**实验9**

**基础信号处理**

# **实验介绍**

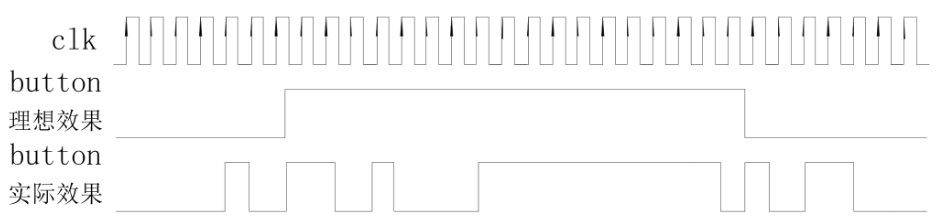
我们已经可以使用 Vivado 完成数字电路设计的完整流程了。然而，你可能会发现有时自己的设计明明没有什么问题，但上板运行的结果总是不如人意。这或许是因为你没有正确地进行信号处理。

在本实验的内容中，我们会为大家介绍一些基础的数字信号处理技术。

# 实验内容

## 1 去毛刺

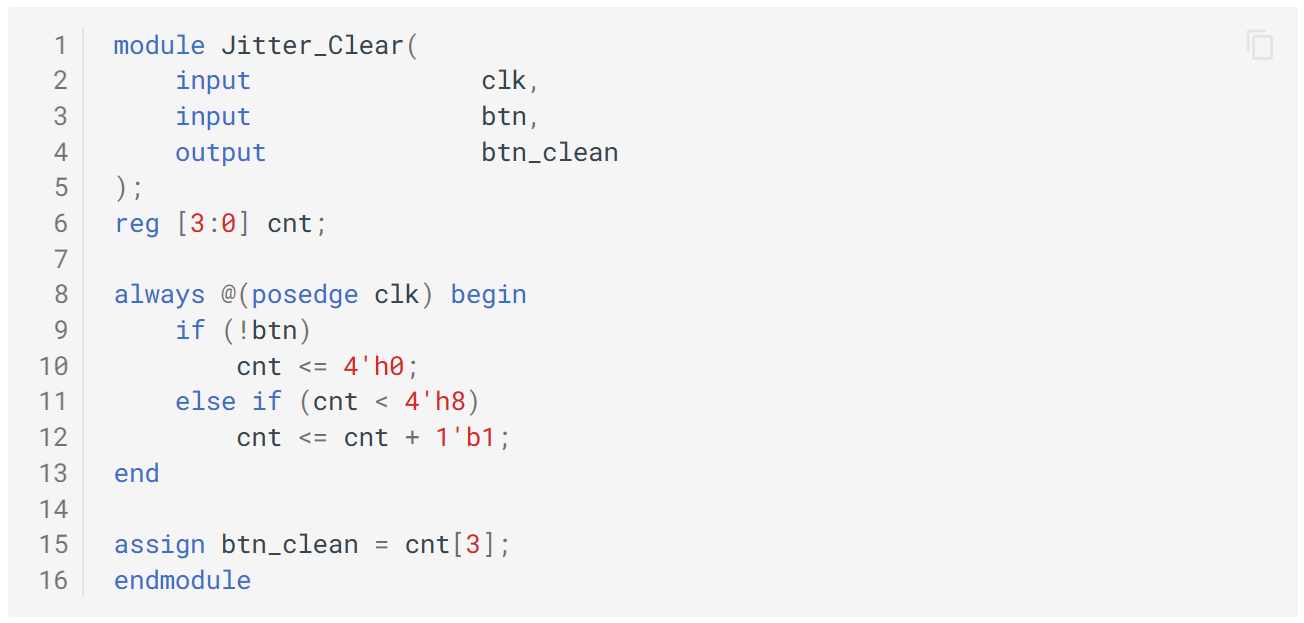
在数字电路设计中，有时需要对输入信号进行整形，将高电平持续时间小于某个阈值的脉冲消除掉，或者将高电平持续时间不同的脉冲信号转换成固定时间脉冲信号。例如：在用 FPGA 开发板的按钮或开关作为输入时，由于其机械特性，在电平转换的瞬间会产生一些毛刺，这些毛刺在用户看来非常短暂，但在 100MHz 的时钟信号下会持续多个周期。以按钮为例，我们希望按钮按下后输入信号会直接从 0 变为 1，按钮松开时会直接从 1 变为 0，但实际情况却并非这样。



对于这种机械式的按钮来说，毛刺很难完全避免，因此，我们需要借助额外的电路结构来达到消除毛刺的目的。

消除毛刺的关键在于区分有效的按钮输入和按钮按下或抬起瞬间的机械抖动。考虑到正常人类的反应速度，按钮按下再抬起的过程，最快也在毫秒以上量级，而抖动一般都在微秒甚至纳秒量级。因此我们可以通过信号电平持续时间的长短来判定是否为一次有效的按按钮操作。通过一个计数器对高电平持续时间进行计时，当按钮输入信号为 0 时，计数器清零，当输入信号为高电平时，计数器进行累加计数，计数达到阈值后则停止计数。

下面的 Verilog 代码实现了基本的消除毛刺功能：

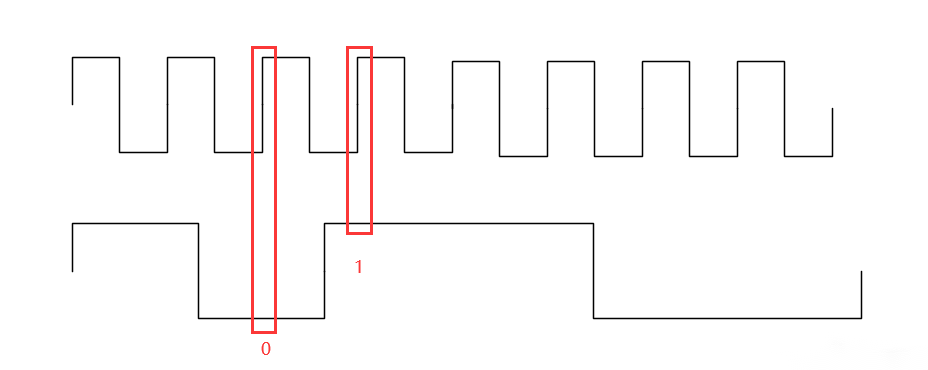


通过仿真，我们发现上述电路能够将持续时间少于 8 个时钟周期的信号毛刺全部滤除，输出变成了一个较为干净的电平信号。作为扩展，我们还可以通过调节计数器的阈值改变该电路的滤除精度。

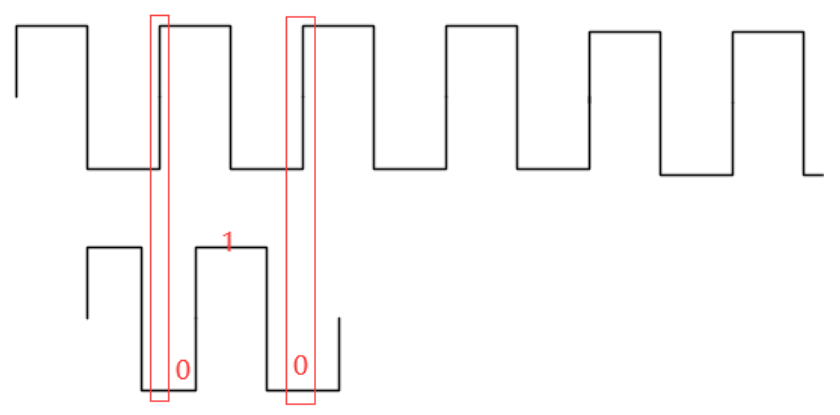
## 2 边沿检测

边沿检测是对于输入信号的基本操作。它将输入信号的高低电平变化转化为一定宽度的脉冲信号，进而实现对后续电路的控制。根据信号变化的不同，边沿检测可以分为上升沿检测、下降沿检测以及双边沿检测。根据输出信号的特征，边沿检测可以分为同步检测与异步检测。

对于同步检测，我们通过比较输入信号在相邻两个时钟周期内的数值从而判断是否发生了电平变化。以下图为例，上方的是 clk 波形，下方的是输入信号波形。在某一个时钟上升沿时，输入信号为低电平（0）；在下一个时钟上升沿时，输入信号为高电平（1）。此时我们就检测到了输入信号的上升沿变化，进而输出一个时钟周期的脉冲信号。

[](https://soc.ustc.edu.cn/Digital/lab3/figs/signals/edge_1.png)

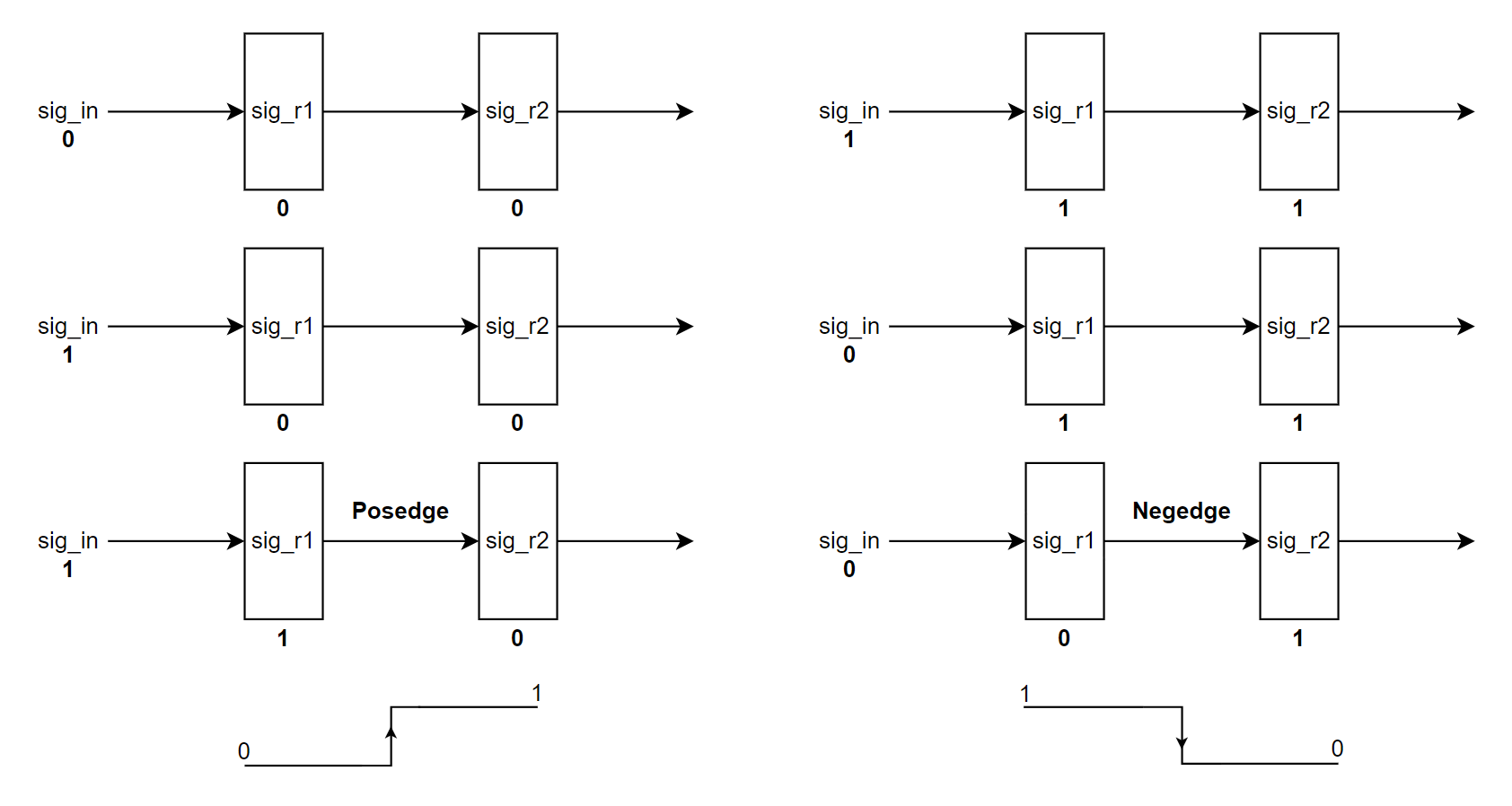
在大多数情况下，同步检测可以满足我们的需要。但当输入信号的变化周期小于时钟周期时，我们就不能使用同步检测了。如下图所示，在相邻两个时钟周期的上升沿时，输入信号均为低电平，因此我们认为输入信号没有发生变化，但实际上输入信号在这个时钟周期内发生了两次电平翻转。此时我们就会使用到异步检测。



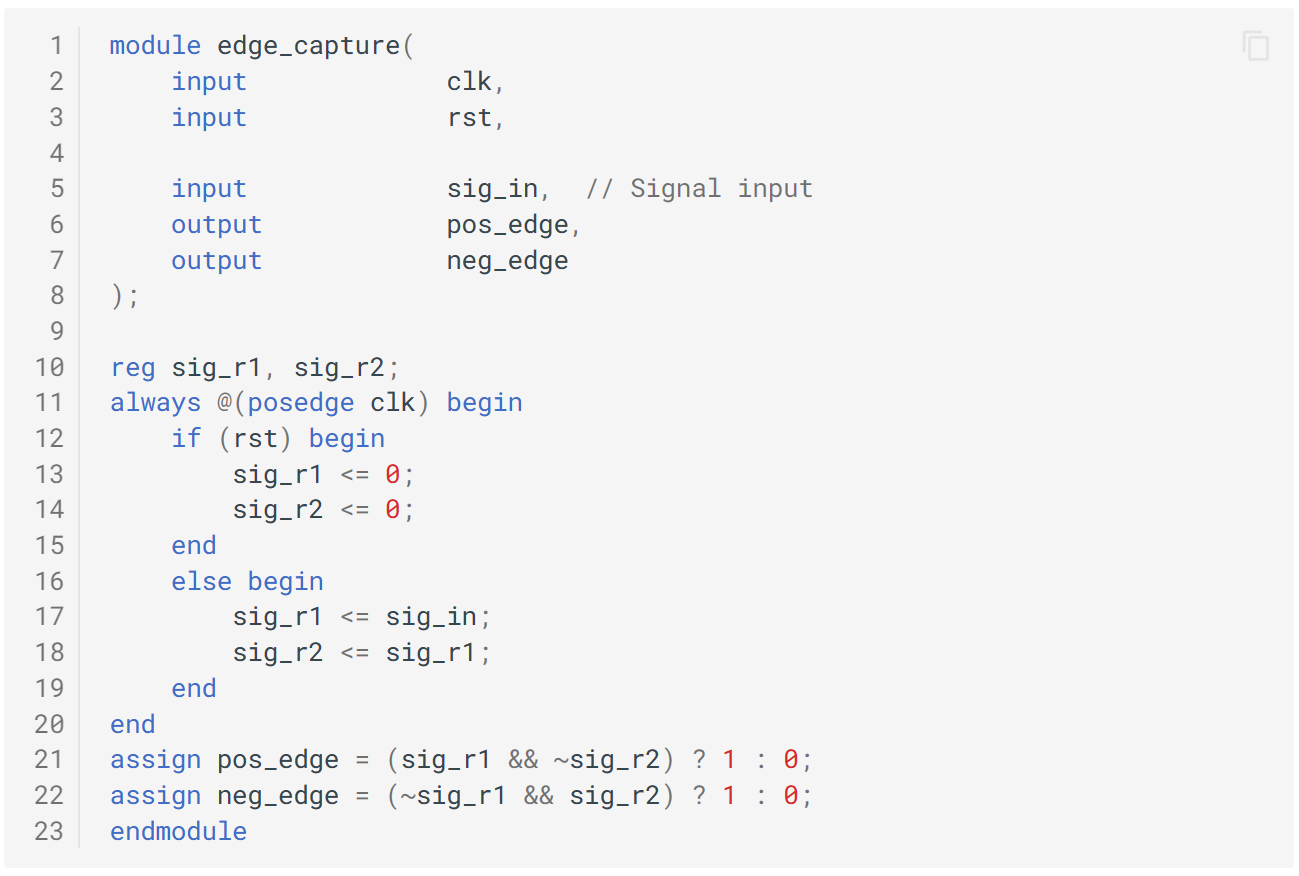
异步检测的原理是直接将输入信号作为敏感变量加入到 always 基本块的头部。这种做法在大部分情况下都是极为不推荐的！在 FPGA 综合时，我们需要避免非时钟信号、复位信号等出现在敏感变量列表中。

所幸的是，在对外设信号进行边沿检测时，小于一个时钟周期的电平变化是极为罕见的，因此同步检测就足以支撑所有的应用场景了。

要实现同步的边沿检测，最直接的想法是两级寄存器法：用第二级寄存器锁存住某个时钟上升沿到来时的输入电平，第一级寄存器锁存住下一个时钟沿到来时的输入电平。如果这两个寄存器锁存住的电平信号不同，就说明检测到了边沿，具体是上升沿还是下降沿可以通过组合逻辑来实现。

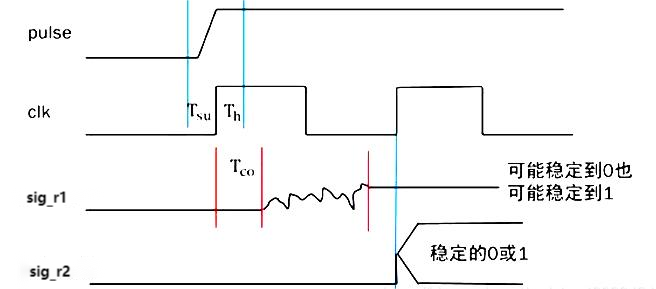
[](https://soc.ustc.edu.cn/Digital/lab3/figs/signals/edge_det.png)

一个典型的 Verilog 程序如下所示：



上面讨论过程都是建立在理想情况下的。在实际的电路情况中（例如 FPGA 开发板上），时序电路的信号均存在建立与保持时间。如果输入信号的变化恰好出现在 clk 的变化边沿中，则第一级寄存器采集到的信号可能并不是明确的高低电平，而是一种中间的结果。此时，第一级寄存器的输出会进入到亚稳态，进而传递给后续的寄存器，影响整个电路的工作情况。

如下图所示，图中 Tco 为第一级寄存器 sig\_r1 的状态建立时间（即 clock to output），一般情况下，亚稳态的决断时间（即从进入亚稳态到稳定下来的时间）不会超过一个时钟周期，因此在下一个 clk 上升沿到来之前，sig\_r1 已经稳定下来（可能稳定到 0 也可能稳定到 1），这样第二级寄存器就会采集到一个稳定的状态，从而把亚稳态限制在第二级寄存器之前，保证了整个电路输出的稳定性。

[](https://soc.ustc.edu.cn/Digital/lab3/figs/signals/edge_4.png)

所以，为了保证系统的稳定性，我们建议大家在实现边沿检测时，均使用三级寄存器的方法。在对系统稳定性要求较高的数字系统中，可以采用更多级的寄存器来减小亚稳态发生的概率，提高系统稳定性。

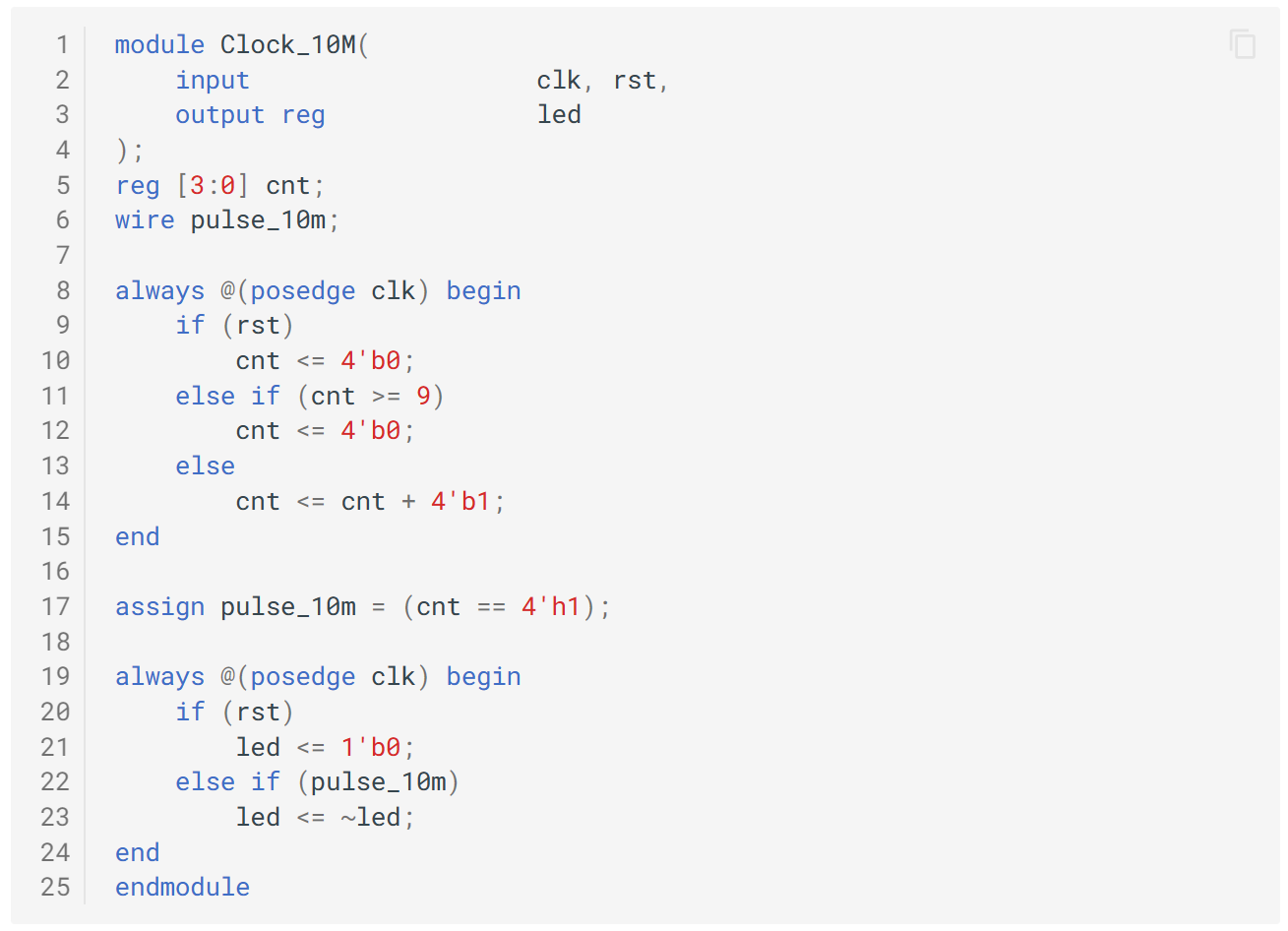
## 3 时钟分频

### 3.1 计数器分频

我们知道，FPGA 开发板的 FPGA 芯片 E3 管脚连接了一个 100MHz 频率的时钟晶振，可用作时序逻辑电路的时钟信号。如果我 们需要一个其它频率的时钟信号，例如 10MHz，应该怎么办呢？一般的做法是通过计数器产生一个低频的脉冲信号，然后再将该脉冲信号控制其他逻辑的控制信号。

计算可知，10MHz 时钟频率为 100MHz 时钟频率的十分之一。为此我们可以设计一个计数器，从 0 开始每 10^−8 秒增加一。当计数器数到 9 时，在下一个周期将其复位至 0。这样就实现了一个 0~9 十个状态的循环计数。

下面的代码让 led 信号以 10MHz 的频率闪烁。



上面的代码中 pulse\_10m 并不是一个均匀的时钟信号。不难看出在一个 10MHz 时钟周期内，其有 9×10^−8 秒为低电平，仅有 10^−8 秒为高电平。但我们可以依据其高电平进行信号驱动，对应的时钟周期依然为 10^−7 秒。

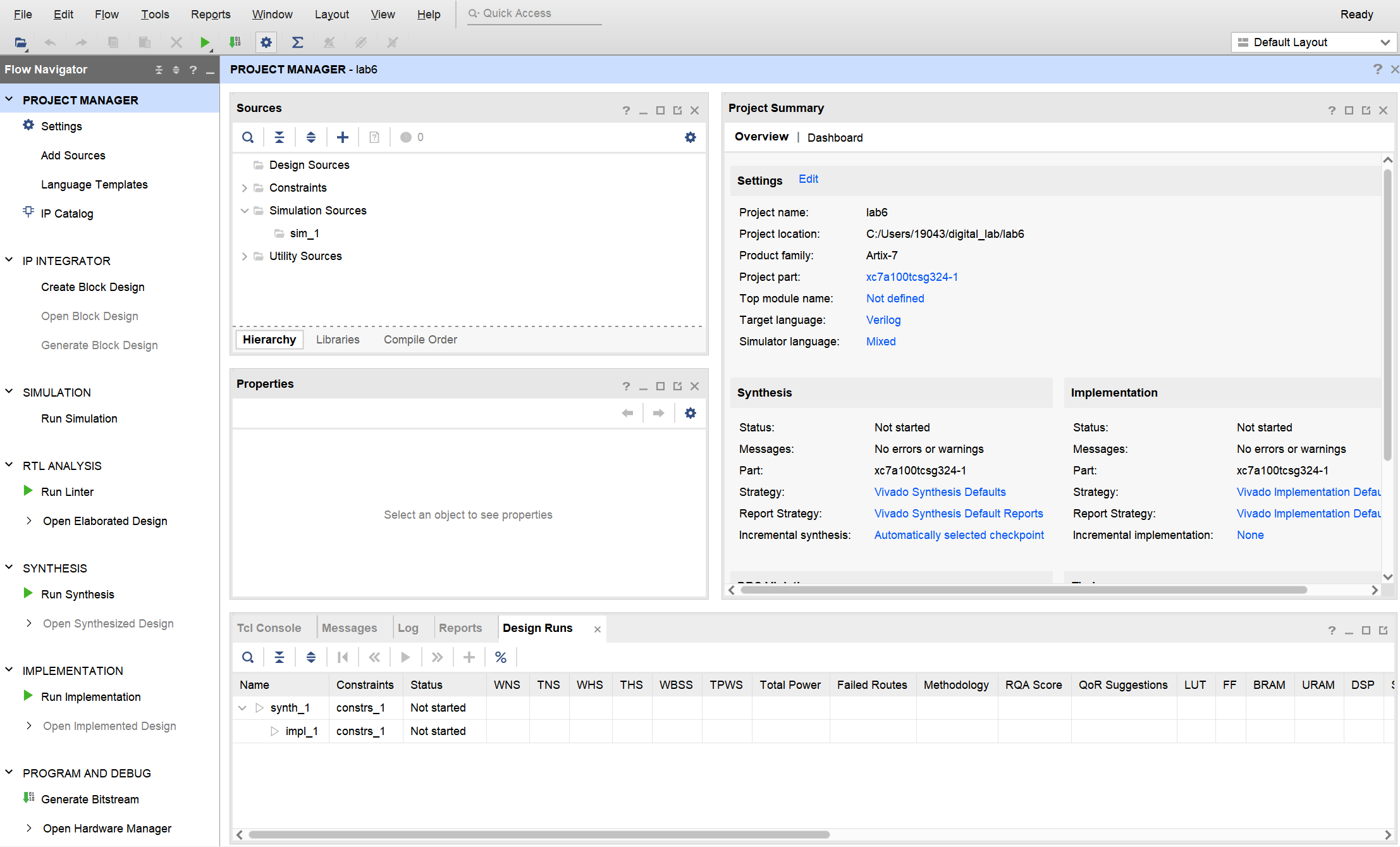
然而，这种分频方式有一个缺点，即无法在 Vivado 的时序仿真中被识别为时钟信号，因而无法检测出可能的时序问题。

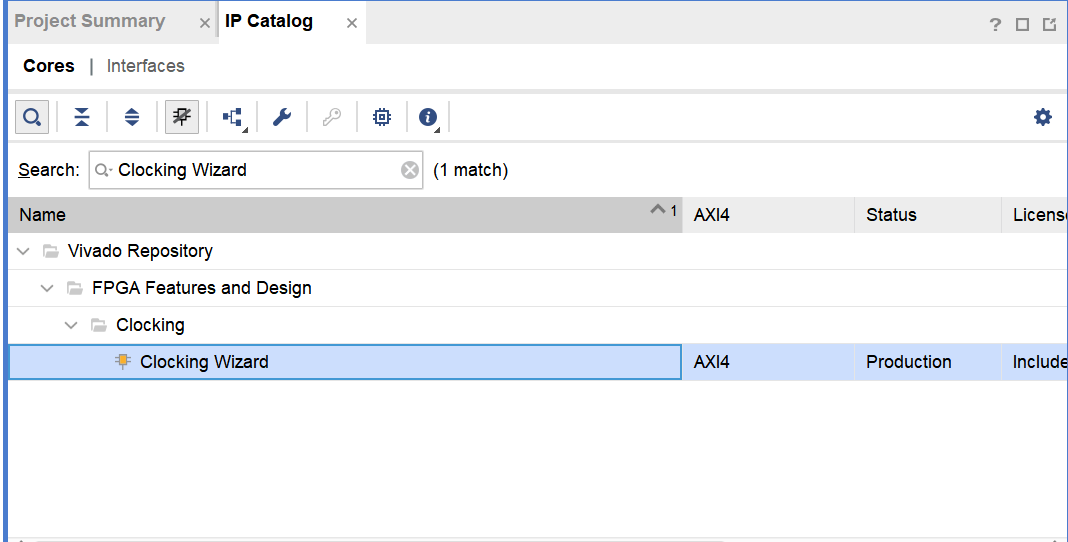
### 3.2 IP核分频

在 FPGA 开发中，有很多常用功能的模块是不需要自己开发的，用户可以复用第三方开发好的模块，这种模块被称为 IP 核（Intellectual Property Core）。在大规模的数字系统设计中，IP 核可重复利用，大幅降低负担及成本，是芯片设计行业不可或缺的重要技术。你可以将 IP 核简单理解为具有较为复杂功能的、封装好的模块。

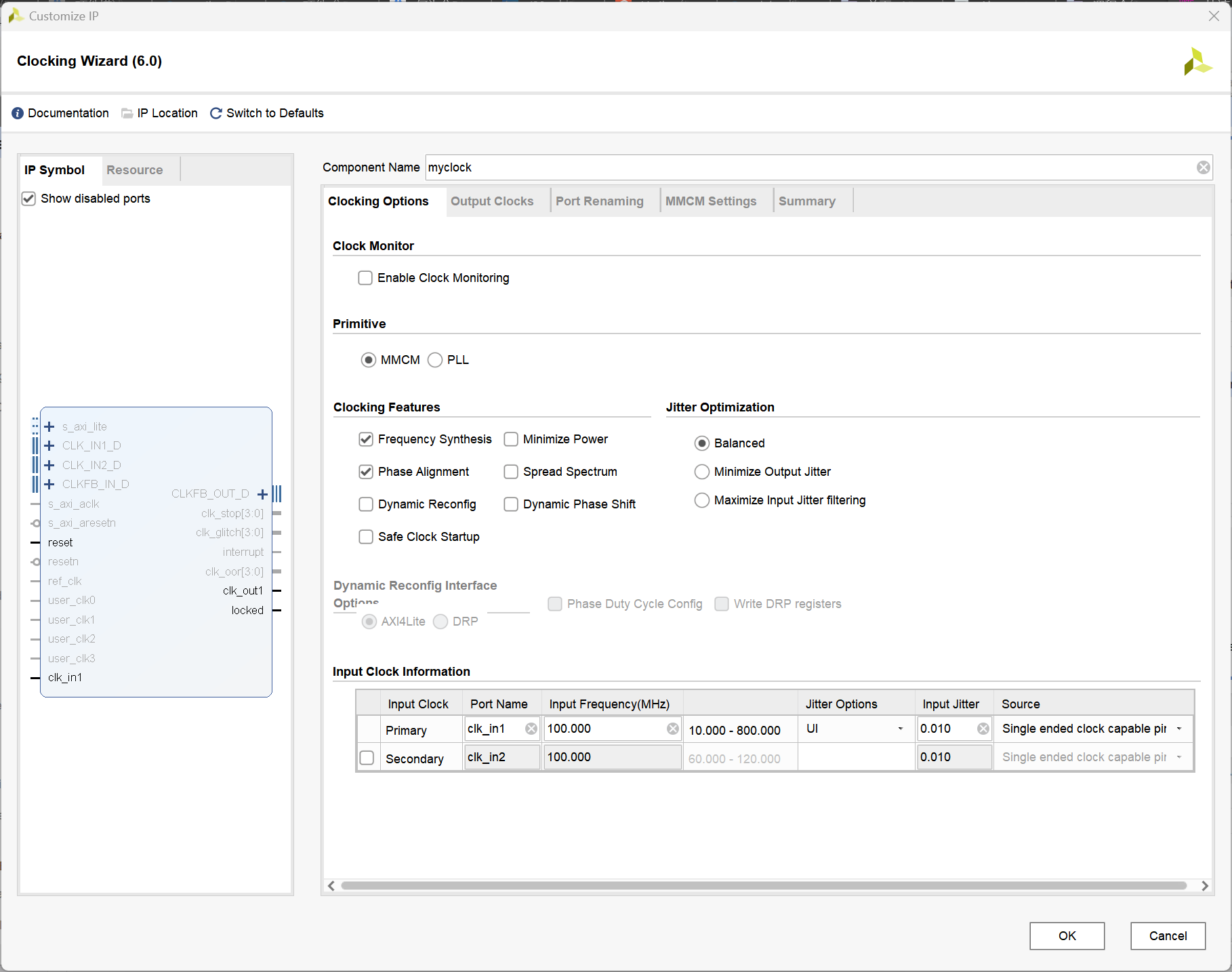
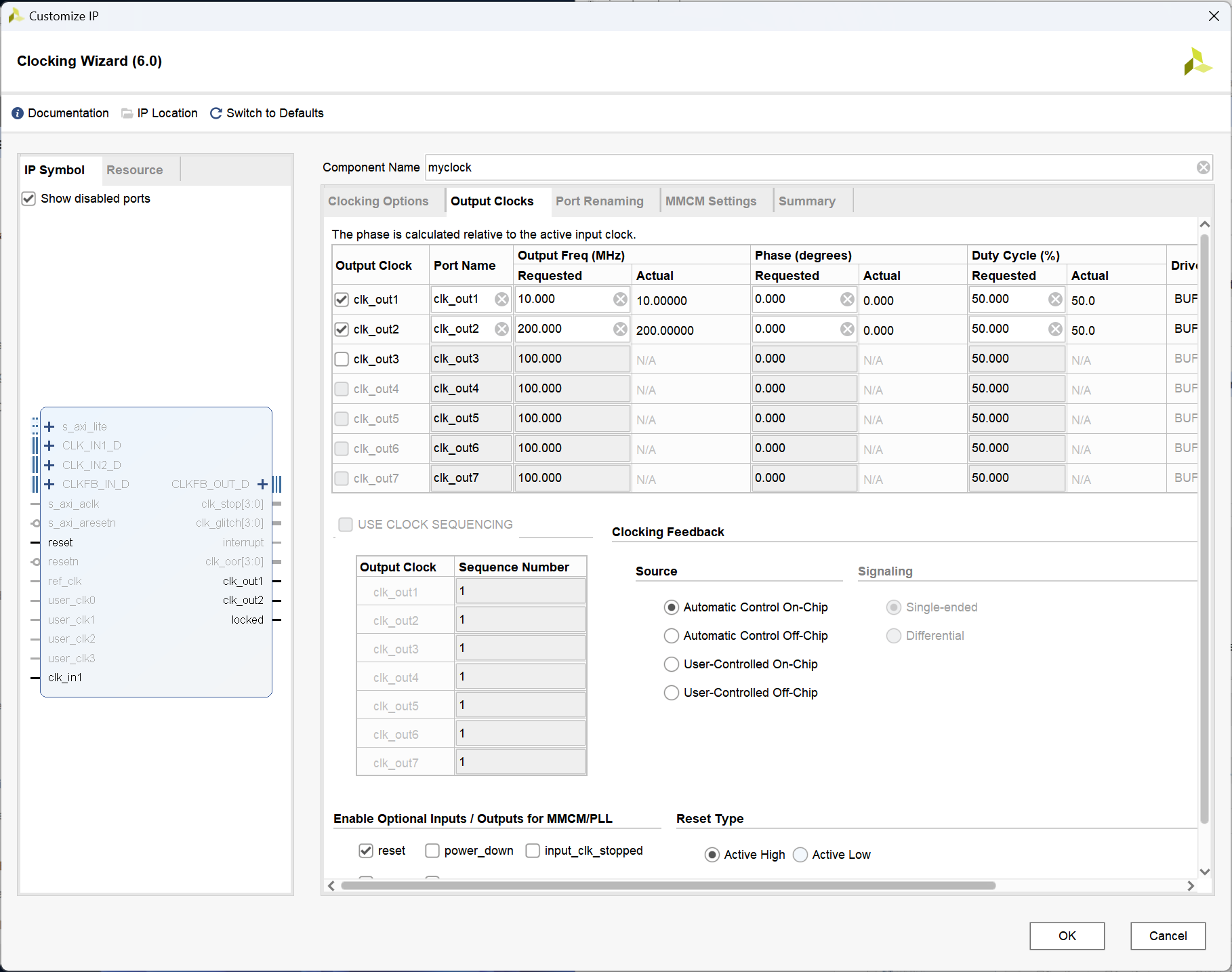
Vivado 给我们提供了时钟 IP 核，支持其他频率的时钟设定。你可以按照如下的步骤进行设置。

打开 Vivado 进入项目界面。点击 IP catalog，在搜索框中输入 "Clocking Wizard"，并在下方窗口中双击 Clocking Wizard。

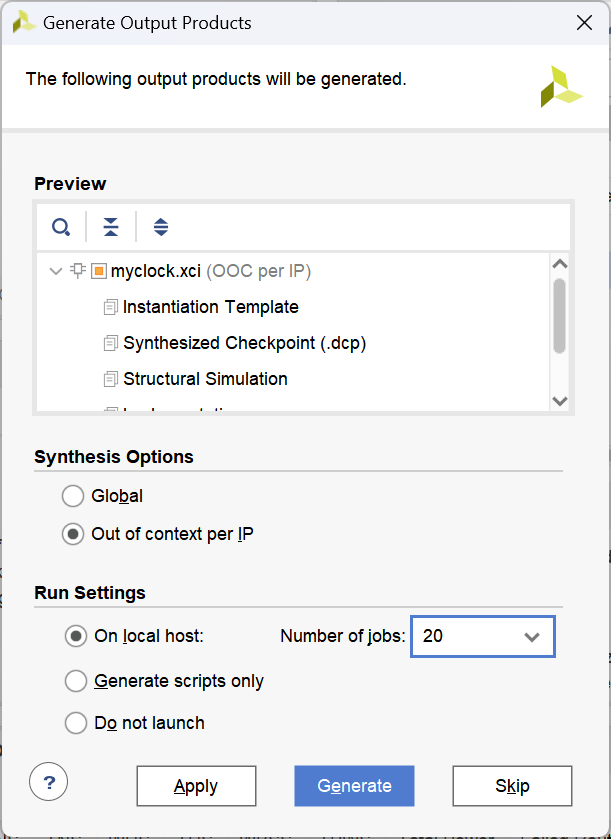
[](https://soc.ustc.edu.cn/Digital/lab3/figs/signals/vivado_base.png)

[](https://soc.ustc.edu.cn/Digital/lab3/figs/signals/vivado_clock_sel.png)

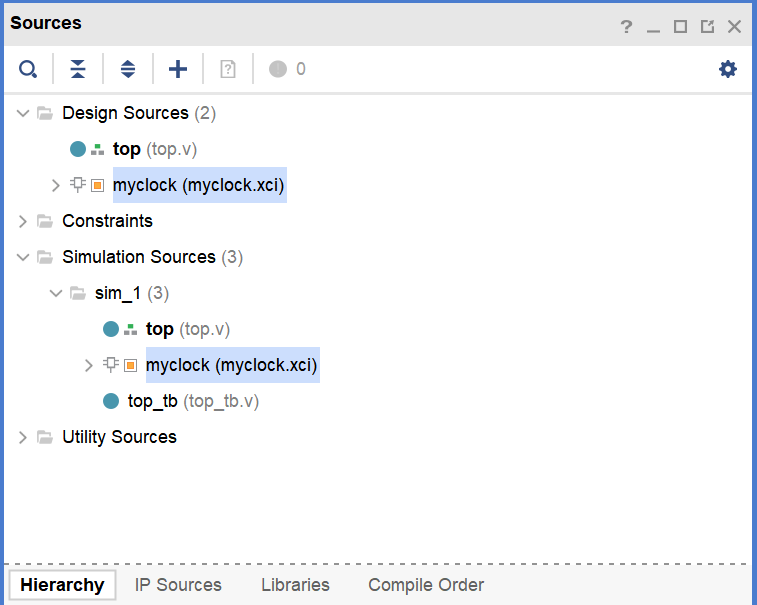
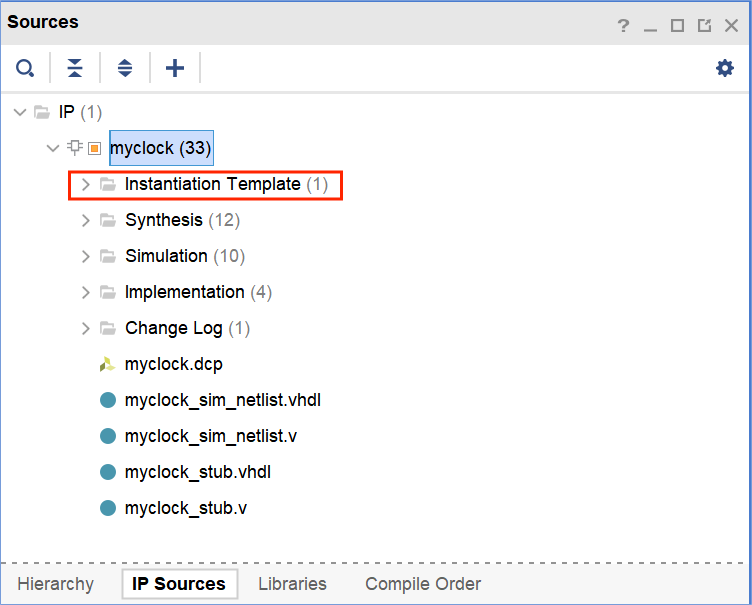
在打开的设置界面中，将输入时钟频率设置为 100MHz；输出时钟设定两个，分别为 10MHz 和 200MHz。将 IP 核的名字更改为 "myclock"。

[](https://soc.ustc.edu.cn/Digital/lab3/figs/signals/vivado_clock_setting_1.png)[](https://soc.ustc.edu.cn/Digital/lab3/figs/signals/vivado_clock_setting_2.png)

确认无误后，单击 OK，随后单击 Generate 以生成 IP 核。

[](https://soc.ustc.edu.cn/Digital/lab3/figs/signals/vivado_clock_gen.png)

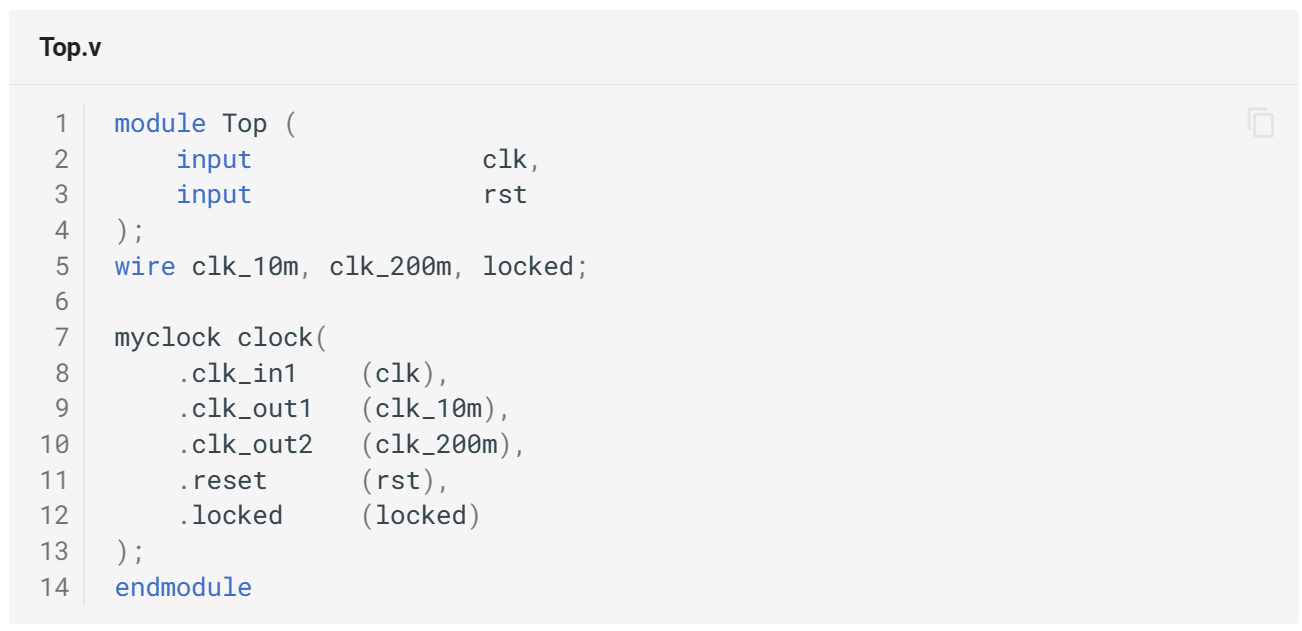
此时，生成的 IP 文件可在“工程目录/工程名.srcs/sources\_1/ip/IP 核名称/IP 核名称.v”找到。你可在设计文件中像调用其它模块一样使用该 IP 核,使用时只需要了解 IP 核的功能及端口信号的含义及时序，而不用关心模块内部的具体实现。

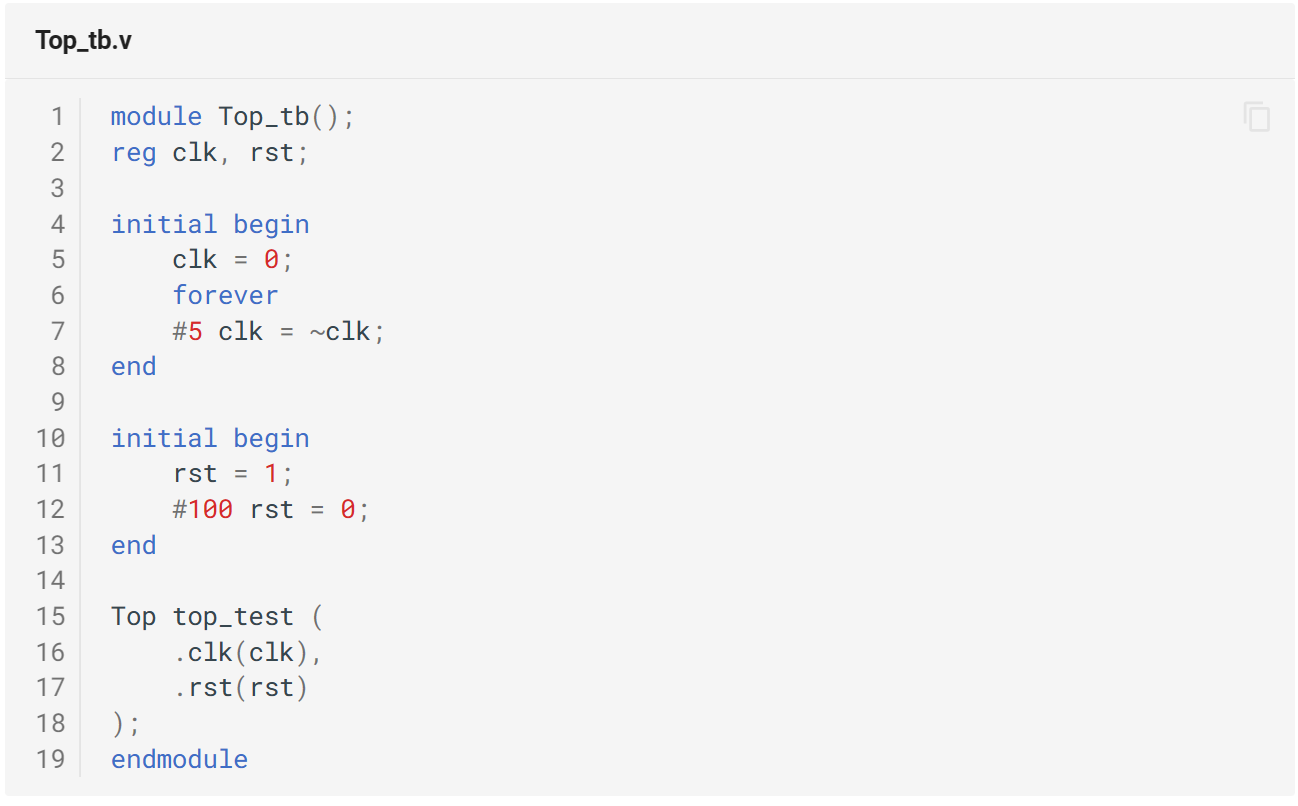
[](https://soc.ustc.edu.cn/Digital/lab3/figs/signals/vivado_clock_src_1.png)[](https://soc.ustc.edu.cn/Digital/lab3/figs/signals/vivado_clock_src_2.png)

在 IP Sources 分窗口下单击 Instantiation Template，可以在 myclock.veo 文件中可以找到该 IP 核的例化格式代码：

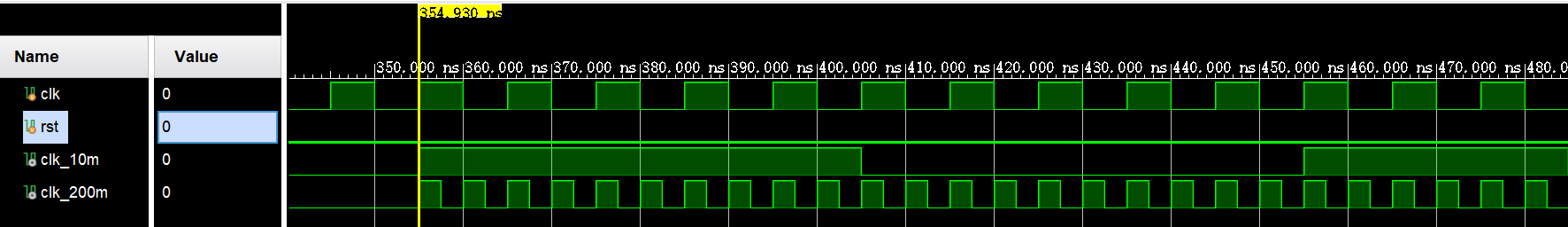


同样地，我们创建模块文件 Top.v 和对应的仿真文件 Top\_tb.v，在其中分别输入如下的代码：





完成后运行仿真，在波形窗口拖入信号 clk\_out1 和 clk\_out2，最终得到的波形如下图所示。可以看到，时钟信号已经被正确生成。

[](https://soc.ustc.edu.cn/Digital/lab3/figs/signals/vivado_clock_sim.png)

# 思考与练习

1. 在教程中我们使用两级寄存器进行边沿检测。为了提高系统的稳定性，请在此基础上完成使用三级寄存器边沿检测的 Verilog 代码。代码框架如下：

