1. BUFG，IBUFG，BUFGP，IBUFGDS等含义以及使用

目前，大型设计一般推荐使用同步时序电路。同步时序电路基于时钟触发沿设计，对时钟的周期、占空比、延时和抖动提出了更高的要求。为了满足同步时序设计的要求，一般在FPGA设计中采用全局时钟资源驱动设计的主时钟，以达到最低的时钟抖动和延迟。 FPGA 全局时钟资源一般使用全铜层工艺实现，并设计了专用时钟缓冲与驱动结构，从而使全局时钟到达芯片内部的所有可配置单元(CLB)、I/O单元(IOB)和选择性块RAM(Block Select RAM)的时延和抖动都为最小。为了适应复杂设计的需要，Xilinx的FPGA中集成的专用时钟资源与数字延迟锁相环(DLL)的数目不断增加，最新的 Virtex II器件最多可以提供16个全局时钟输入端口和8个数字时钟管理模块(DCM)。

与全局时钟资源相关的原语常用的与全局时钟资源相关的Xilinx器件原语包括：

BUFG：Single-ended Input Global Clock Buffer

IBUFGDS：Differential Input Global Clock Buffer

BUFG： Global Clock Buffer

BUFGP：IBUFG + BUFG

BUFGCE：Global Clock Buffer Clock Enable

BUFGMUX：全局时钟选择

BUFGDLL：全局缓冲延迟锁相环（舍，被 DCM 代替）

DCM: DCM\_ADV DCM\_BASE

1. IBUFG即输入全局缓冲，是与专用全局时钟输入管脚相连接的首级全局缓冲。所有从全局时钟管脚输入的信号必须经过IBUF元，否则在布局布线时会报错。 IBUFG支持AGP、CTT、GTL、GTLP、HSTL、LVCMOS、LVDCI、LVDS、LVPECL、LVTTL、PCI、PCIX和 SSTL等多种格式的IO标准。

2. IBUFGDS是IBUFG的差分形式，当信号从一对差分全局时钟管脚输入时，必须使用IBUFGDS作为全局时钟输入缓冲。IBUFG支持BLVDS、LDT、LVDSEXT、LVDS、LVPECL和ULVDS等多种格式的IO标准。

3. BUFG是全局缓冲，它的输入是IBUFG的输出，BUFG的输出到达FPGA内部的IOB、CLB、选择性块RAM的时钟延迟和抖动最小。

4. BUFGCE是带有时钟使能端的全局缓冲。它有一个输入I、一个使能端CE和一个输出端O。只有当BUFGCE的使能端CE有效(高电平)时，BUFGCE才有输出。

5. BUFGMUX是全局时钟选择缓冲，它有I0和I1两个输入，一个控制端S，一个输出端O。当S为低电平时输出时钟为I0，反之为I1。需要指出的是BUFGMUX的应用十分灵活，I0和I1两个输入时钟甚至可以为异步关系。

6. BUFGP相当于IBUG加上BUFG。

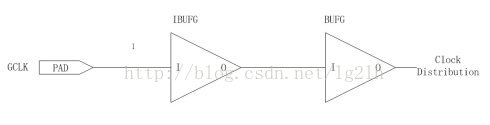
7. BUFGDLL是全局缓冲延迟锁相环，相当于BUFG与DLL的结合。BUFGDLL在早期设计中经常使用，用以完成全局时钟的同步和驱动等功能。随着数字时钟管理单元(DCM)的日益完善，目前BUFGDLL的应用已经逐渐被DCM所取代。

8. DCM即数字时钟管理单元，主要完成时钟的同步、移相、分频、倍频和去抖动等。DCM与全局时钟有着密不可分的联系，为了达到最小的延迟和抖动，几乎所有的DCM应用都要使用全局缓冲资源。DCM可以用Xilinx ISE软件中的Architecture Wizard直接生成。

1.1全局时钟资源的使用方法(五种)

1：IBUFG / IBUFGDS+ BUFG的使用方法：

IBUFG后面连接BUFG的方法是最基本的全局时钟资源使用方法，由于IBUFG组合BUFG相当于BUFGP，所以在这种使用方法也称为BUFGP方法，当信号从全局时钟引脚输入时，无论是否为时钟信号，都必须使用IBUFG/IBUFGDS; 反之，如果使用了IBUFG/ IBUFGDS，则信号必须从全局时钟引脚输入，否则布局布线会报错。IBUFG/IBUFGDS的输入只与芯片的专用全局时钟输入引脚有物理连接，与普通的I/O和其他内部CLB没有物理连接，所以后面要加BUFG。



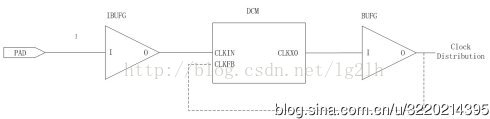
2. IBUFGDS + BUFG的使用方法：

当输入时钟信号为差分信号时，需要使用IBUFGDS代替IBUFG。

3. IBUFG + DCM + BUFG的使用方法：

这种使用方法最灵活，对全局时钟的控制更加有效。通过DCM模块不仅仅能对时钟进行同步、移相、分频和倍频等变换，而且可以使全局时钟的输出达到无抖动延迟。

平时用得最多的还是IBUFG+DCM（PLL）+BUFG方案，如下图所示。将时钟管脚输入的时钟作为IBUFG的输入，然后将IBUFG的输出作为DCM（PLL）的输入，将经DCM（PLL）频率变换后的输出再作为BUFG的输入这种方案使用方法最为灵活，对全局时钟的控制更加有效。通过DCM（PLL）模块不仅能对时钟进行同步、移相、分频、倍频等变换，而且可以使全局时钟的输出达到无抖动延迟（“0”skew）。



4. Logic ＋ BUFG的使用方法：

BUFG不但可以驱动IBUFG的输出，还可以驱动其它普通信号的输出。当某个信号(时钟、使能、快速路径)的扇出非常大，并且要求抖动延迟最小时，可以使用BUFG驱动该信号，使该信号利用全局时钟资源。但需要注意的是，普通IO的输入或普通片内信号进入全局时钟布线层需要一个固有的延时，一般在 10ns左右，即普通IO和普通片内信号从输入到BUFG输出有一个约10ns左右的固有延时，但是BUFG的输出到片内所有单元(IOB、CLB、选择性块RAM)的延时可以忽略不计为“0”ns。

5． Logic + DCM + BUFG的使用方法：

DCM同样也可以控制并变换普通时钟信号，即DCM的输入也可以是普通片内信号。使用全局时钟资源的注意事项全局时钟资源必须满足的重要原则是：使用 IBUFG或IBUFGDS的充分必要条件是信号从专用全局时钟管脚输入。换言之，当某个信号从全局时钟管脚输入，不论它是否为时钟信号，都必须使用 IBUFG或IBUFGDS；如果对某个信号使用了IBUFG或IBUFGDS硬件原语，则这个信号必定是从全局时钟管脚输入的。如果违反了这条原则，那么在布局布线时会报错。这条规则的使用是由FPGA的内部结构决定的：IBUFG和IBUFGDS的输入端仅仅与芯片的专用全局时钟输入管脚有物理连接，与普通IO和其它内部CLB等没有物理连接。另外，由于BUFGP相当于IBUFG和BUFG的组合，所以BUFGP的使用也必须遵循上述的原则。

1.2 全局时钟资源的例化方法

全局时钟资源的例化方法大致可分为两种：

一是在程序中直接例化全局时钟资源；

二是通过综合阶段约束或者实现阶段约束实现对全局时钟资源的使用；

第一种方法比较简单，用户只需按照前面讲述的5种全局时钟资源的基本使用方法编写代码或者绘制原理图即可。

第二方法是通过综合阶段约束或实现阶段的约束完成对全局时钟资源的调用，这种方法根据综合工具和布局布线工具的不同而异。

1.3详解IBUFG/IBUFGDS+BUFG用法

IBUFG(全局时钟缓冲)后面接BUFG（全局缓冲）是最基本的全局时钟使用方法。

IBUFG CLKIN\_IBUFG (

.I(CLKIN)，

.O(CLKIN\_IBUFG)

);

BUFG CLK\_BUFG (

.I(CLKIN\_IBUFG),

.O(CLK\_OUT)

);

当输入信号是差分信号时，需要使用IBUFGDS代替IBUFG，相应语法如下：

IBUFGDS CLKIN\_IBUFGDS (

.I(CLKIN\_P\_IN),

.IB(CLKIN\_N\_IN),

.O(CLKIN\_IBUFGDS)

);

BUFG CLK\_BUFG (

.I(CLKIN\_IBUFGDS),

.O(CLK\_OUT)

);

一般来说，外部提供的时钟信号都需要进行倍频/分频才可以使用，这时候需要组合各种时钟缓冲器和 DCM、PLL 等模块，我们有两种方法：

**1. 代码中例化原语**，手动组合各种时钟缓冲器和 DCM、PLL

**2. 使用 IP core 向导**，创建时钟管理器（可以发现 IP core 生成的代码就是上面 5 种组合方式）

New source -> IP core -> Clocking Wizard

个人感觉使用第二种方法应该更加简洁、方便，不容易出错吧。（如果在代码中没有明确声明使用buffer，ISE 综合属性、IP core 属性设置里面默认会给所有的输入输出自动加上缓冲器）

对 FPGA 设计而言，全局时钟是最简单最可预测的时钟，最好的时钟方案是：由专用的全局时钟输入引脚驱动单个全局时钟，并用后者去控制设计中的每个触发器。全局时钟资源是专用布线资源，存在与全铜布线层上，使用全局时钟资源不影响芯片的其他布线资源，因此在可以使用全局时钟的时候尽可能使用。

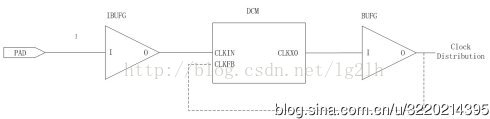
1.4全局时钟系统使用的问题与注意事项

1. 全局时钟的时钟源必须从全局时钟管脚输入，并且要先经过**IBUFG**。

2. 如果全局时钟涉及到逻辑运算，可以将BUFG换成BUFGCE（与）、BUFGMUX（二选一）等，事实上，BUFG、BUFGCE等资源均是由BUFGMUX生成。

3. 从BUFG输出的时钟，是不能直接连接普通I/O管脚输出的，会报错，若要将全局时钟输出，有两种方法：一是直接将BUFG的输入连接普通I/O管脚输出，或者将BUFG的输出经由ODDR2后再连接I/O管脚输出。需要注意的是，将CLOCK\_DEDICATED\_ROUTE属性设为FALSE虽然会将ERROR降为WARNING，但是这样时钟系统将不再是全局时钟系统，不建议这样做。

4. DCM（PLL）资源中有时钟反馈输入和时钟反馈输出管脚，一般的处理方法是将时钟反馈输出管脚经由BUFG后输入到时钟反馈输入管脚。



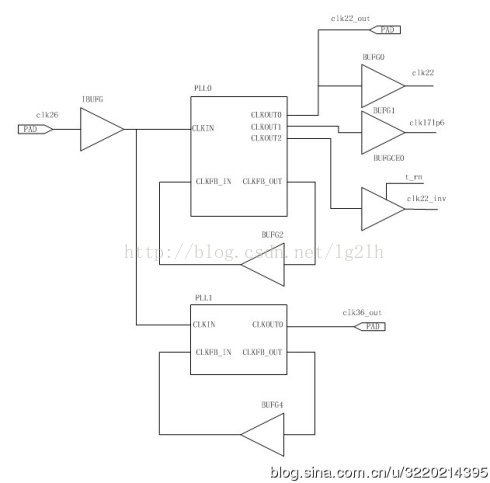
1.5全局时钟系统使用举例

下面将举出实例，来看一下如何组建一个全局时钟系统。

某FPGA从全局时钟管脚引入频率为26M的晶振源，FPGA内部需要22MHz、22MHz反向、171.6MHz、36MHz，其中22MHz、22MHz反向、171.6MHz为FPGA内部使用，22MHz、36MHz经由普通I/O管脚输出，而且22MHz反向还要与信号t\_rn作逻辑与运算，试给出该需求的全局时钟系统解决方案。

在ISE中组建时钟系统有两种方法：一是用IP核生成器配置生成相应IP，再用线连接起来， 二是在文件中直接用语言配置生成各类资源（DCM、PLL、IBUFG、BUFG等），再用线连接。第一种方法较为简单但灵活性差，而第二种方法稍微复杂但灵活性较强，可以随时修改，使用哪种方法取决于个人喜好。

这里用第二种方法来生成上文需求的全局时钟系统，下图为最终效果图。



1. clk26为从全局时钟管脚输入的时钟源，从管脚输入后先经过一个IBUFG。

2. 因为22MHz、171.6MHz和36MHz不能由一个PLL产生，故将IBUFG的输出时钟信号输入到两个PLL。

3. PLL0有三个输出：CLKOUT0、CLKOUT1、CLKOUT2，分别为22MHz、171.6MHz和22MHz反向。clk22\_out为22MHz时钟的I/O输出信号，故不经BUFG直接输出；clk22为22MHz经过BUFG后的信号，为全局时钟信号；clk171p6为171.6MHz经过BUFG后的信号，为全局时钟信号；clk22\_inv为22MHz反向经过BUFGCE（与t\_rn作逻辑与运算）后的信号，为全局时钟信号。

4. PLL1有一个输出：CLKOUT0，为36MHz，clk36\_out为36MHz的I/O输出信号，故不经BUFG直接输出。

5. PLL的时钟反馈输入和时钟反馈输出管脚相连，上面处理方法是将时钟反馈输出管脚经由BUFG后输入到时钟反馈输入管脚

# 第二全局时钟资源

第二全局时钟资源属于长线资源，长度和驱动能力仅次于全局时钟资源，也可以驱动芯片内部的任何一个逻辑，抖动和延时仅次于全局时钟。在设计中，一般将高频率、高扇出的时钟使能信号以及高速路径上的关键信号指定为全局第二时钟信号。使用全局时钟资源并不占用逻辑资源，也不影响其他布线资源；第二时钟资源占用的是芯片内部的资源，占用部分逻辑资源，各个部分的布线会相互影响，所以建议在设计中逻辑占用资源不超过70%时使用。

使用第二时钟资源：可以在约束编辑器中的专用约束Misc选项中，指定所选信号使用低抖动延迟资源“Low Skew”来指定，也可以在ucf文件中添加“USELOWSKEWLINES"约束命令。比如：

NET "s1" USELOWSKEWLINES;

# ISE时序分析器

单击Design Summary中的Static Timing就可以启动时序分析器（Timing Analyzer）。在综合、布局布线阶段ISE就会估算时延，给出大概的时延和所能达到的最大时钟频率，经过PAR后，在Static Timing中给出的是准确的时延，给出的时序报告可以帮助我们找到关键路径，然后针对其进行优化，提高系统的时钟频率。

造成时序性能差的原因很多，主要分为下面3种：

**1. 布局太差**

一般和代码本身没有关系。解决方案：只能从软件自身的布局算法考虑（调整布局的努力程度）或者使用高端芯片

**2. 逻辑级数太多**

逻辑级数越多，资源的利用率越高，但是对工作频率的影响也越大。解决方案：1.使用流水线技术；2.如果是多周期路径，添加多周期约束；3.良好的编码习惯，不要过多嵌套if-else，尽量使用case代替if语句。

**3. 信号扇出过高**

高扇出会造成信号传输路径过长，从而降低时序性能。解决方案：1.逻辑复制；2.区域约束，想过逻辑放置在一起。

**4. 不要同时使用双边沿触发**

FPGA的底层工艺都是单向的同步电路，所以本身不支持统一信号的爽边沿触发，ISE在实际处理的时候，会自动将该信号2倍频，然后利用第一个沿处理上升沿，第二个沿处理下降沿。这样在分析时序时，自动把约束升级为ucf文件中的两倍。

**5. Xilinx最优时序解决方案**

**I/O约束**

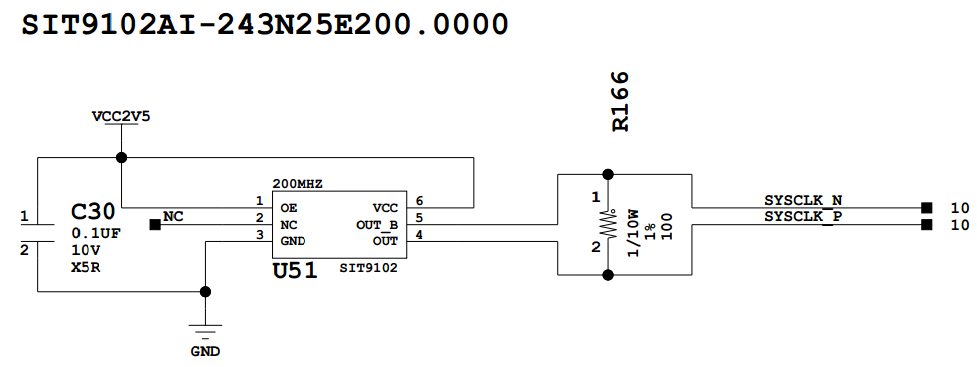
根据Xilinx器件的特点，控制信号置于器件的顶部或底部，且垂直布置；数据总线的I/O置于器件的左右两侧，且水平布置，这样可以最大程度的利用芯片底层结构。

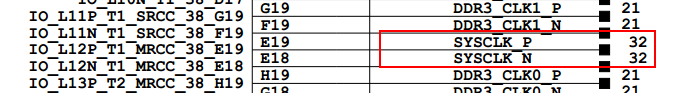
**6. ISE实现工具**

ISE中的工具具备不同的努力程度，直接使用最高级别的可以提高时序性能，但是会耗费很多时间，所以应该逐步调整努力程度。第一遍使用默认的参数选项，如果不满足再调整综合、映射、布局布线的参数。

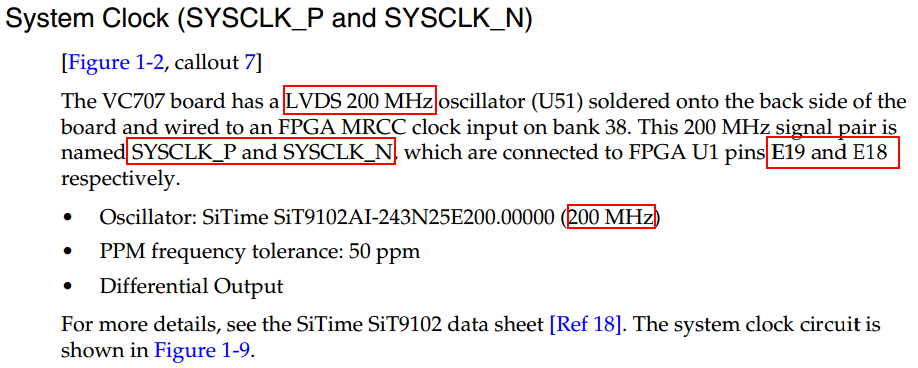
# Virtex-7时钟管脚具体分配

在使用Vrtex-7 VX485T-2FFG1761的时候，由于相比于Virtex-6系列，7系列把全局时钟管脚去掉了，所以在应用中，使用了Virtex-7的系统时钟（User Clock一般用作iic的传输的配置时钟），由于是差分时钟，不用直接用单端时钟连接，或者用单端时钟链接差分时钟的P级。差分时钟具体电路和说明如下:





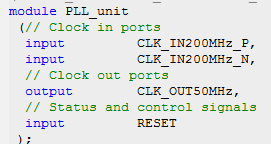
1. VC707 管脚图



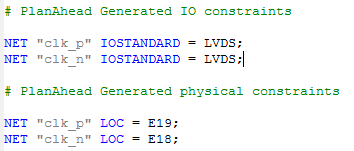
1. 官方文档Ug885

具体调用方法 new source -> IP Core -> Clocking Wizard -> input Freq 选择200M(根据晶振) -> Source选择Differential clock capable pin -> 输出频率，相位 -> 完成

点看生成的PLL的Function Model可以看到



设置管脚的ucf约束文件如下：



一般来说如果是差分时钟输入，设置好P级就可以，N级会自动配对，如果不放心就可以像上面那样，手动设置N级，还要注意的地方是IOSTANDARD，如果时钟类型是单端，则无法选取IOSTANDARD为LVDS类型，为什么要选择LVDS? 因为官方文档就这样写着！之后的输出时钟就可以使用了。

附录：

## A: ３个原语的组合代码

使用全局时钟资源可以直接用原语例化，也可以使用IP核。仔细观察了一下IP核产生的源文件，发现 IP核生成的就是这３个原语的组合：

module DCM\_100M(CLKIN\_IN,

RST\_IN,

CLKIN\_IBUF6G\_OUT,

CLK0\_OUT,

CLK2X\_OUT,

LOCKED\_OUT);

input CLKIN\_IN;

input RST\_IN;

output CLKIN\_IBUFG\_OUT;

output CLK0\_OUT;

output CLK2X\_OUT;

output LOCKED\_OUT;

wire CLKFB\_IN;

wire CLKIN\_IBUFG;

wire CLK0\_BUF;

wire CLK2X\_BUF;

wire GND\_BIT;

wire [6:0] GND\_BUS\_7;

wire [15:0] GND\_BUS\_16;

assign GND\_BIT = 0;

assign GND\_BUS\_7 = 7'b0000000;

assign GND\_BUS\_16 = 16'b0000000000000000;

assign CLKIN\_IBUFG\_OUT = CLKIN\_IBUFG;

assign CLK0\_OUT = CLKFB\_IN;

**IBUFG** CLKIN\_IBUFG\_INST (.I(CLKIN\_IN),

.O(CLKIN\_IBUFG));

**BUFG** CLK0\_BUFG\_INST (.I(CLK0\_BUF),

.O(CLKFB\_IN));

**BUFG** CLK2X\_BUFG\_INST (.I(CLK2X\_BUF),

.O(CLK2X\_OUT));

DCM\_ADV #( .CLK\_FEEDBACK("1X"), .CLKDV\_DIVIDE(2.0), .CLKFX\_DIVIDE(1),

.CLKFX\_MULTIPLY(4), .CLKIN\_DIVIDE\_BY\_2("FALSE"),

.CLKIN\_PERIOD(10.000), .CLKOUT\_PHASE\_SHIFT("NONE"),

.DCM\_AUTOCALIBRATION("TRUE"), .DCM\_PERFORMANCE\_MODE("MAX\_SPEED"),

.DESKEW\_ADJUST("SYSTEM\_SYNCHRONOUS"), .DFS\_FREQUENCY\_MODE("LOW"),

.DLL\_FREQUENCY\_MODE("LOW"), .DUTY\_CYCLE\_CORRECTION("TRUE"),

.FACTORY\_JF(16'hF0F0), .PHASE\_SHIFT(0), .STARTUP\_WAIT("FALSE"),

.SIM\_DEVICE("VIRTEX5") ) DCM\_ADV\_INST (.CLKFB(CLKFB\_IN),

.CLKIN(CLKIN\_IBUFG),

.DADDR(GND\_BUS\_7[6:0]),

.DCLK(GND\_BIT),

.DEN(GND\_BIT),

.DI(GND\_BUS\_16[15:0]),

.DWE(GND\_BIT),

.PSCLK(GND\_BIT),

.PSEN(GND\_BIT),

.PSINCDEC(GND\_BIT),

.RST(RST\_IN),

.CLKDV(),

.CLKFX(),

.CLKFX180(),

.CLK0(CLK0\_BUF),

.CLK2X(CLK2X\_BUF),

.CLK2X180(),

.CLK90(),

.CLK180(),

.CLK270(),

.DO(),

.DRDY(),

.LOCKED(LOCKED\_OUT),

.PSDONE());

Endmodule

时钟从CLKIN\_IN输入，经过IBUFG，输出为CLKIN\_IBUFG，然后输入到DCM\_ADV，输出为CLK0\_BUF和CLK2X\_BUF，CLK0\_BUF经过BUFG得到CLKFB\_IN，一方面反馈到DCM的，另一方面也从CLK0\_OUT输出；CLK2X\_BUF则经过BUFG后直接输出为CLK2X\_OUT。

LOGIC + DCM + BUFG

和前一种的区别在于DCM的输入是从内部输入还是外部输入。从外部输入则用IBUFG，保证时钟信号由芯片引脚输入；从内部输入则可以选择内部逻辑的任意信号，在FPGA内部是没有差分信号的，所有内部时钟信号都是单端信号。

## B: 差分时钟的UCF和例化代码

ucf为：

NET SYSCLK\_OUT\_P LOC = E19 | IOSTANDARD= LVDS\_25;

NET SYSCLK\_OUT\_N LOC = E18 | IOSTANDARD= LVDS\_25;

// System Clock (SYSCLK\_P and SYSCLK\_N) //E19 E18 UG885

module counter(

SYSCLK\_OUT\_P,

SYSCLK\_OUT\_N,

dout

);

input SYSCLK\_OUT\_P;

input SYSCLK\_OUT\_N;

output [1:0] dout;

wire CLK\_OUT;

wire CLK\_IN\_IBUFGDS;

reg [1:0] dout;

IBUFGDS CLK\_OUTIN\_IBUFGDS\_INST(

.I(SYSCLK\_OUT\_P),

.IB(SYSCLK\_OUT\_N),

.O(CLK\_IN\_IBUFGDS)

);

BUFG CLK\_OUT\_BUFG\_INST(

.I(CLK\_IN\_IBUFGDS),

.O(CLK\_OUT)

);

always @(posedge CLK\_OUT)

dout <= dout + 1;

endmodule

## C: IOSTANDARD 设置参考

NET  "clk\_p"  IOSTANDARD = LVDS\_25 ;  
NET  "clk\_n"  IOSTANDARD = LVDS\_25 ;  
NET  "clk\_n"  LOC = "Y12" ;  
NET  "clk\_p"  LOC = "W12" ;  
NET "clk\_buf" TNM\_NET = "SYS\_CLK";  
TIMESPEC "TS\_SYS\_CLK" = PERIOD "SYS\_CLK"  10  ns HIGH 50 %;  
NET "test"  LOC =     W9     | IOSTANDARD =     LVCMOS33; //IO\_L47P\_2  
NET "din"  LOC =     V9     | IOSTANDARD =   LVCMOS33;// IO\_L23N\_3