xilinx 的约束实现

前一段时间调试了 xilinx 的板子上跑代码，自己加 IP 核，看了它的约束文件，在网上找了一些讲语法的资料，自己整理了一下，我感觉在你了解了语法之后，确实得好好看一下它自己给出的约束，有些我自己没用到，我就没整理了。

1. 约束文件的概念

FPGA 设计中的约束文件有 3 类：用户设计文件（.UCF 文件）、网表约束文件（.NCF 文件）以及物理约束文件（.PCF 文件），

可以完成时序约束、管脚约束以及区域约束。3 类约束文件的关系为：

用户在设计输入阶段编写 UCF 文件，然后 UCF 文件和设计综合后生成 NCF 文件，最后再经过实现后生成 PCF 文件。

本节主要介绍 UCF 文件的使用方法。

UCF 文件是 ASC 2 码文件，描述了逻辑设计的约束，可以用文本编辑器和 Xilinx 约束文件编辑器进行编辑。

NCF 约束文件的语法和 UCF 文件相同，二者的区别在于： UCF 文件由用户输入，NCF 文件由综合工具自动生成，

当二者发生冲突时，以 UCF 文件为准，这是因为 UCF 的优先级最高。PCF 文件可以分为两个部分：

一部分是映射产生的物理约束，另一部分是用户输入的约束，同样用户约束输入的优先级最高。

一般情况下，用户约束都应在 UCF 文件中完成，不建议直接修改 NCF 文件和 PCF 文件。

2.UCF 文件的语法说明

UCF 文件的语法为：

{NET|INST|PIN} ″signal_name″ Attribute; 其中，“signal_name”是指所约束对象的名字，包含了对象所在层次的描述；

“Attribute”为约束的具体描述；语句必须以分号“；”结束。可以用“#”或“/* */”添加注释。

需要注意的是：UCF 文件是大小写敏感的，信号名必须和设计中保持大小写一致，但约束的关键字可以是大写、小写甚至大小写混合。

在 UCF 文件中描述管脚分配的语法为： NET “端口名称” LOC = 引脚编号;

NET ″CLK″ LOC = P30; “CLK”就是所约束信号名，LOC = P30；是约束具体的含义，将 CLK 信号分配到 FPGA 的 P30 管脚上。

通配符

在 UCF 文件中，通配符指的是“*”和“?”。“*”可以代表任何字符串以及空，“?”则代表一个字符。

在编辑约束文件时，使用通配符可以快速选择一组信号，当然这些信号都要包含部分共有的字符串。例如：

NET ″*CLK?″ FAST;

将包含“CLK”字符并以一个字符结尾的所有信号，并提高了其速率。

在位置约束中，可以在行号和列号中使用通配符。例如：

INST "/CLK_logic/*" LOC = CLB_r*c7;
把 CLK_logic 层次中所有的实例放在第 7 列的 CLB 中。
3.约束
3.1 管脚约束：
最简单的应用主要是位置约束 LOC，和电平标准 IOSTANDARD
NET CLK_27MHZ_FPGA LOC="AG18"; # Bank 4, Vcco=3.3V, No DCI
NET CLK_33MHZ_FPGA LOC="AH17"; # Bank 4, Vcco=3.3V, No DCI
NET CLK_FPGA_N LOC="K19"; # Bank 3, Vcco=2.5V, No DCI

fpga_0_SysACE_CompactFlash_SysACE_MPA_pin<5> IOSTANDARD = LVCMOS33;
3.2 时序约束：
1）周期约束

PERIOD 约束是一个基本时序和综合约束，它附加在时钟网线上，时序分析工具根据 PERIOD 约束检查时钟域内所有同步元件的时序是否满足要求，
它将检查与同步时序约束端口相连接的所有路径的延迟，但是不会检查 PAD 到寄存器的路径。
附加时钟周期约束的首选方法（Preferred Method）语法如下：
TIMESPEC “TSidentifier” = PERIOD “TNM_reference” period {HIGH|LOW} [high_or_low_time]
其中“[]”内为可选项，“{}”为必选项，参数 period 为要求的时钟周期，可以使用 ps、ns、us 或者 ms 等单位，大小写都可以，缺省单位为 ns。
HIGH|LOW 关键词指出时钟周期里的第一个脉冲是高电平还是低电平，而 high_or_low_time 为脉冲的延续时间，缺省单位也是 ns，如果不提供该项，则缺省占空比为 50%。
TIMESPEC 是一个基本时序相关约束标识，表示本约束为时序规范。TSidentifier 包括字母 TS 和一个标识符 identifier（为 ASCII 码字符串）
共同组成一个时序规范。例如定义时钟周期约束时，首先在时钟网线 clk 上附加一个 TNM_NET 约束，把 clk 驱动的所有同步元件定义为一个名为 sys_clk 的分组，然后使用 TIMESPEC 约束定义时钟周期。
NET “clk” TNM_NET=”sys_clk”;
TIMESPEC “TS_sys_clk”= PERIOD “sys_clk” 50 HIGH 30;
而定义派生时钟的语法如下：

TIMESPEC “TSidentifier_2”=PERIOD “timegroup_name” “TSidentifier_1” [*or/] factor PHASE [+|-] phase_value [units];

其中 TSidentifier_2 为要定义的派生时钟，TSidentifier_1 为已经定义的时钟，factor 指出两者周期的辈出关系，是一个浮点数。
phase_value 指出两者之间的相位关系，为浮点数。例如：

定义主时钟 clk0：
TIMESPEC “TS01” = PERIOD “clk0” 10.0 ns;

定义派生时钟 clk180，其相位与主时钟相差 180°：
TIMESPEC “TS02” = PERIOD “clk180” TS01 PHASE + 5.0 ns;

定义派生时钟 clk180_2,其周期为主时钟的 1/2，并延迟 2.5ns：
TIMESPEC “TS03” = PERIOD “clk180_2” TS01 /2 PHASE + 2.5 ns;

2)偏移约束

偏移约束规定了外部时钟和数据输入输出引脚之间的时序关系，只用于与 PAD 相连的信号，不能用于内部信号。
使用该约束可以为综合实现工具指出输入数据到达的时刻，或者输出数据稳定的时刻，从而在综合实现中调整布局布线过程，
使正在开发的 FPGA/CPLD 的输入建立时间以及下一级电路的输入建立时间满足要求。

基本语法如下：

OFFSET = {IN|OUT} “offset_time” [units] {BEFORE|AFTER} “clk_name” [TIMEGRP “group_name”];

其中{IN|OUT}说明约束的是输入还是输出，offset_time 为 FPGA 引脚数据变化与有效时钟沿之间的时间 差，BEFORE|AFTER 说明该时间差在有效时钟沿的前面还是后面，TIMEGRP “group_name”定义了约束的触发器组，缺省时约束该时钟驱动的所有触发器。
OFFSET 根据芯片外围电路的时序特性约束了内部延时。
OFFSET_IN 约束输入信号
OFFSET_IN_AFTER：输入信号（in）在时钟后（after）多长时间进入芯片。
OFFSET_IN_BEFORE：输入信号（in）在时钟前（before）多长时间进入芯片。
显然 CLOCK = OFFSET_IN_AFTER + OFFSET_IN_BEFORE。
OFFSET_IN_AFTER 反映的是外围电路的时序特性，我们无法左右。
OFFSET_IN_BEFORE 是留给芯片的时序余量。PAR 的目的就是要满足这个余量，不得超过。
OFFSET_OUT 约束输出信号
OFFSET_OUT_AFTER：输出信号（out）在时钟后（after）多长时间输出芯片。
OFFSET_OUT_BEFORE：输出信号（out）在时钟前（before）多长时间输出芯片。
显然 CLOCK = OFFSET_OUT_AFTER + OFFSET_OUT_BEFORE。
OFFSET_OUT_BEFORE 反映的是外围电路的时序特性，我们无法左右。
OFFSET_OUT_AFTER 是留给芯片的时序余量。PAR 的目的就是要满足这个余量，不得超过。

例
OFFSET = IN 10 ns BEFORE "i_ref_clk";
NET “Q_OUT” OFFSET=OUT 15.0 BEFORE “CLK_SYS”;
说明：第二个例子好理解，第一个例子没有指明对象，表示所有受到“i_ref_clk”约束的信号都被施加了该约束

3）分组约束

使用 TNM（Timing Name）约束可以选出构成一个分组的元件，并赋予一个名字，以便给它们附加约束。
TNM_NET（timing name for nets）约束只加在网线上，其作用与 TNM 加在网上时基本相同，即把该网线所在路径上的所有有效同步元件作为命名组的一部分。
不同之处在于当 TNM 约束加在 PAD NET 上时，TNM 的值将被赋予 PAD，而不是该网线所在的路径上的同步元件，即 TNM 约束不能穿过 IBUF。
而用 TNM_NET 约束就不会出现这种情况。
TNM 是 Timing Name 的缩写，是一种 grouping 约束，用于把若干信号组合成一个特定的组以施加相同的约束。例如
INST "o_dbg_out[0]" TNM = "dbg_out";
INST "o_dbg_out[1]" TNM = "dbg_out";
.....
INST "o_dbg_out[63]" TNM = "dbg_out";
采用该方法后一些列信号都被列入名为“dbg_out”的组合。然后可以统一施加约束。如
TIMESPEC "TS_FFS_2_dbg_out" = FROM "FFS" TO "dbg_out" TIG;

4）专门约束

附加约束的一般策略是首先附加整体约束，例如 PERIOD、OFFSET 等，然后对局部的电路附加专门约束，这些专门约束通常比整体约束宽松，
通过在可能的地方尽量放松约束可以提高布线通过率，减小布局布线的时间。

FROM_TO 约束在两个组之间定义时序约束，对两者之间的逻辑和布线延迟进行控制，这两个组可以是用户定义的，也可以是与定义的。
用户可以使用 TNM_NET、TNM 和 TIMEGRP 定义组，而与定义组主要包括 FFS、LATCHES、PADS 和 RAMS 等。语法如下：

TIMESPEC “TSname” = FROM “group1” TO “group2” value;

其中 value 为延迟时间，可以使具体数值或表达式。

MAXDELAY 约束定义了特定网线上的最大延迟，其语法如下：

NET “net_name” MAXDELAY = value units;

TIG 是 Timing IGnore 的缩写。表示忽略该对象上的时序约束。例如
NET “RESET” TIG=TS_fast, TS_even_faster;
表示对 RESET 信号，时序约束 TS_fast 和 TS_even_faster 无效。

关于 from to 的约束是一个比较实用，也比较好用的约束。 尤其在跨时钟域处理中的应用，特别有效。

举例： 设计中有两个时钟，一个是 PLL 的输入，一个是 PLL 的输出，并且两者为异步形式，当设计中有数据在该两个时钟域中传递时，

又没有用到 fifo 隔离，那么这个使用就需要用这个约束。

```
TIMEGRP "tg_from" = RISING "PLL_IN" ;
TIMEGRP "tg_to" = RISING "PLL_OUT" ;

TIMESPEC "TS_CLK" = FROM "PLL_IN" TO "PLL_OUT" TIG;
TIMESPEC "TS_CLK" = FROM "PLL_OUT" TO "PLL_IN" TIG;
```

这样工具就不会对该两个跨时钟路径就不会分析了。

如果不加这个约束，时序会比较紧，很容易造成其它模块的时序过不去。

当然如果不用 TIG，给具体的时间也是可以的，那么时钟域之间的延时只要小于你给的时间值就不会报错。