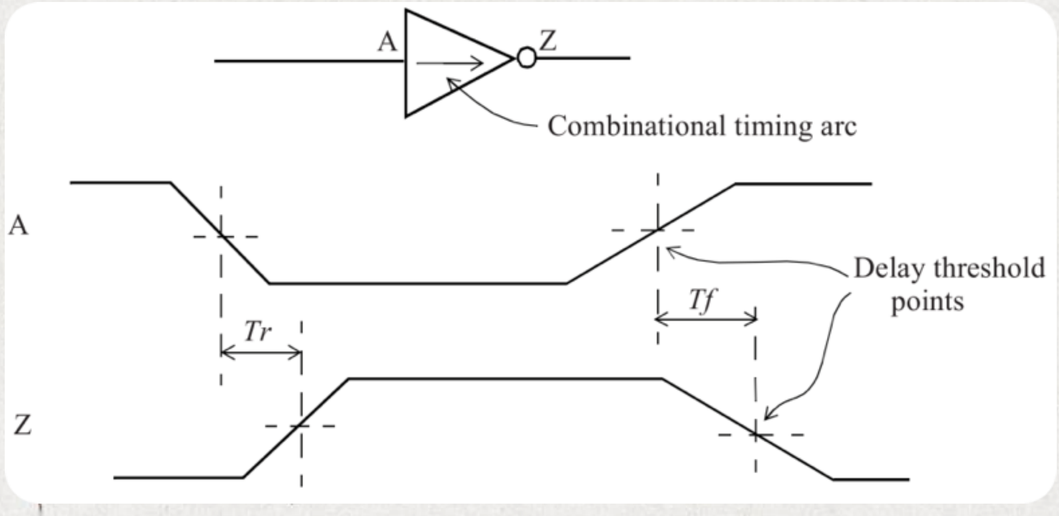
- 旨在为设计环境中的单元的各种实例提供精确的时序

时序模型 → 单元的详细电路模拟

↓

模拟单元操作的实际情况

简单的反相器逻辑单元的timing arc



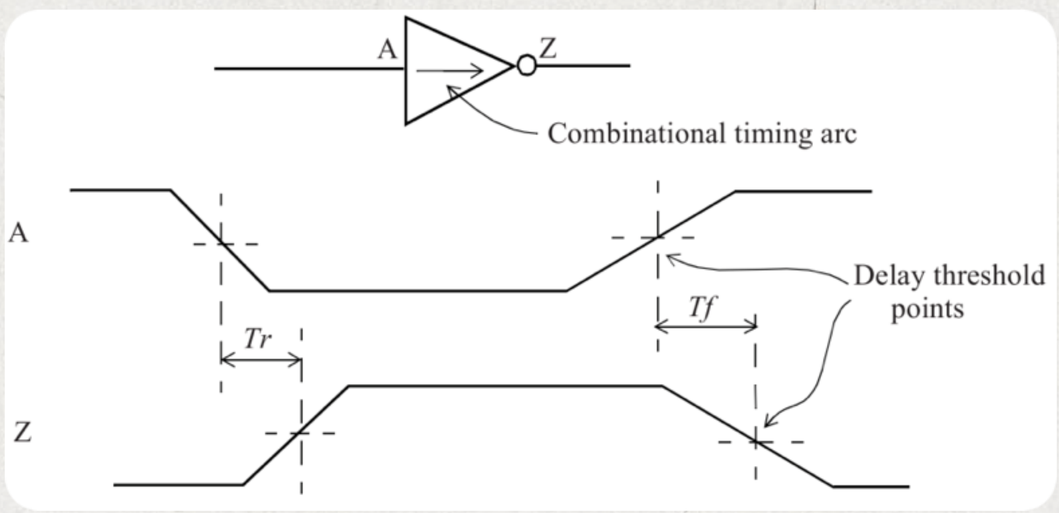
- 输入端的上升（下降）转换会导致输出端的下降（上升）转换

以单元为特征的两种延迟:

T_r 输出上升延迟

T_f 输出下降延迟

延迟是根据单元库中定义的阈值点测量的，通常为50% VDD



Timing arc的延迟取决于两个因素:

- 输出负载，即反相器输出引脚的电容负载
- 输入信号的转换时间

延迟值与负载电容直接相关： 负载电容越大，延迟越大

在大多数情况下，延迟随着输入转换时间的增加而增加。

线性时序模型

- 简单的时序模型是线性延迟模型，其中单元的延迟和输出转换时间表示为两个参数的线性函数：输入转换时间和输出负载电容。

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

D0 D1 D2 - 常数

S - 输入转换时间

C - 输出负载电容

线性延迟模型在亚微米技术的输入转换时间和输出电容范围内不准确

非线性时序模型

- 延迟的非线性时序模型（NLDM）以二维形式呈现

输入转换时间

输出负载电容

二维表

```
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");
    }
```

延迟

非线性时序模型

- 单元描述的这一部分包含从引脚INP1到引脚OUT的timing arc的上升和下降延迟模型，以及引脚OUT上的max_transition允许时间。

上升延迟模型

cell_rise

下降延迟模型

cell_fall

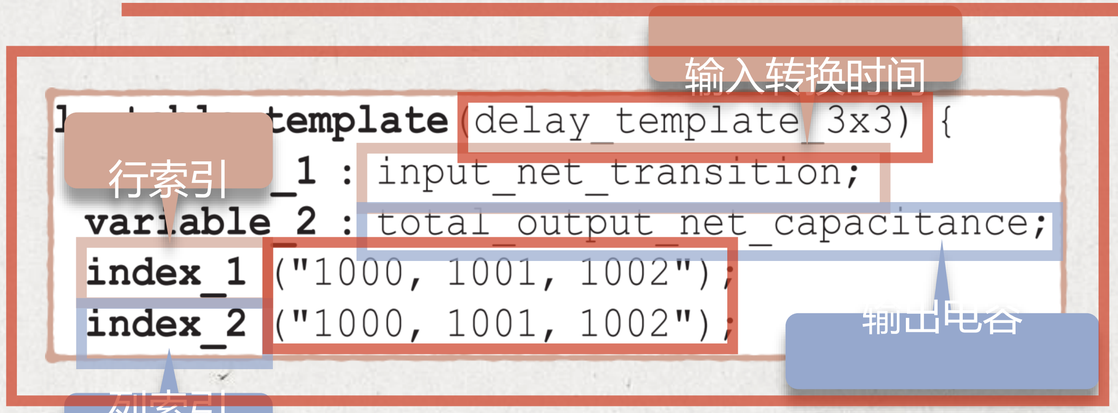
```
pin (OUT) {
  max_transition : 1.0;
  timing() {
    related_pin : "INP1";
    timing_sense : negative_unate;
    cell_rise(delay_template_3x3) {
      index_1 ("0.1, 0.3, 0.7"); /* Input transition */
      index_2 ("0.16, 0.35, 1.43"); /* Output capacitance */
      values (  /*   0.16      0.35      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");
    }
```

非线性时序模型

- 索引的类型和表查找索引的顺序在查找表模板delay_template_3x3中描述。

```
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");
    }
```

- 此查找表模板指定表中的第一个变量是输入转换时间，第二个变量是输出电容。
- 每个变量有三个条目，因此它对应于一个3乘3的表。
- 在大多数情况下，表的条目也像表一样格式化，然后第一个索引（index_1）可以被视为行索引，第二个索引（index_2）变得等于列索引。索引值（例如1000）是虚拟占位符，它们被cell_fall和cell_rise延迟表中的实际索引值覆盖。




```
lu_table_template(delay_template_3x3) {
  variable_1 : input_net_transition;
  variable_2 : total_output_net_capacitance;
  index_1 ("0.1, 0.3, 0.7");
  index_2 ("0.16, 0.35, 1.43");
}
```

另一种方法是在模板定义中指定索引值，而不在cell_rise和cell_fall表中指定它们。

Threshold Specifications and Slew Derating

Slew值基于库中指定的测量阈值

大多数库（0.25或更早版本）

- 使用10%和90%作为转换或转换时间的测量阈值。选择转换阈值以对应于波形的线性部分。

- 随着技术变得更精细，实际波形最线性的部分通常在30%和70%之间。

大多数新一代时序库

- 将转换测量点指定为Vdd的30%和70%。

- 由于转换时间之前的测量值介于10%和90%之间，因此在填充库时，测量的转换时间通常会增加一倍，介于30%和70%之间。

转换Derate系数

- 通常指定为0.5。阈值为30%和70%
- Slew Derating降低为0.5，导致等效测量点为10%和90%

```
/* 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 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;
```

- 转换时间必须乘以0.5，以获得与转换阈值（30-70）设置相对应的转换时间。
- 这意味着转换表中的值（以及相应的索引值）实际上是10-90值。
- 在表征期间，转变在30-70处测量，并且库中的转变数据对应于测量值的外推至10%至90%（（70-30）/（90-10）= 0.5）。

另一组具有不同的转换阈值设置的示例可能包含:

```
/* 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 */
```

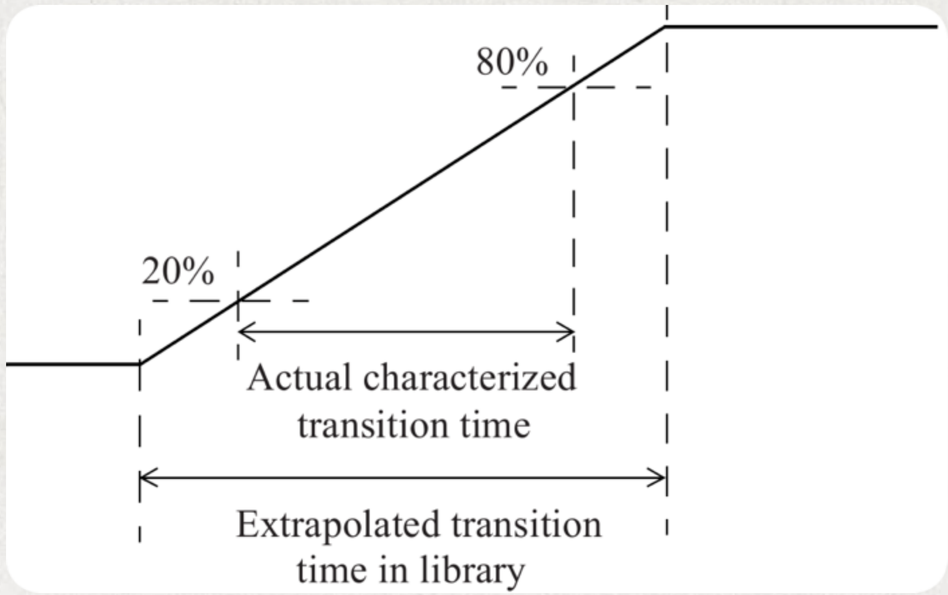
- 在此示例中，20-80转换阈值设置，未指定slew_derate_from_library（默认值为1.0），这意味着库中的转换时间数据未降低额定值。转换表中的值直接对应于20-80特征转换值。

以下是单元库中转换阈值设置的另一个示例:

```
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;
```

- 在这种情况下，slew_derate_from_library设置为0.6，表征转换点指定为20%和80%。这意味着库中的转换表数据对应于0%至100%（（80-20）/（100-0）= 0.6）外推值。

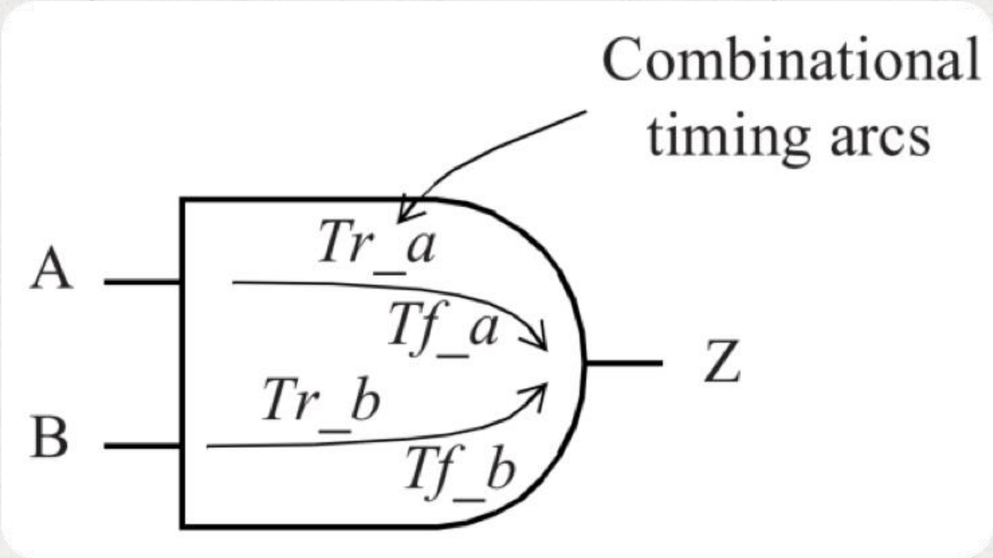
非线性时序模型



- 这是延迟计算工具内部使用的转换，对应于特征化的转换阈值测量点。
- 当指定转换Derate系数时，延迟计算期间内部使用的transition_time为：

library_transition_time_value * slew_derate

时序模型 - 组合单元

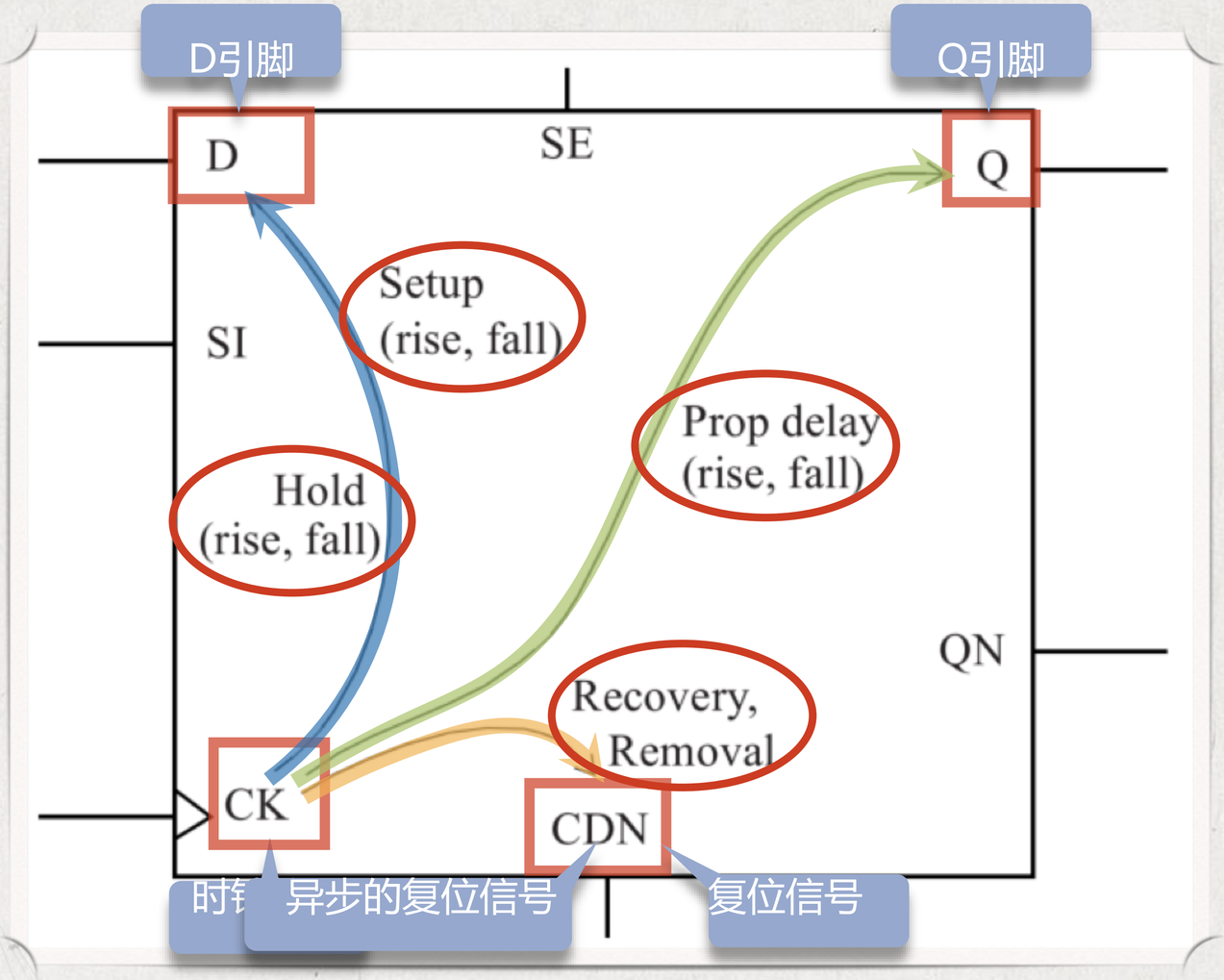


- 让我们考虑双输入和单元的时序弧。这个单元的两个时序弧都是positive_unate;因此输入引脚上升对应于输出上升，反之亦然。
- 这意味着对于NLDM模型，将有四个用于指定延迟的表模型。类似地，将有四个这样的表模型用于指定输出转换时间。

时序模型 - 组合单元

```
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");
```

时序模型 - 时序单元


D引脚
Q引脚
D
SE
Q
Setup
(rise, fall)
SI
Prop delay
(rise, fall)
Hold
(rise, fall)
QN
Recovery,
Removal
CK
CDN
异步的复位信号
复位信号

```
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 () {
  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");
  }
}
```

- 时序单元的传播延迟是从时钟的有效边沿到输出的上升沿或下降沿。下面是负边沿触发触发器的传播延迟弧示例，从时钟引脚CKN到输出Q.这是一个非整数的timing arc，因为时钟的有效边沿可能导致上升或下降输出Q的边缘。这是延迟表：

```
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");
  }
}
```