**静态时序分析圣经翻译计划——附录B：SDF（上）**

[](https://www.zhihu.com/people/zhao-jun-jun-19)

[**赵俊军**](https://www.zhihu.com/people/zhao-jun-jun-19)

[**​**](https://www.zhihu.com/question/48510028)

上海交通大学 电子与通信工程硕士在读

18 人赞同了该文章

本附录将介绍标准延迟标注格式，并说明了如何在仿真中执行反标。

延迟格式描述了设计网表的单元延迟和互连走线延迟，无论设计是用两种主要硬件描述语言（VHDL或Verilog HDL）中的哪一种所描述的。

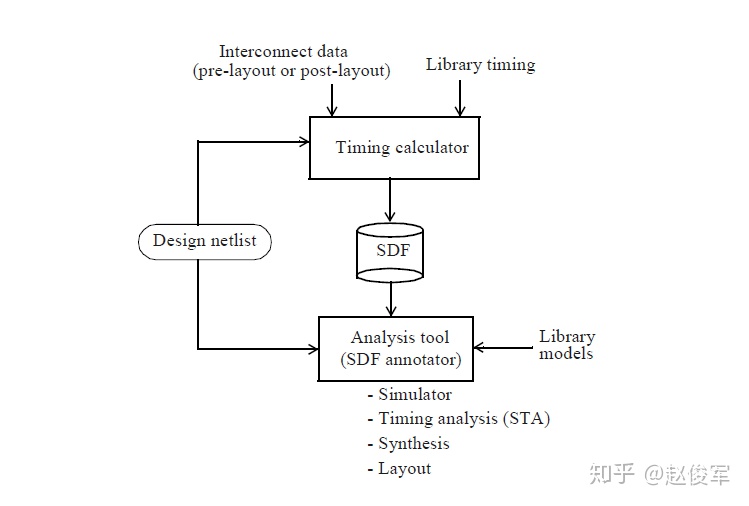
本章还会介绍仿真的反标（backannotation），STA的反标其实是一个简单直接的过程，其中DUA中的时序弧将由SDF所指定的延迟进行标注。

**B.1 什么是SDF？**

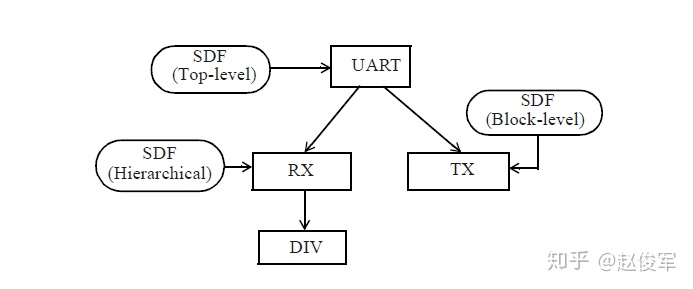
SDF是指标准延迟格式（Standard Delay Format）。它是一个IEEE标准——IEEE Std1497，它是ASCII文本文件，它描述了时序信息和约束，其目的是用作各种工具之间的文本类型的时序信息交换媒介，它也可以用来描述需要它的工具的时序数据。由于它是IEEE标准，因此由一个工具生成的时序信息可以被支持该标准的许多其它工具所使用。SDF中的数据与工具和语言都无关，且包括了互连走线延迟、器件延迟以及时序检查的规范。

由于SDF是ASCII文件，因此它易于阅读，尽管对于实际设计而言，这些文件往往很大。但是，它是作为工具之间的交换媒介。经常在进行信息交换时，一个工具可能会在生成SDF文件时产生一个问题，而另一个读取SDF的工具可能无法正确读取SDF。读取SDF的工具可能会在读取SDF时产生一个错误或警告，或者它可能会错误地解释SDF中的值。在这种情况下，设计人员可能必须查看SDF文件，看看出了什么问题。本章介绍了SDF文件的基础知识，并提供了必要和足够的信息，以帮助理解和调试任何标注问题。

图B-1显示了如何使用SDF文件的典型流程。时序计算工具通常会生成时序信息存储在SDF文件中。然后，通过读取SDF的工具将该信息反标到设计中。请注意，完整的设计信息不会都存储到SDF文件中，而只会存储延迟值。例如，实例名称和实例的引脚名称将被存储到SDF文件中，因为它们对于指定实例相关或引脚相关的延迟是必需的。因此，必须为SDF生成工具和SDF读取工具提供相同的设计。

图B-1

一个设计可以具有多个与之关联的SDF文件。可以为一个设计创建一个SDF文件，在分层设计中，也可以为分层中的每个块创建多个SDF文件。在标注期间，每个SDF都将应用于适当的分层实例中，如图B-2所示。

图B-2

SDF文件包含了用于反标和标注的时序数据。更具体地说，它包含：

● 单元延迟（Cell delays）

● 脉冲传播（Pulse propagation）

● 时序检查（Timing checks）

● 互连走线延迟（Interconnect delays）

● 时序环境（Timing environment）

引脚到引脚的延迟（pin-to-pin delay）和分布式延迟（distributed delay）都可以针对单元延迟进行建模。引脚到引脚的延迟使用**IOPATH**结构（construct）表示，这些结构定义了每个单元输入到输出的路径延迟。**COND**结构还可以用于额外指定有条件的引脚到引脚延迟。状态相关（state-dependent）的路径延迟也可以使用**COND**结构来指定，分布式延迟的建模是使用**DEVICE**结构指定的。

脉冲传播结构——**PATHPULSE**和**PATHPULSEPERCENT**可用于指定使用引脚到引脚延迟模型时允许传播到单元输出端口的毛刺大小。

可以在SDF中指定的时序检查包括：

● 建立时间：SETUP，SETUPHOLD

● 保持时间：HOLD，SETUPHOLD

● 恢复时间：RECOVERY，RECREM

● 撤销时间：REMOVAL，RECREM

● 最大偏斜：SKEW，BIDIRECTSKEW

● 最小脉冲宽度：WIDTH

● 最小周期：PERIOD

● 不变化：NOCHANGE

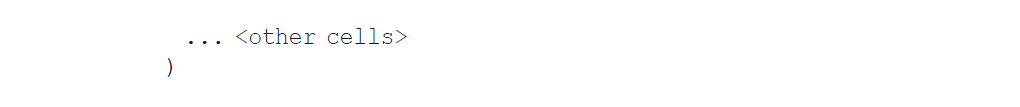
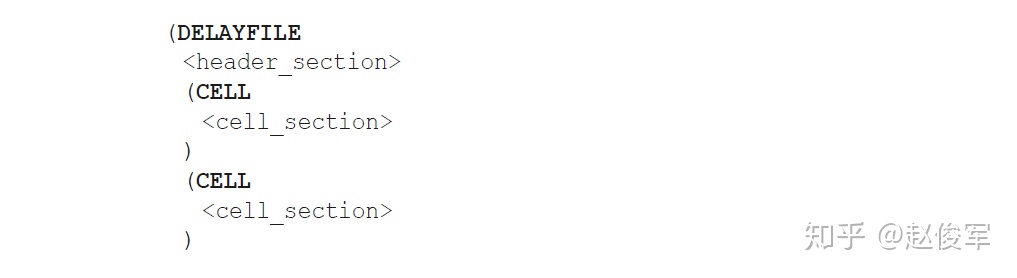
时序检查中的信号可能存在某些条件。在时序检查中允许使用负值，不支持负值的工具可以选择将其替换为零。

SDF描述中支持三种类型的互连走线建模。**INTERCONNECT**结构是最通用且最常用的，可用于指定点对点延迟（从源端到接收端），因此单个网络可以具有多个**INTERCONNECT**结构。**PORT**结构可用于指定负载端口处的网络延迟，假定网络只有一个源端驱动。**NETDELAY**结构可用于指定整个网络的延迟，而无需考虑其源端或接收端，因此是指定网络上延迟的最不具体的方法。

时序环境提供了设计在工作时所依据的信息，这些信息包括**ARRIVAL**，**DEPARTURE**，**SLACK**和**WAVEFORM**结构。这些结构主要用于标注，例如可用于综合。

**B.2 SDF格式**

SDF文件包含一个首部（header section），后跟一个或多个单元。每个单元代表设计中的一个区域或范围，它可以是库原语（primitive）或用户自定义的黑盒。



首部包含一般信息，除了层次结构分隔符、时间刻度（timescale）和SDF版本号外，都不会影响SDF文件的语义。默认情况下，层次结构分隔符**DIVIDER**是点字符（“.”）。通过以下方法，可以将其替换为“/”字符：

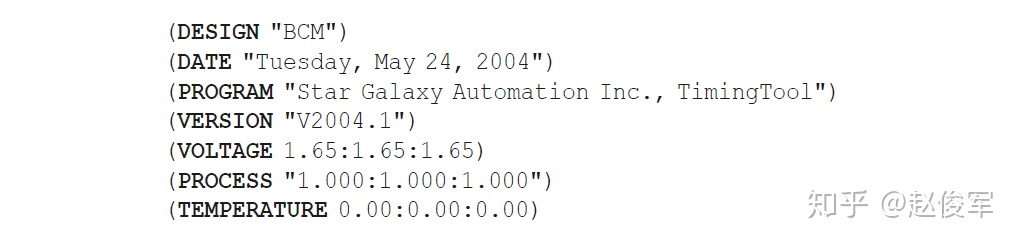
● （**DIVIDER** /）

如果首部中没有时间刻度信息，则默认值为1ns。否则，可以使用以下命令明确指定时间刻度**TIMESCALE**：

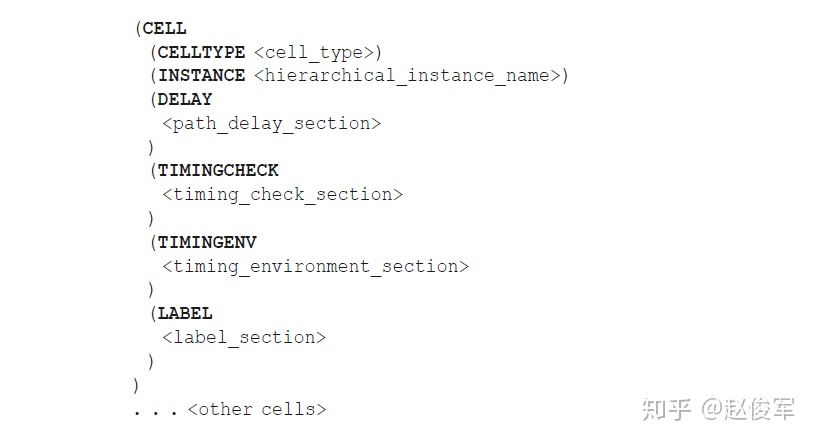
● （**TIMESCALE** 10ps）

也就是说，将SDF文件中指定的所有延迟值乘以10ps。

SDF版本号**SDFVERSION**是必需的，SDF文件的使用者会根据它来确保文件符合指定的SDF版本。首部中可能存在的其它信息（属于常规信息类别）包括日期、程序名称、版本和工作条件。

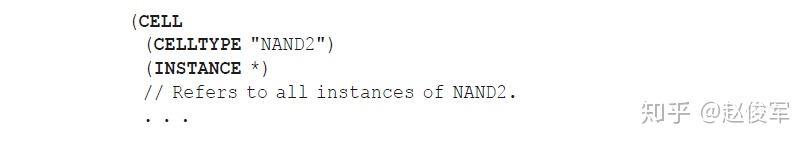


首部之后是一个或多个单元的描述，每个单元在设计中代表一个或多个实例（使用通配符），单元可以是库原语或分层块（hierarchical block）。



单元的顺序很重要，因为数据是从上到下进行处理的。后面的单元描述可以覆盖前面的单元描述所指定的时序信息（通常，两次定义同一单元实例的时序信息并不常见）。另外，可以将时序信息标注为绝对值或增量的形式。如果时序信息使用增量的形式，它将会把新值添加到现有值中。而如果时序信息是绝对值，它将覆盖任何先前指定的时序信息。

单元实例可以是分层实例名称。用于层次结构分隔的分隔符必须符合首部中指定的分隔符。单元实例名称可以选择为“ \* ”字符，即通配符，这表示指定类型的所有单元实例。



单元中可以描述四种类型的时序规范：

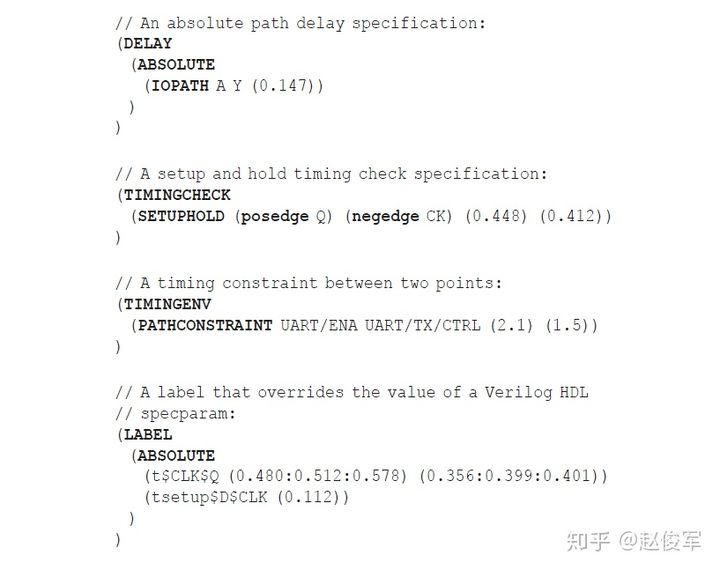
● **DELAY**：用于描述延迟

● **TIMINGCHECK**：用于描述时序检查

● **TIMINGENV**：用于描述时序环境

● **LABEL**：声明可用于描述延迟的时序模型变量。

以下是一些例子：



**DELAY**时序规范有四种类型：

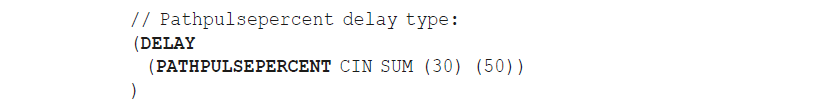
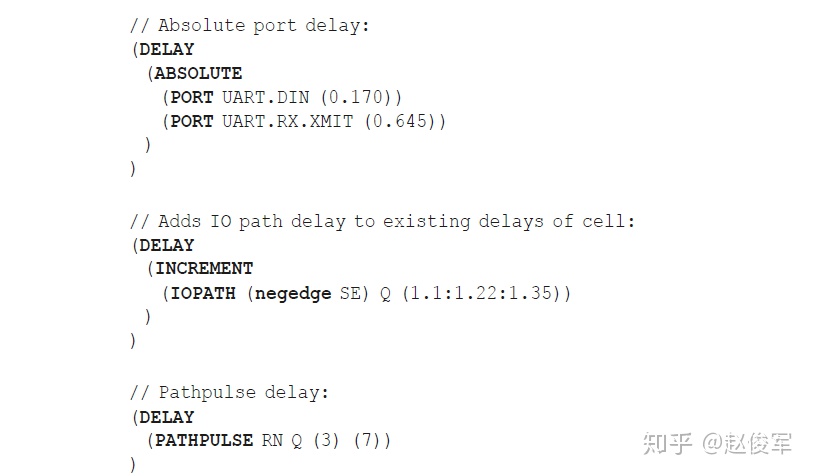
● **ABSOLUTE**：在反标期间替换单元实例的现有延迟值。

● **INCREMETN**：将新的延迟数据添加到单元实例的任何现有延迟值。

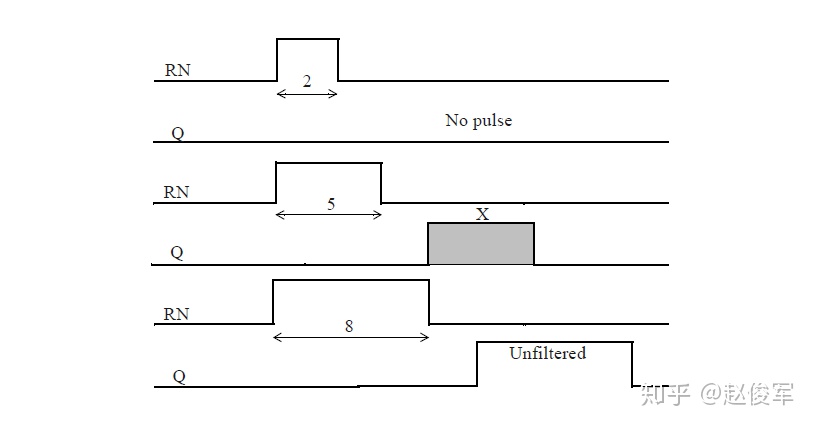
● **PATHPULSE**：指定设计输入和输出之间的脉冲传播极限。此极限值用于决定是将出现在输入上的脉冲传播到输出，还是将其标记为“ X ”，或者将其滤除。

● **PATHPULSEPERCENT**：除了值以百分比表示外，这与**PATHPULSE**完全相同。

以下是一些例子：



RN和Q是单元的输入端口和输出端口。第一个值3是脉冲抑制极限（pulse rejection limit），称为r-limit，它定义了可以出现在输出上的最窄脉冲。窄于此的任何脉冲都会被拒绝通过，也就是说，它将不会出现在输出上。第二个值7（如果存在）是错误极限（error limit），也称为e-limit。任何小于e-limit的脉冲都会导致输出为“ X ”。e-limit必须大于r-limit，如图B-3所示。 当出现小于3（r-limit）的脉冲时，该脉冲不会传播到输出；当脉冲宽度在3（r-limit）和7（e-limit）之间时，输出为X ；当脉冲宽度大于7（e-limit）时，脉冲会传播到输出且没有任何滤除（unfiltered）。

图B-3

可以使用**ABSOLUTE**或**INCREMENT**描述八种延迟定义：

● **IOPATH**：输入到输出路径的延迟。

● **RETAIN**：保留时间定义，可以用于指定输出端口在其相关输入端口改变后应保留其先前值的时间。

● **COND**：条件路径延迟，可以用于指定状态相关的输入到输出路径延迟。

● **CONDELSE**：默认路径延迟，可以用于指定条件路径的默认值。

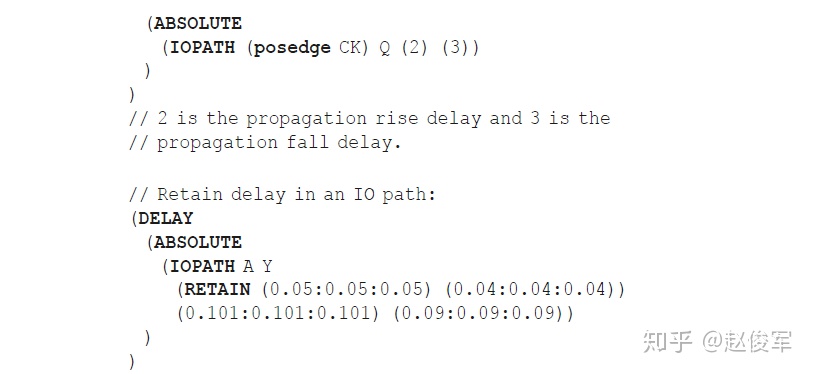
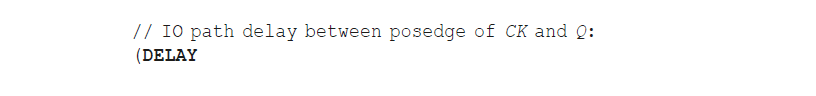
● **PORT**：端口延迟，可以用于指定互连走线延迟，该延迟被建模为输入端口的延迟。

● **INTERCONNECT**：互连走线延迟，可以用于指定从其源端到接收端的整个网络的传播延迟。

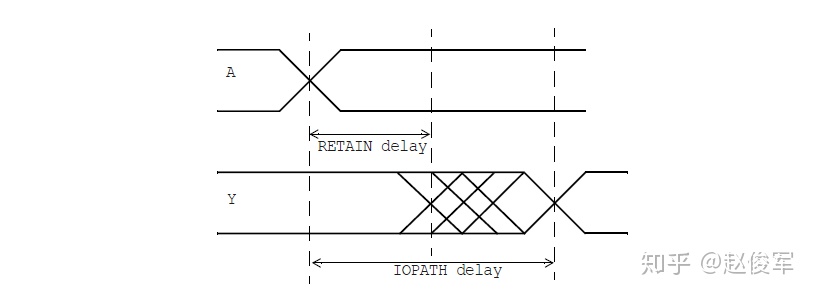
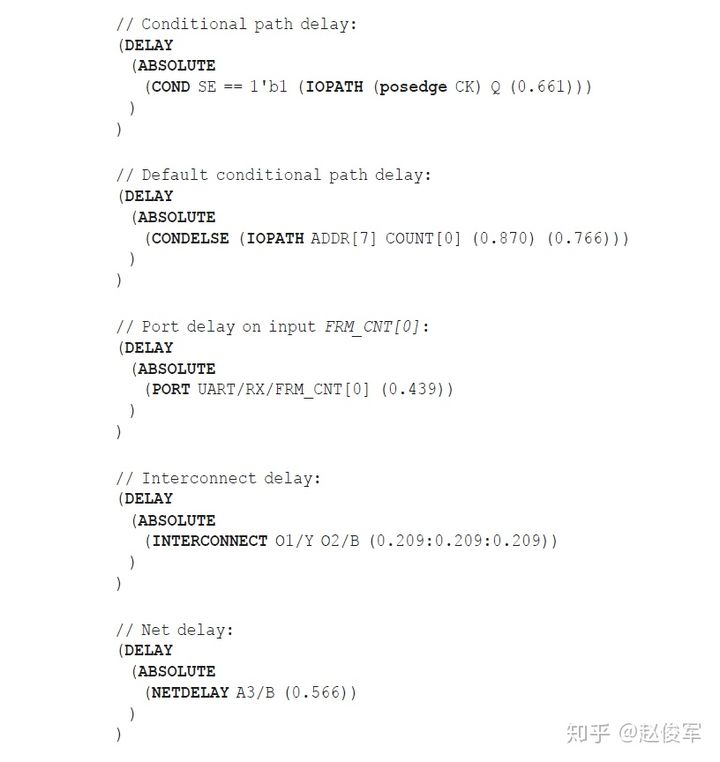
● **NETDELAY**：网络延迟，可以用于指定从一个网络的所有源端到所有接收端的传播延迟。

● **DEVICE**：器件延迟，主要用于描述分布式时序模型，可以用于指定通过单元到输出端口的所有路径的传播延迟。

以下是一些例子：

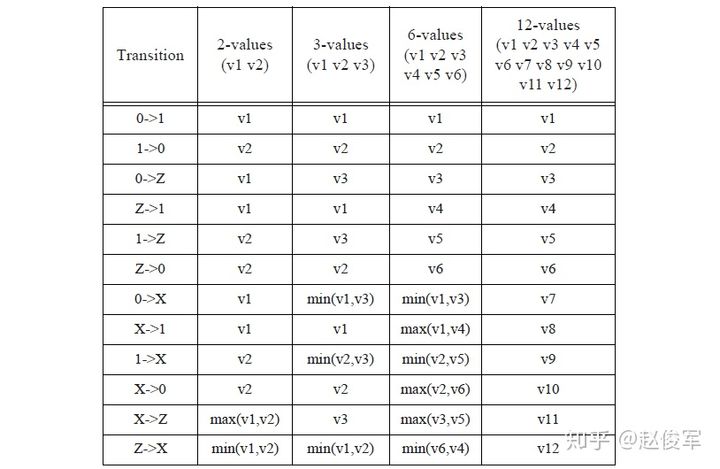


输入A上的值发生更改后，Y将保留其先前值50ps（低电平为40ps）。50ps是保持高电平的值，40ps是保持低电平的值，101ps是传播上升沿延迟，90ps是传播下降沿延迟，如图B-4所示。

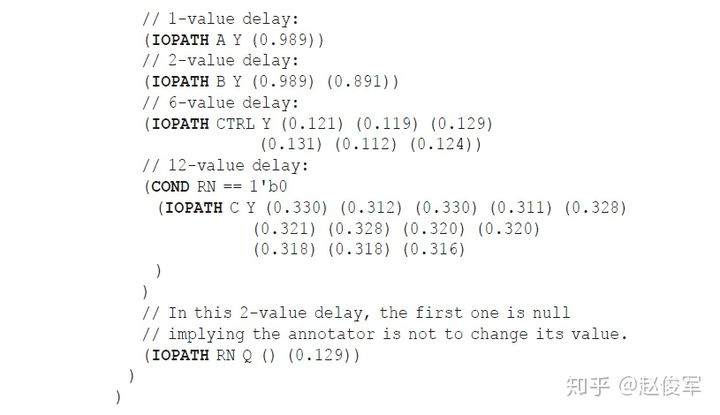
图B-4

**延迟**

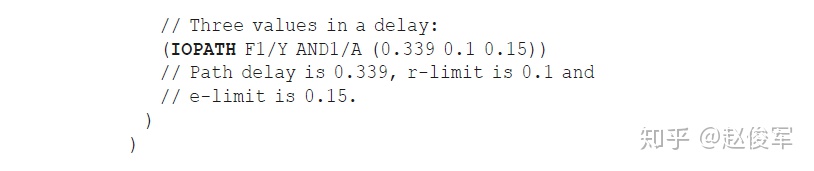
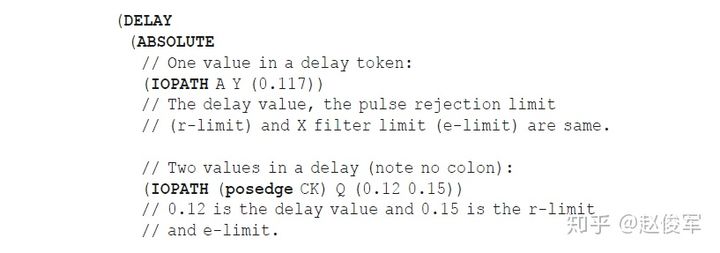
到目前为止，我们已经看过了许多不同形式的延迟，其实延迟规范还有其它形式。通常，可以将延迟指定为一个、两个、三个、六个或十二个令牌（token）的集合，这些令牌可用于描述以下过渡的延迟：0-> 1、1-> 0、0-> Z，Z-> 1，1-> Z，Z-> 0，0-> X，X-> 1,1-> X，X-> 0，X-> Z，Z-> X。下表展示了如何使用少于十二个延迟令牌来表示十二种过渡情况。



以下是这些延迟的一些示例：



每个延迟令牌可以依次写为一个、两个或三个值，如以下示例所示：



一个SDF文件中的延迟值可以使用有符号的实数或以下形式的三元数组来编写：

● （ 8.0：3.6：9.8 ）

为了表示出设计在三个工艺工作条件下的最小、典型以及最大延迟，注释器通常会根据用户提供的选项来决定去选择哪个值。三元数组形式中的值是可选的，但是至少应有一个。例如，以下形式是规范的：

● （ ： ： 0.22）

● （ 1.001 ： ： 0.998 ）

未指定的值就不会去标注。

**时序检查**

在以**TIMINGCHECK**关键字开头的部分中指定了时序检查的极限。在任何这些检查中，可以使用**COND**结构指定有条件的时序检查。在某些情况下，可以指定两个额外的条件检查**SCOND**和**CCOND**，它们与**stamp event**和**check event**关联。

以下是一组检查：

● **SETUP**：建立时间检查

● **HOLD**：保持时间检查

● **SETUPHOLD**：建立时间和保持时间检查

● **RECOVERY**：恢复时间检查

● **REMOVAL**：撤销时间检查

● **RECREM**：恢复时间和撤销时间检查

● **SKEW**：单向偏斜时序检查

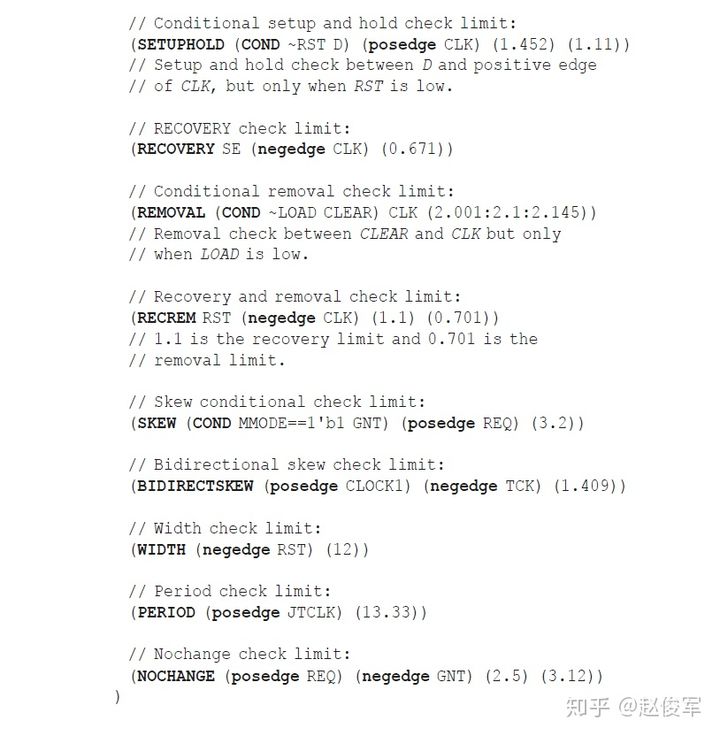
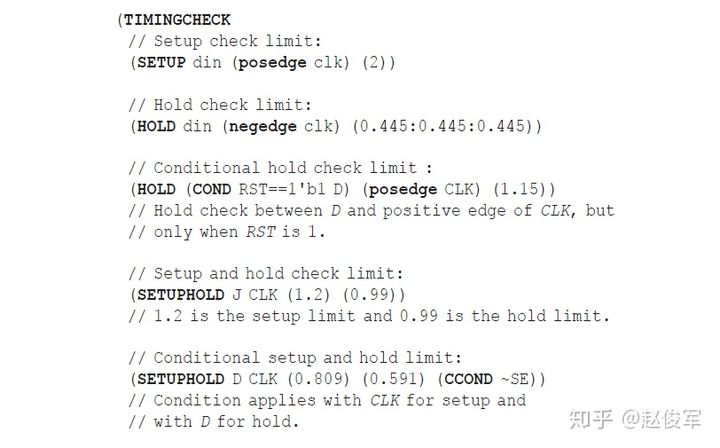
● **BIDIRECTSKEW**：双向偏斜时序检查

● **WIDTH**：脉宽时序检查

● **PERIOD**：周期时序检查

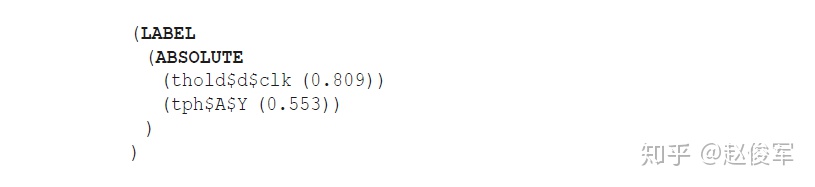
● **NOCHANGE**：不变化时序检查

以下是一些例子：



**标签**

标签可用于指定VHDL泛型（generics）或Verilog HDL参数的值。



**时序环境**

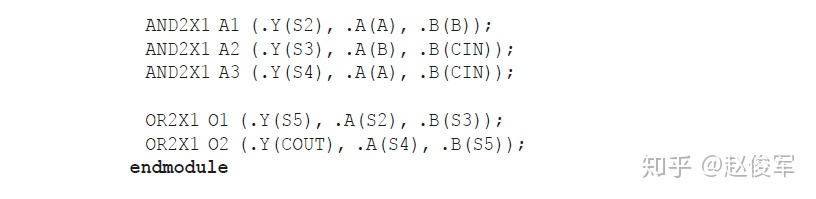
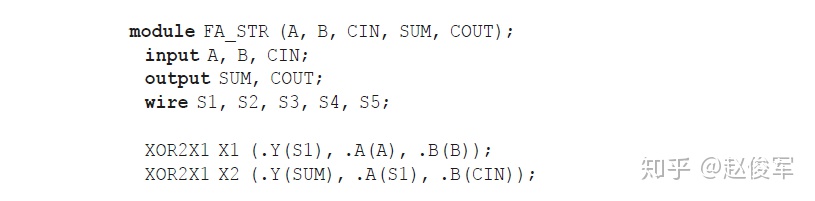
有许多结构可用于描述设计的时序环境。但是，这些结构可用于标注，而不是用于反标，例如在逻辑综合工具中。这些未在本文中描述。

**B.2.1 例子**

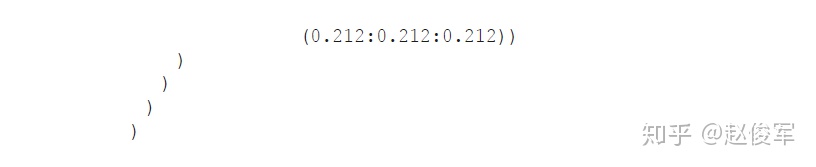
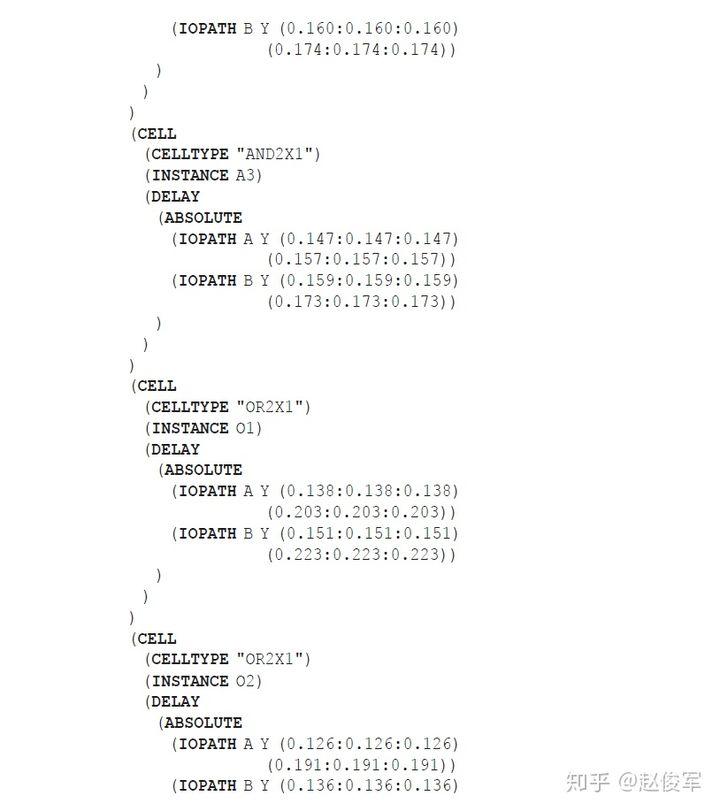
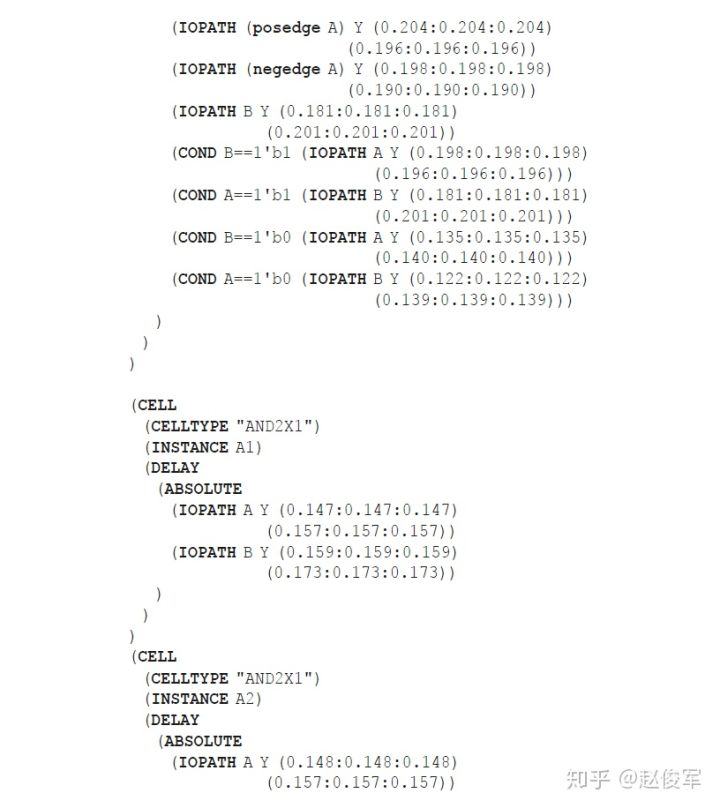
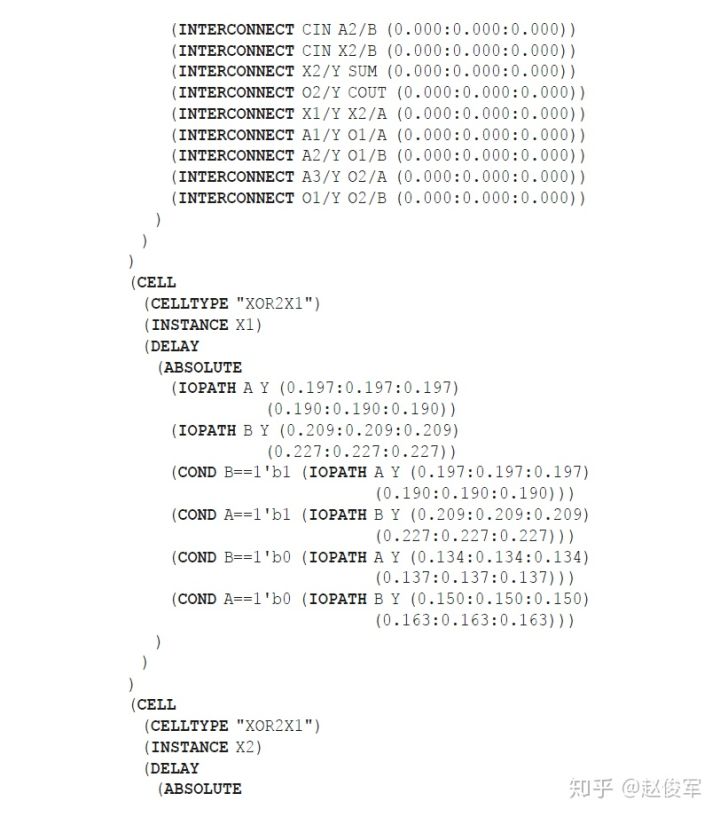
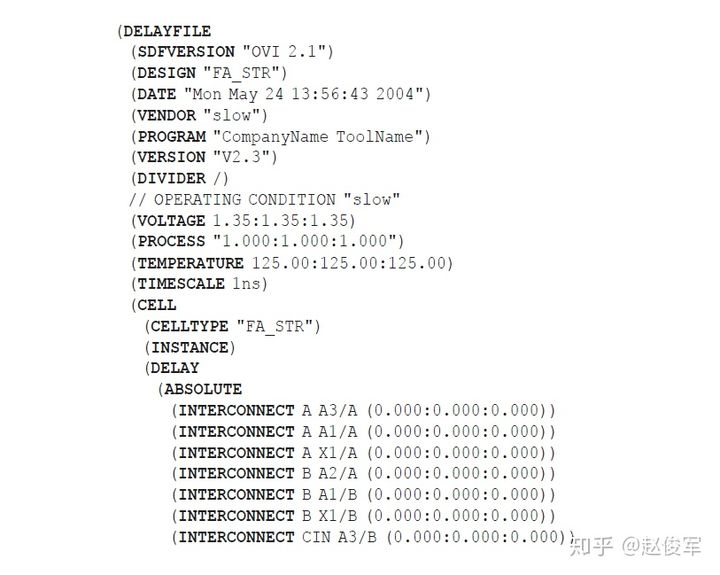
接下去，我们将为两个设计提供完整的SDF文件。

**全加器**

这是用于全加器（full-adder）电路的Verilog HDL网表（netlist）：



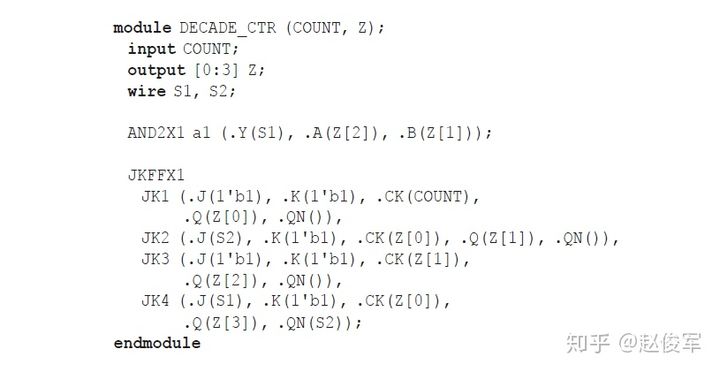
以下是时序分析工具生成的完整SDF文件：



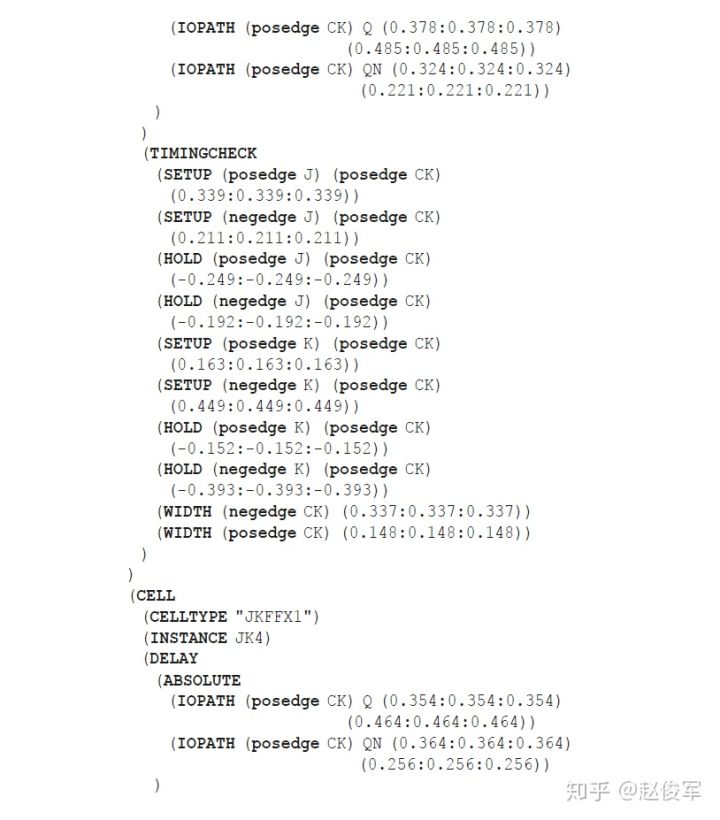
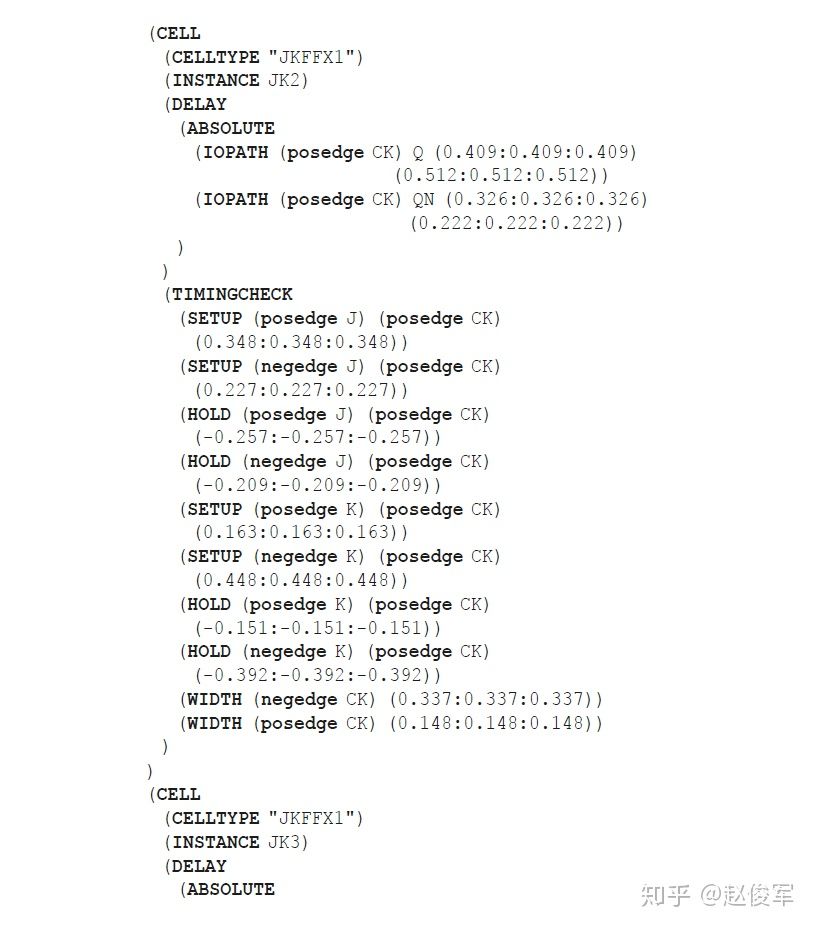
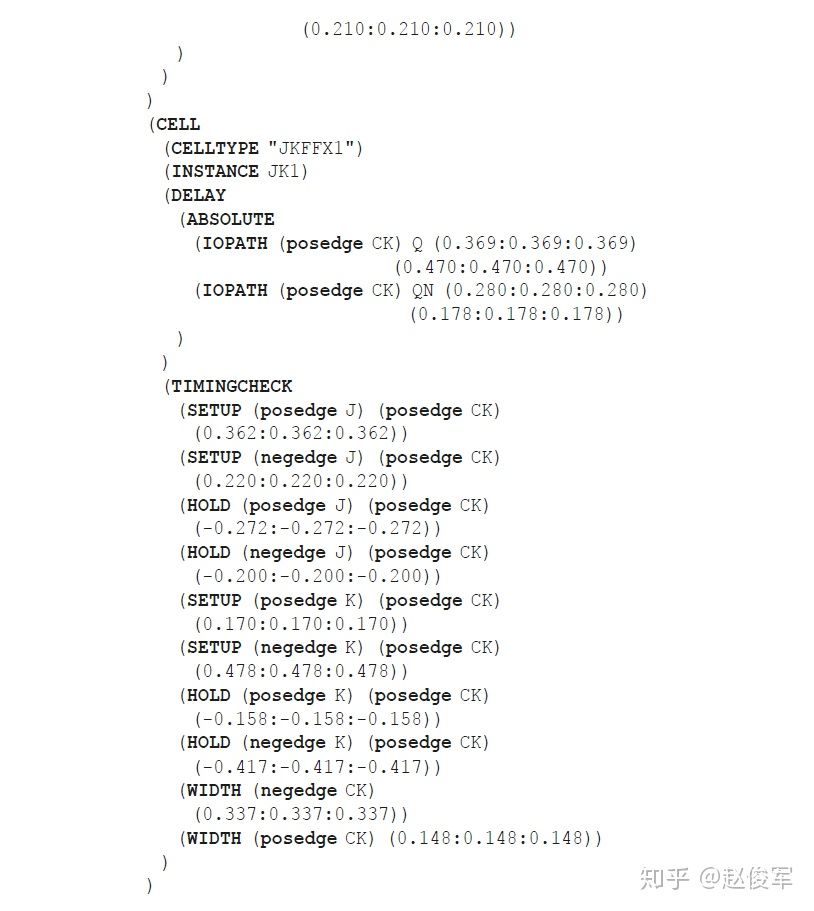
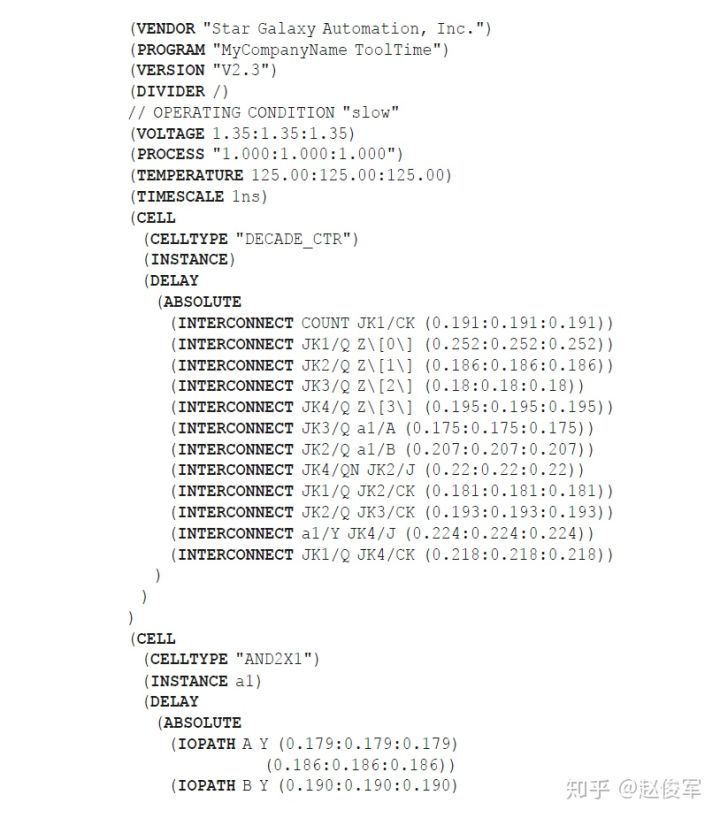
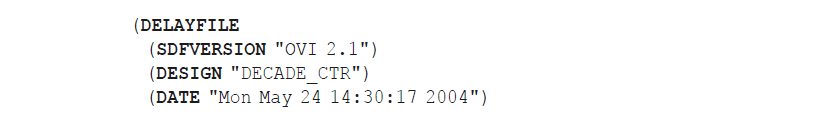
**INTERCONNECT**中的所有延迟均为0，因为这是布局前的数据，因此建模的是理想互连走线模型。

**十进制计数器**

这是十进制计数器的Verilog HDL模型：



对应的完整SDF文件如下：



**B.3 标注过程**

在本节中，我们将介绍如何在HDL描述中进行SDF的标注（annotation）。SDF的标注可以通过多种工具执行，例如逻辑综合工具、仿真工具和静态时序分析工具。SDF标注器（annotator）是这些工具的组件，可用于读取SDF、解释并向设计中标注时序值。假定会使用与HDL模型一致的信息创建SDF文件，并且在反标期间使用相同的HDL模型。此外，SDF标注器还需要负责正确解释SDF中的时序值。

SDF标注器标注了反标时序的泛型和参数。如果在语法或映射（mapping）过程中不符合该标准，它将给出错误报告。如果一个SDF标注器不支持某些SDF结构，则不会产生任何错误，标注器将忽略这些错误。

如果SDF标注器未能修改反标时序的泛型，则在反标过程中不会修改泛型的值，即保持不变。

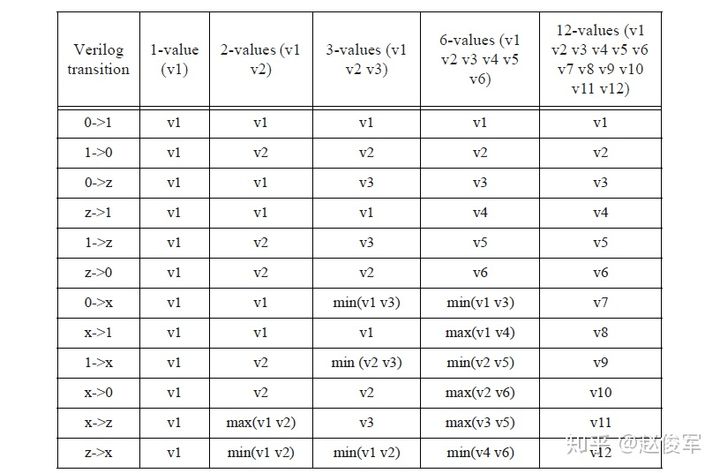
在仿真工具中，反标通常发生在规划（elaboration）阶段之后，紧接在负约束延迟计算之前。

**B.3.1 Verilog HDL**

在Verilog HDL中，标注的主要机制是指定块（specify block），指定块可以指定路径延迟和时序检查。实际延迟值和时序检查极限值是通过SDF文件指定的，映射是一种行业标准，在IEEE Std 1364中定义。

从SDF文件中获得并在Verilog HDL模块的指定块中标注的信息包括指定路径的延迟、参数值、时序检查约束极限值和互连走线延迟。向一个Verilog HDL模型进行标注时，将忽略SDF文件中的其它结构。SDF中的**LABEL**部分定义了参数值。通过将SDF结构与相应的Verilog HDL声明进行匹配，然后将现有的时序值替换为SDF文件中的时序值，即可完成反标。

下表显示了SDF延迟值如何映射到Verilog HDL延迟值：



下表描述了SDF结构到Verilog HDL结构的映射：



有关示例，请参见后面部分。

**B.3.2 VHDL**

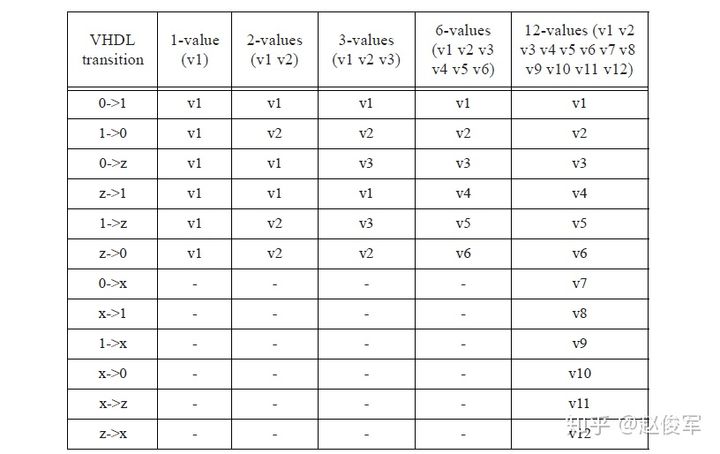
SDF到VHDL的标注是一个行业标准，它在VITAL ASIC建模规范的IEEE标准IEEE Std 1076.4中定义，该标准的其中一部分描述了SDF延迟到ASIC库的标注。在这里，我们仅介绍与SDF映射有关的VITAL标准的相关部分。

SDF可用于直接在符合VITAL的模型中修改反标时序泛型，只能使用SDF为符合VITAL的模型指定时序数据。有两种方法可以将时序数据传递到VHDL模型中：通过配置，或直接传递到仿真中去。SDF标注过程包括在仿真期间在符合VITAL的模型中映射SDF结构和相应的泛型。

在符合VITAL的模型中，存在着有关如何命名和声明泛型的规则，以确保可以在模型的时序泛型和相应的SDF时序信息之间建立映射。

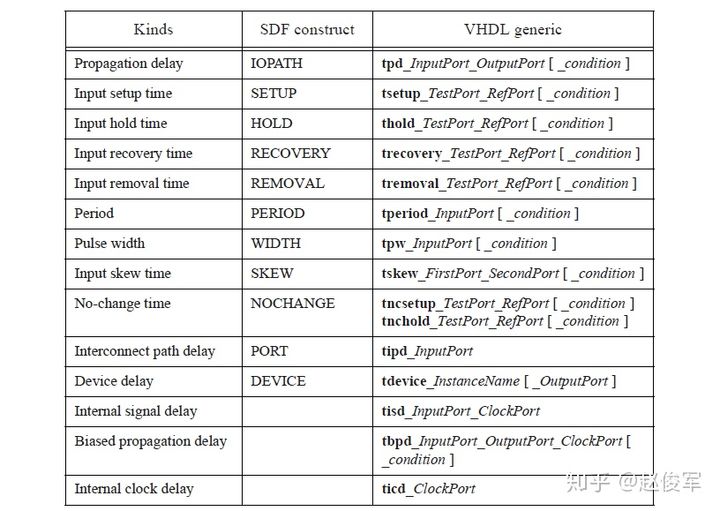
时序泛型由泛型名称及其类型组成，名称指定时序信息的种类，类型指定时序值的种类。如果泛型名称不符合VITAL标准，则它不是时序泛型，也不会被标注。

下表显示了SDF延迟值如何映射到VHDL延迟：



在VHDL中，时序信息是通过泛型进行反标的。泛型名称遵循一定的规则，以便保持一致或从SDF结构中获取。利用每个时序泛型名称，可以指定条件边沿的可选后缀。边沿可以指定一个与时序信息相关联的边沿。

下表列出了各种时序泛型名称：



附录B： SDF （未完待续......）