1.1 ILA 工具简介

在 FPGA 开发中，当完成代码设计后，为验证代码的正确性和各种不同条件下的可靠性，设计师往往优先想到通过逻辑仿真进行相关验证。使用逻辑仿真进行验证虽然可以周密的考虑给出不同输入条件下输出结果或交互结果，但也有其相对局限性：

使用仿真需要设计人员写出 testbench 代码，从而增加代码的书写量，随之而产生提高验证工作的门槛和排除错误的工作量等一系列问题。特别是对于刚刚步入 FPGA 殿堂的初学者，有时候很难对 testbench 的设计方法有准确的把握，甚至将 testbench 的书写规则和设计源码的规则混淆，从而对学习源码设计产生适得其反的效果。

使用仿真进行逻辑验证还有另一个局限性，其体现在：有些驱动模块的模型无法获取的情况下，仿真几乎是一项不可能完成的任务。比如，在没有 DDR2，DDR3，摄像头模组等代码模型的条件下，设计的驱动模块缺乏能够模拟真实外部器件的数据交互环境，从而很难实现仿真的验证过程。

最后，仿真大多适用于以时间为标尺，对条件变化有清晰的时间节点预设的验证需求。这就导致对工程的验证很难形成 FPGA 芯片、计算机和人的三者互动，也很难通过捕捉随机触发条件而进行验证，从而使触发条件的选择，受较大的局限性。

能够较好解决以上问题的一种方案，是考虑使用逻辑分析仪。而采用传统的逻辑分析仪，又存在额外的设备采购开销。得益于 FPGA 的硬件定制自由属性，FPGA 生产设计公司从自身发掘潜力，并配套以软件开发环境，推出了在一定程度上能够代替逻辑分析仪的在线调试工具。

FPGA 两大主流厂商都有自己的在线调试工具。Intel（Altera）的 Quartus 开发工具自带 SignalTap 在线调试工具，AMD(Xilinx)的 VIVADO 开发工具自带 ILA 在线调试工具。使用在线调试工具，能够使用计算机代替逻辑分析仪的绝大多数功能，实现代码设计、编译、仿真、调试及板级验证工具的一体化集成，从而大大减少硬件电路设计后的验证周期。

所以，作为一个优秀的 FPGA 工程师，根据实际情况进行开发和调试工具的选择和结合，是一项十分重要的技能。本章内容正是基于这一需求，推出了 ILA 工具的使用方法详细展示，以此向读者介绍 ILA 工具的使用方法和技巧。

为了让讲解既更加贴近实际使用环境又不至于太难上手，我们以 ACX720 开发板作为实验平台搭建了如下工程作为范例：工程核心部分包含 4 个端口和 3 个模块单元,如下图：

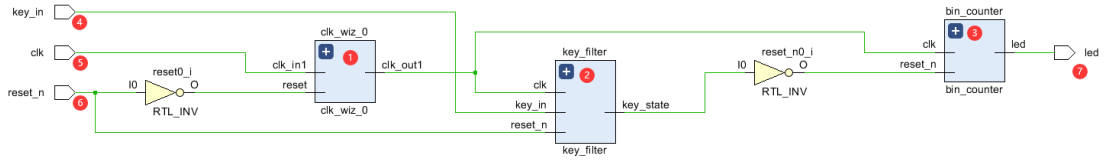


图 1.1-1 范例工程 RTL 框图

① **clk_wiz_0：** pll 锁相环，设计该环节可以通过调整输出验证 ILA 在不同频率下的工作状况

② **key_filter：** 按键消抖模块，当按键按下后，key_state 为低电平，取反传递给 bin_counter 模块的 reset_n 复位管脚后释放 bin_counter 模块的复位信号。

③ **bin_counter：** 计数器控制 led 翻转模块。当按键按下后释放复位，计数器计数开始，同时实现计数满后 led 翻转。

④ **key_in：**输入接口，按下开发板按键后，按键由高电平转为低电平传递到 FPGA。(连接到开发板 S0，由于人按下按键会有机械抖动，故需要配合消抖模块使用)

⑤ **clk：** 从晶振接入到 FPGA 的主时钟。(时钟频率为 50M)

⑥ **reset_n：** 复位信号(连接到开发板 S4 按键)

⑦ **led：** 开发板的 LED 输出。（连接到开发板 led0）

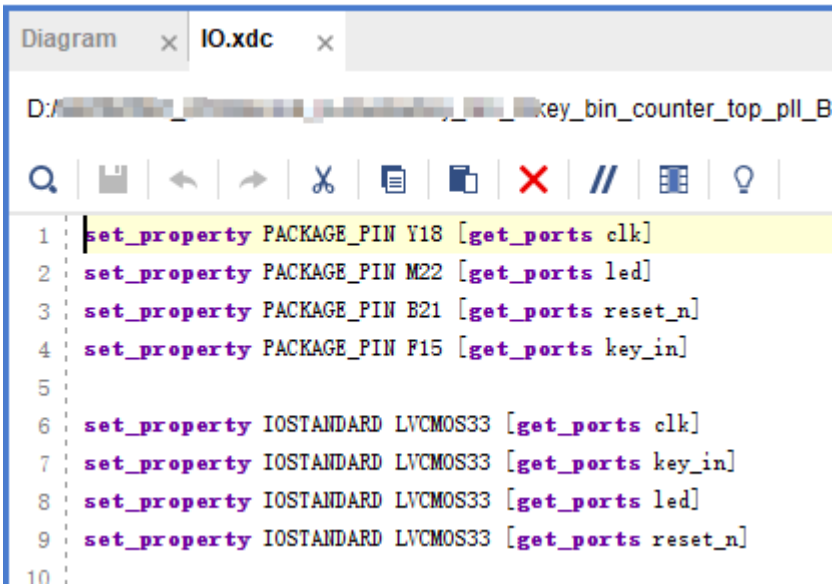


图 1.1-2 本工程涉及管脚在 ACX720 开发板的绑定

在后续内容中，我们将均以此工程作为背景和蓝本讲解 4 种不同应用需求下创建 ILA 工具的方法，并详细讲解 ILA 调试面板的使用和调试方法。相信通过本课程的学习，各位读者能够全面掌握 ILA 的使用过程。

1.2 使用 IP 核创建 ILA 调试环境

本节内容将讲解使用 IP 核创建 ILA 调试环境的操作方法。这种方法虽然操作流程稍显复杂，但其通用性强，作为 ILA 初学者，可以优先考虑完全掌握本方法。

回到我们前面介绍的工程范例，打开工程后，可以看到工程架构如下：

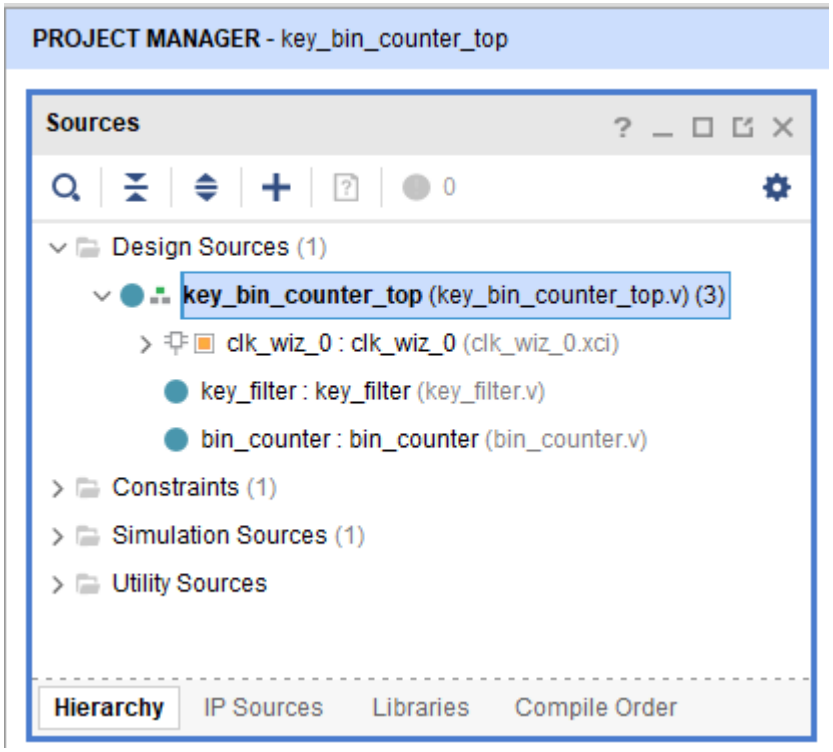


图 1.2-1 工程打开后的原始架构

前面我们已经介绍了工程的设计背景，这里，我们直接使用 IP 核创建 ILA 调试环境。

1.2.1 创建 ILA IP 核

点击 Flow Navigator 窗口中 IP Catalog 选项或 Window 菜单下 IP Catalog 选项，启动 IP 筛选器。

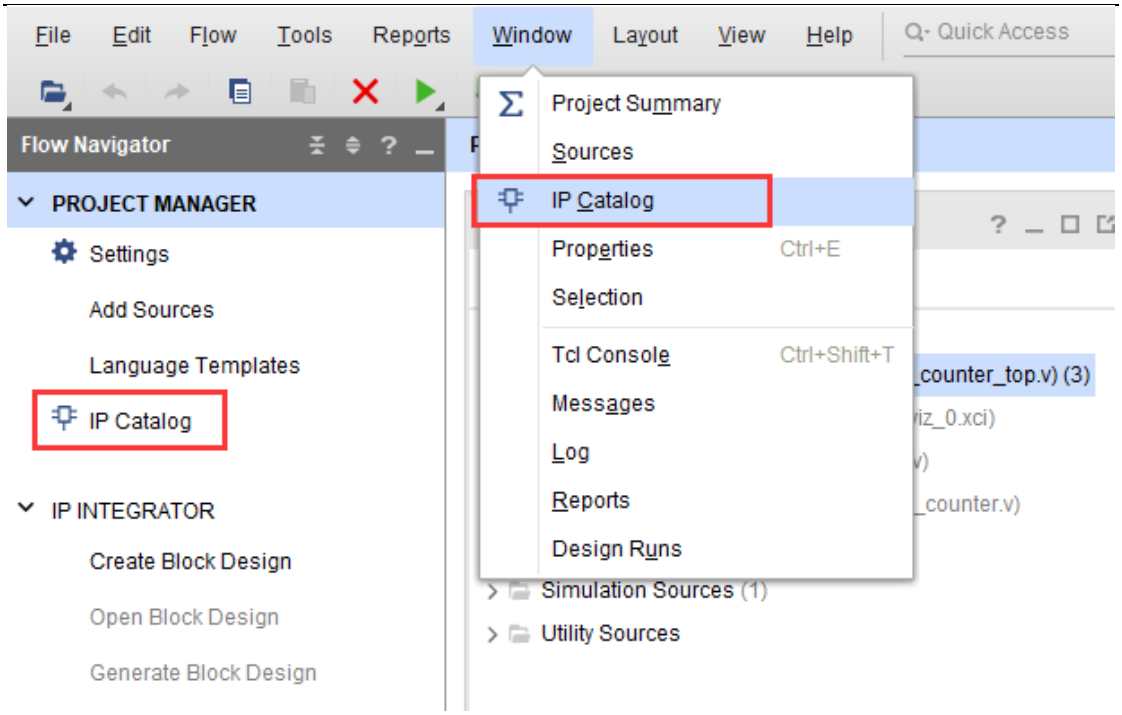


图 1.2-2 点击 IP Catalog（任选 1 种方法）

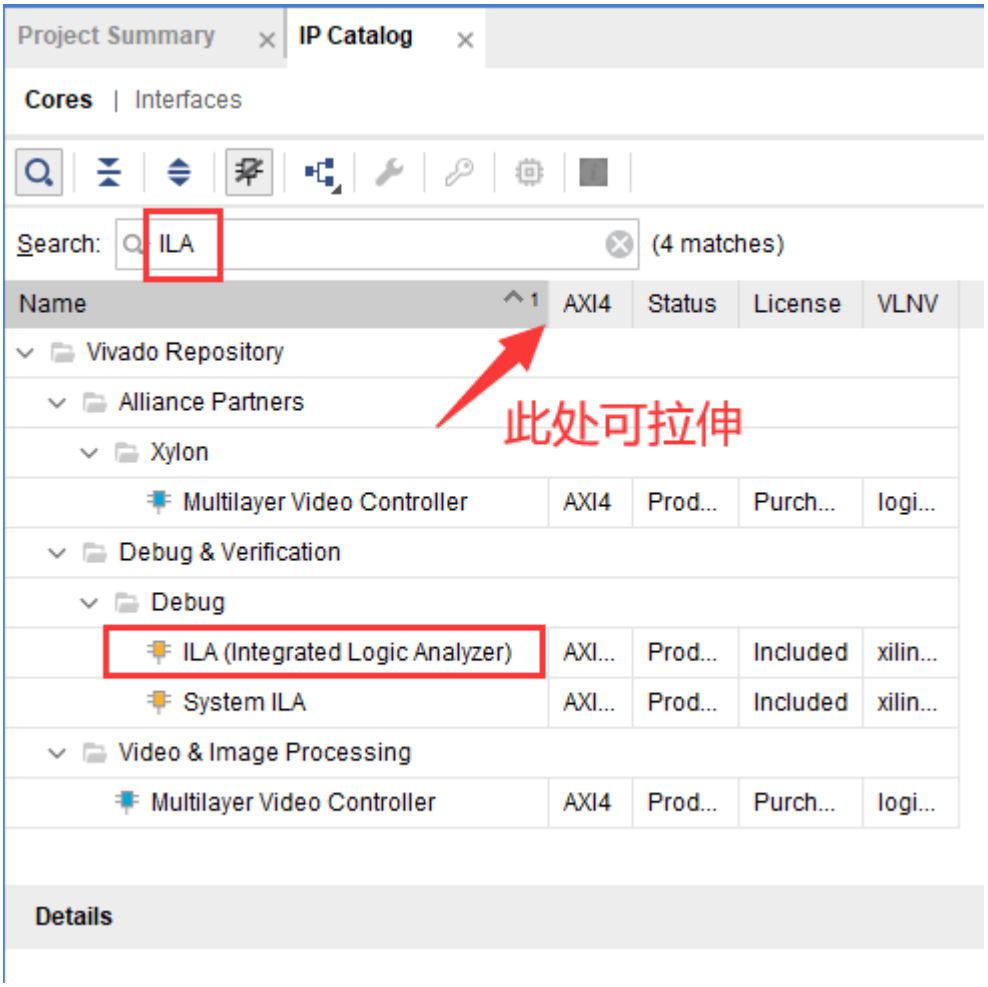


图 1.2-3 筛选 ILA IP 核

这样，创建 ILA IP 核的工作，就完成了。

1.2.2 ILA IP 核的配置界面介绍

在点击 ILA IP 核的启动界面后，VIVADO 软件会带你来到 ILA IP 核的配置界面。配置界面主要窗口可以分为两个页面。

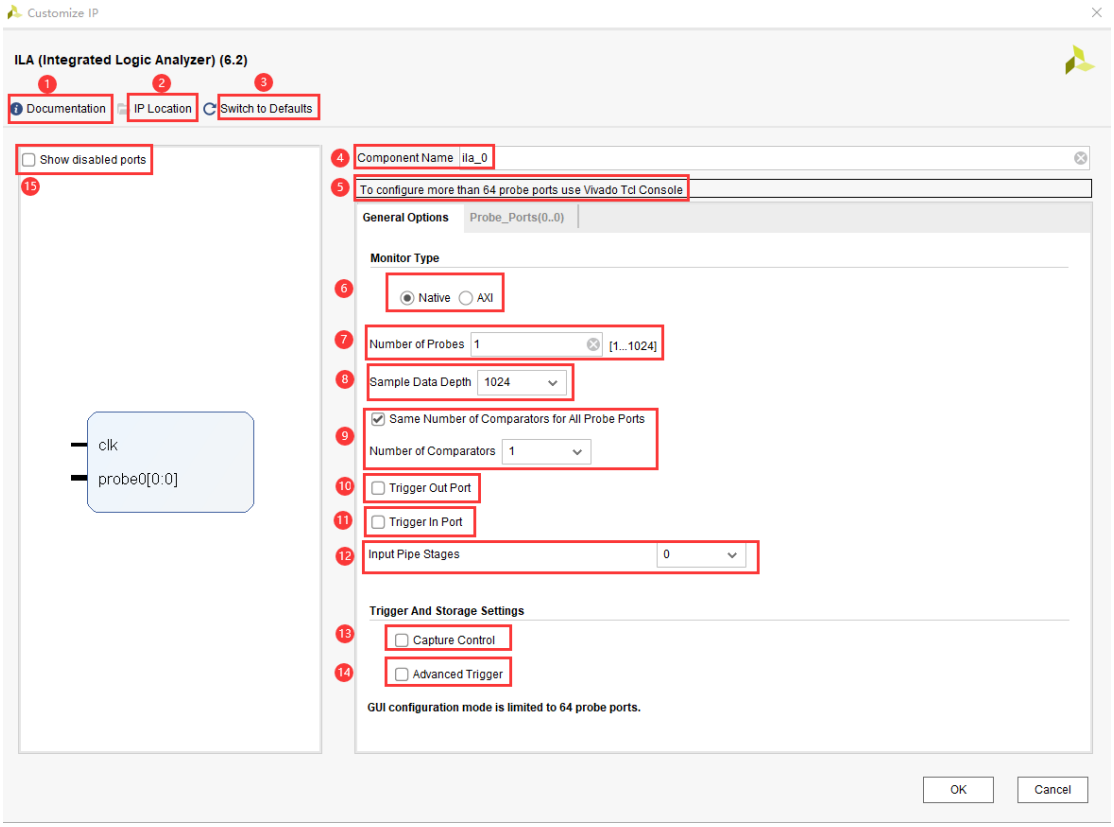


图 1.2-4 ILA IP 设置窗口 1

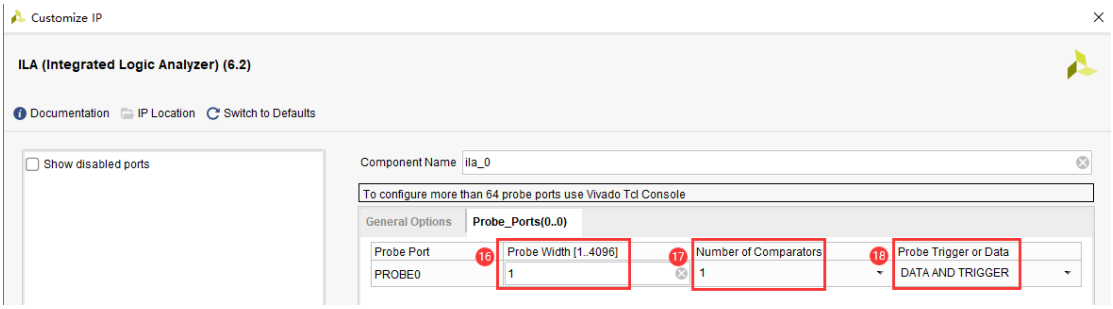


图 1.2-5 IP 设置窗口 2

窗口具体设置介绍如下。

（1）Documentation：IP 相关文档入口，点击后出现如下内容

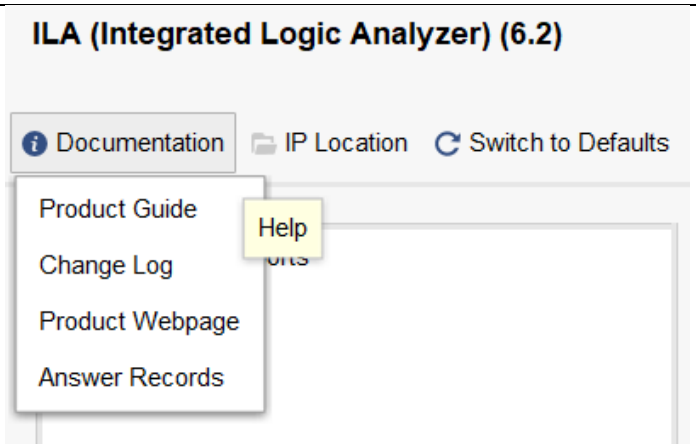


图 1.2-6 Documentation 下可选项窗口

①Product Guide 是 IP 手册查看入口，点击可自动跳转到 Xilinx 官方文档 DocNav 软件该 IP 手册的界面（前提是有安装 DocNav）；

②Change Log 是 IP 版本更新记录，点击可以看到如图 1.2-7 所示的 ILA IP 更新记录。

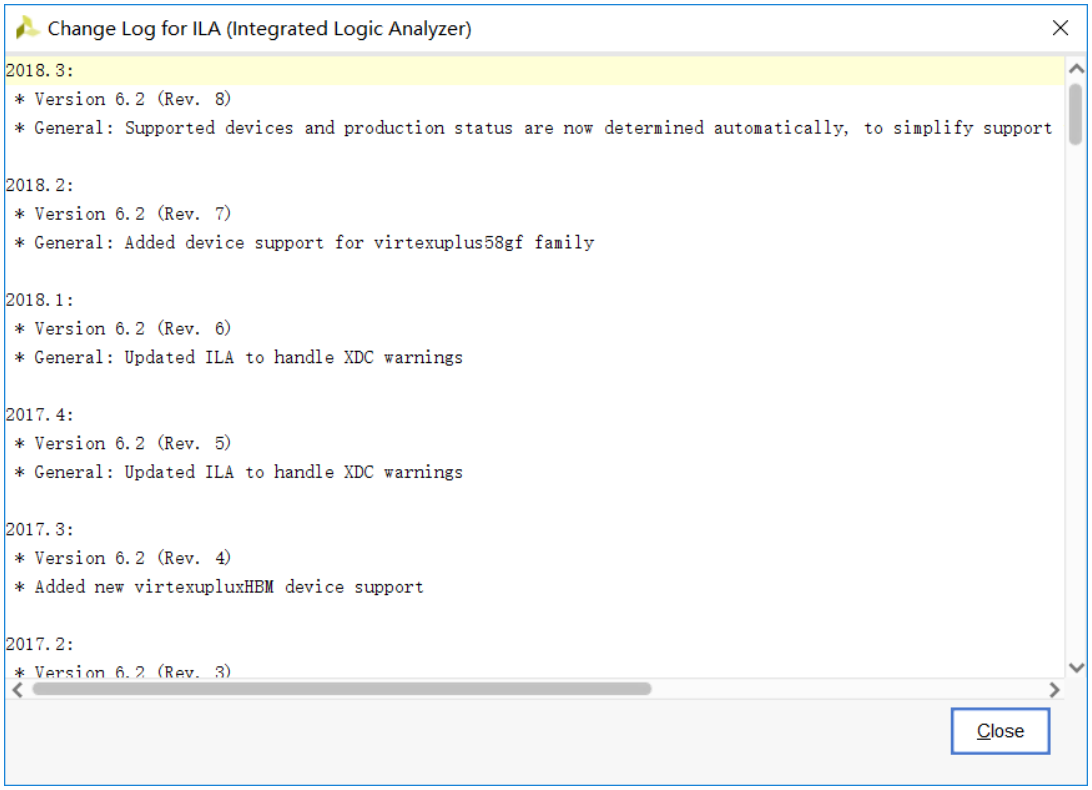


图 1.2-7 ILA IP 更新记录

③Product Webpage 是 IP 相关介绍的网页版，点击可跳转到如下图所示的 Xilinx 官方有关该 IP 介绍的网站。

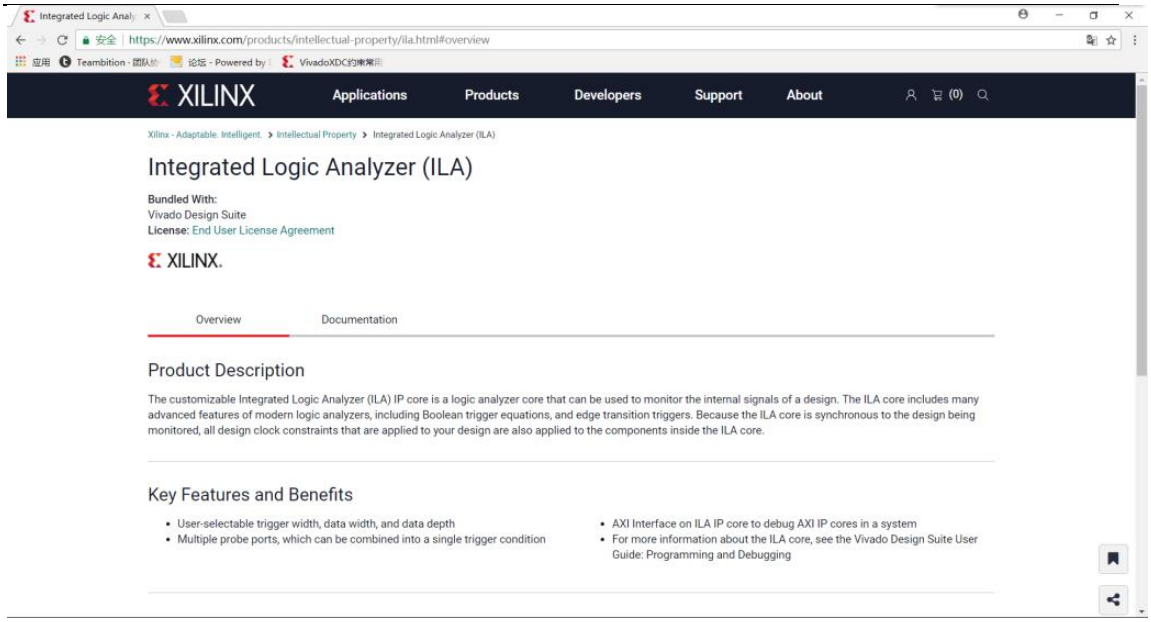


图 1.2-8 IP 相关介绍的网页界面

④Answer Records 是与 IP 相关的 Xilinx 官方疑问解答记录网页，点击可跳转到如下图所示的网页。

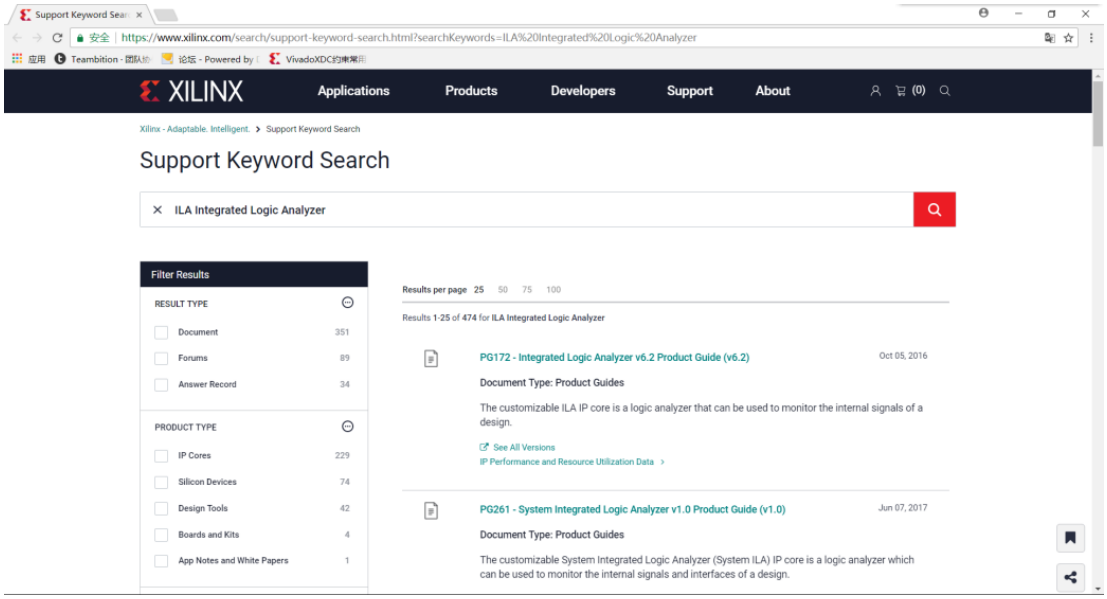


图 1.2-9 IP Answer Records 官方网页

（2）IP Location：设置 IP 的存放路径入口，点击出现如下图所示窗口，在窗口里可以通过点击“…”设置更换存放路径，默认是存放在工程路径下的.....<工程名>.srcs\ sources_1\ ip，这里我们就保持默认。

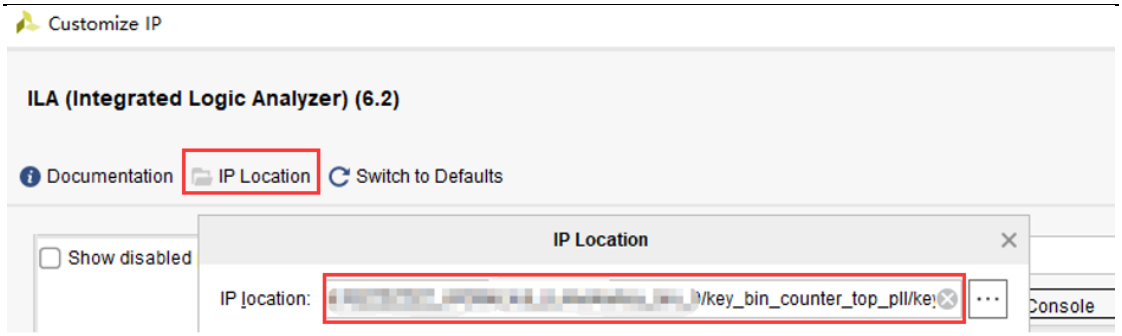


图 1.2-10 IP 路径设置窗口

（3）Switch to Default：点击后所有的设置恢复到默认值。

（4）Component Name：设置生成 IP Core 的名称，这里保持默认设置。

（5）这里是一个提示，提示通过该界面设置最多可设置 64 个探针，如果想设置更多的探针需要使用 Tcl 脚本命令去设置，这里就不做详细描述，想要知道具体用法的，可以查询 IP 手册。

（6）ILA 探针接口类型设置

- Native：常规普通接口模式
- AXI：AXI 接口模式，用于调试 AXI 接口信号

（7）Number of Probes：探针数量设置，在 GUI 界面最大可设置 64 个，如果需要调试查看的内部信号超过 64 个，可以通过 TCL 脚本去生成 IP Core（这个在（5）处有这个提示，这里不做详细描述，具体读者可以查看 IP 使用手册），或着可以通过生成多个 ILA IP Core 去调试更多信号。根据前面我们的需求，我们希望对信号 cnt 和 key_state 信号进行抓线调试，这里需要设置探针数量为 2。

（8）Sample Data Depth：采样数据深度，设置的数值越大，采样的数据越多，看到的波形数据越多，但是最终占用的资源也会越多，并不是设置的越大越好。如下图所示，从下拉框也能看出最大也只能设置为 131072，这个根据实际需求进行合适的设置即可，我们这里选择设置 4096。

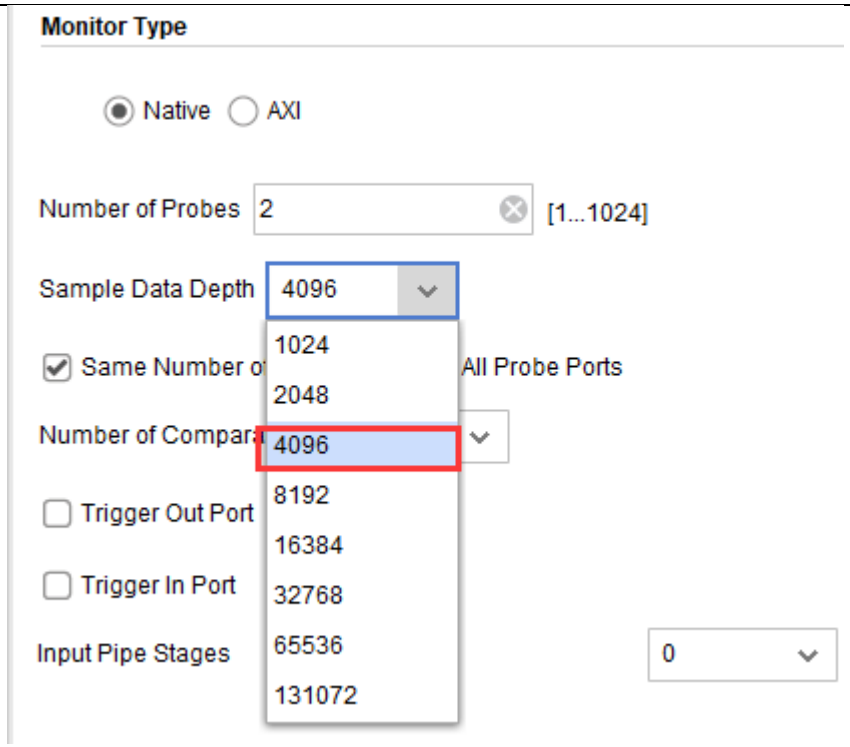


图 1.2-11 设置采样深度窗口

（9）Same Number of Comparators for All Probe Ports：这里是设置相同探针接口的 Comparators 的个数，不勾选，下面的 Number of Comparators 就会消失。默认勾选，后面的参数也保持默认，我们使用其基本功能。

（10）Trigger Out Port：触发输出端口，可用于 ILA 模块的级联或一些高级功能，具体使用参考 IP 手册，这里保持默认不勾选。

（11）Trigger In Port：触发输入端口，可用于手工设置添加触发信号或进行 ILA 模块的级联或一些高级功能，具体使用参考 IP 手册，这里保持默认不勾选。

（12）Input Pipe Stages：设置待探测信号打拍次数，如下图所示，可设置数值 0~6，一般情况下，采样时钟和探测信号是一个时钟域下，这里可以默认设置为 0 即可。

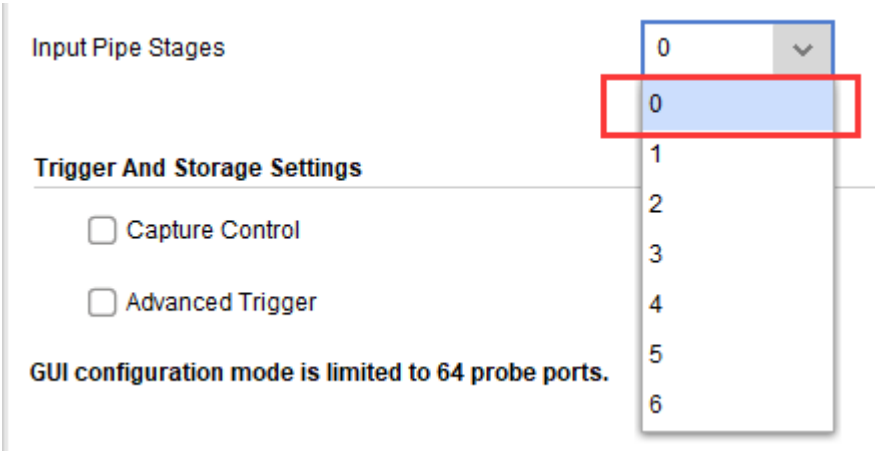


图 1.2-12 Input Pipe Stages 设置窗口

（13）Capture Control：这里勾选后可在调试查看波形窗口进行对 Capture mode 的设置，如果不勾选后面有关 Capture mode 就不可修改，这里勾选，具体使用在后面讲解。

（14）Advanced Trigger：这里勾选后可在调试查看波形窗口进行对 Trigger mode 的设置，如果不勾选后面有关 Trigger mode 就不可修改，这里勾选，具体使用在后面讲解。

（15）Show disabled ports：在 IP 核的模拟示意图中展示出未使用信号引脚。

（16）Probe Width：探针数据信号的位宽设置，我们需要对一个 25bit 和 1bit 信号进行在线观察，这里将 Probe0 位宽设置为 25，Probe1 位宽设置为 1。

（17）Number of Comparators：如果已经勾选（9）处，这里就不可设置，如果没有勾选，这里就可以设置。保持默认即可。

（18）Probe Trigger or Data：对探针设置触发器或数据，如**错误!未找到引用源。**所示，有 3 种可选项。

- DATA AND TRIGGER：既是数据又可作为触发条件；
- DATA：仅作为数据，不可作为触发条件；
- TRIGGER：仅可作为触发条件。

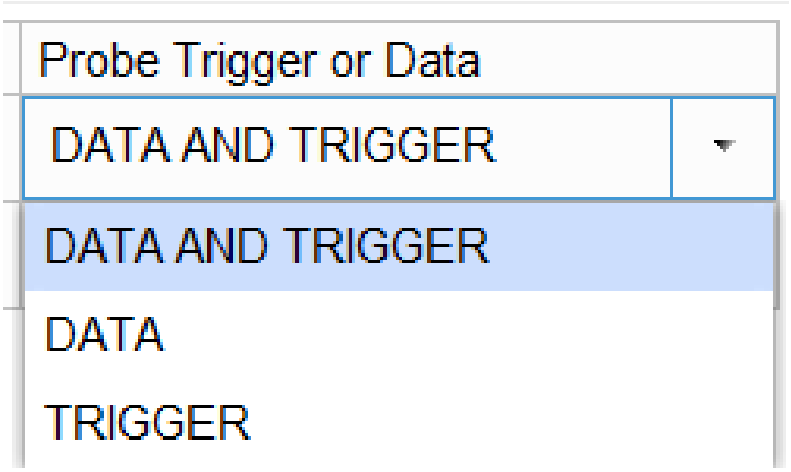


图 1.2-13 Probe Trigger or Data 设置窗口

这里将 Probe0 和 Probe1 均设置为 DATA AND TRIGGER。以上所有设置完后的两个界面如下图所示。

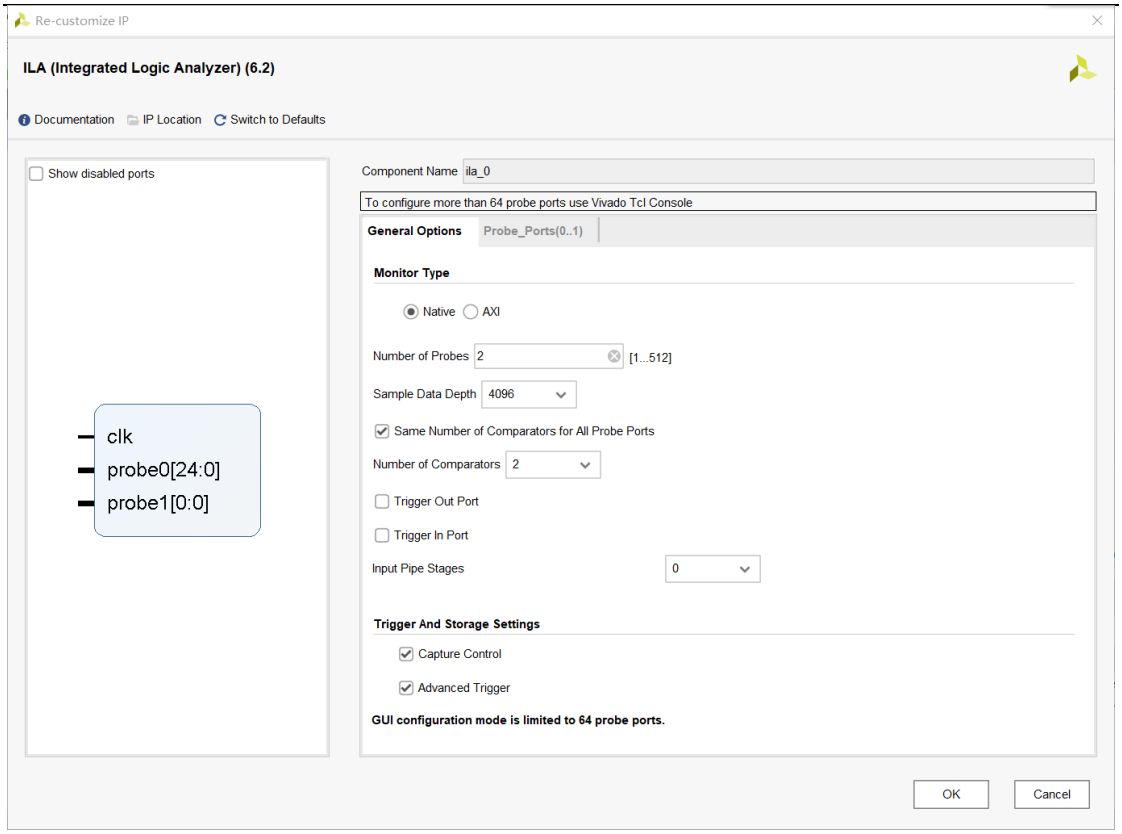


图 1.2-14 IP 设置完后的窗口界面 1

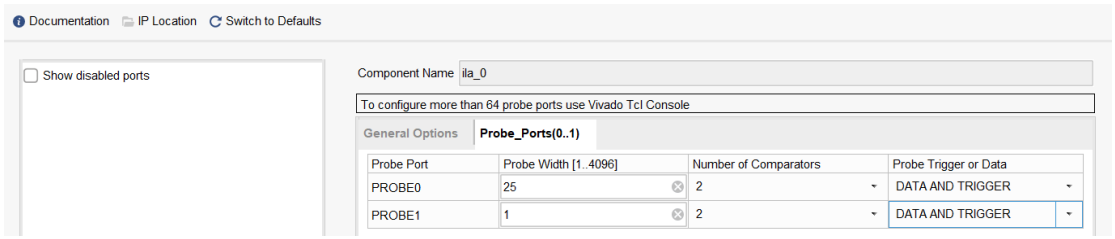


图 1.2-15 IP 设置完后的窗口界面 2

设置完后，点击 OK。

1.2.3 ILA 生成 IP 界面

IP 核配置完成并点击 Ok 后，出现如下图所示弹窗。这里，Number of jobs 和计算机编译时调用的 CPU 核的数量相关，利用较多的资源能够获得较快的编译速度。

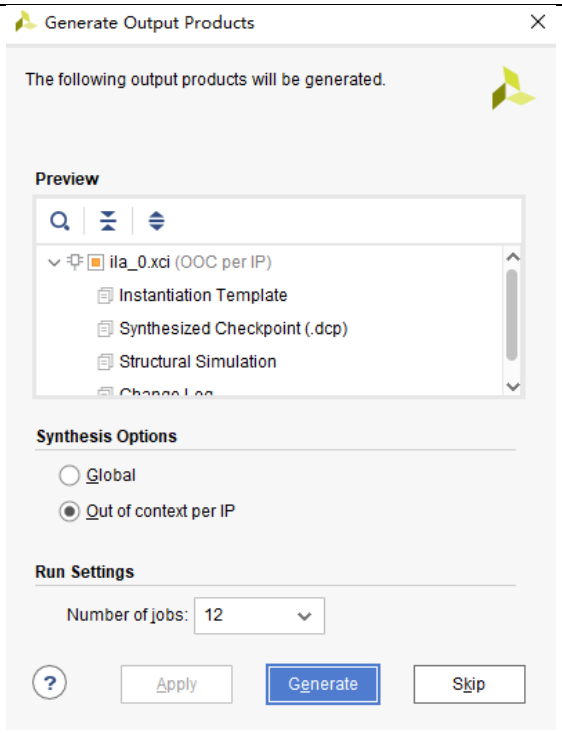


图 1.2-16 生成 IP 窗口

点击 Generate，出现如下图所示生成 IP 过程弹窗。

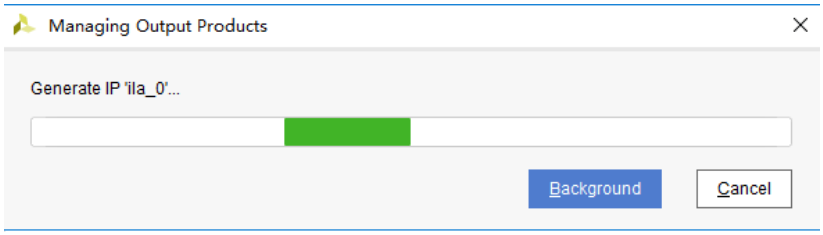


图 1.2-17 生成 IP 过程弹窗

生成完后，出现如下图所示弹窗，点击 OK。

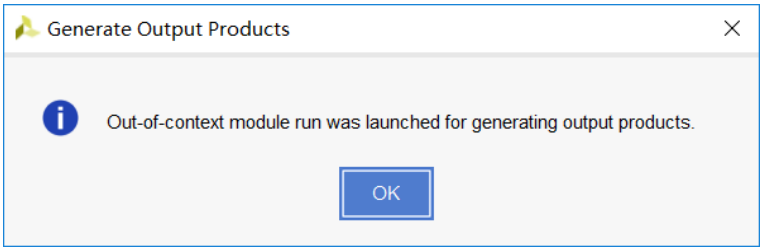


图 1.2-18 IP 生成完成弹窗

到这里 ILA IP 就已经生成好了。可以在如下图所示的工程 Source 窗口点击 IP Sources 可以查看生成的 ILA Core 的一些文件。

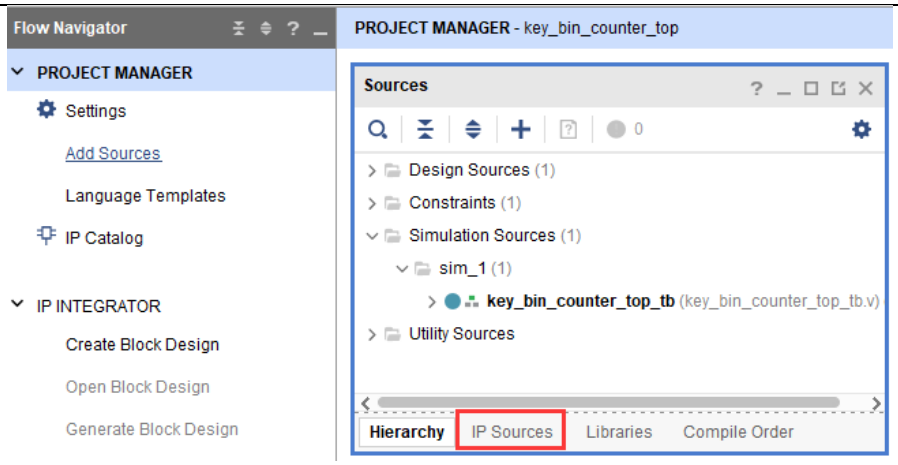


图 1.2-19 查看 IP Core 生成文件入口

如下图所示，通过点开左侧的扩展，可以在 Instantition Template 下找到.veo 的文件，该文件是这个 IP Core 的例化模板，双击 ila.veo 文件，可通过双击查看该模板的代码。例化代码中提供了时钟端口和刚刚配置完成的两组探针端口。同时在注释中提示了生成 IP 核时给出的位宽配置。

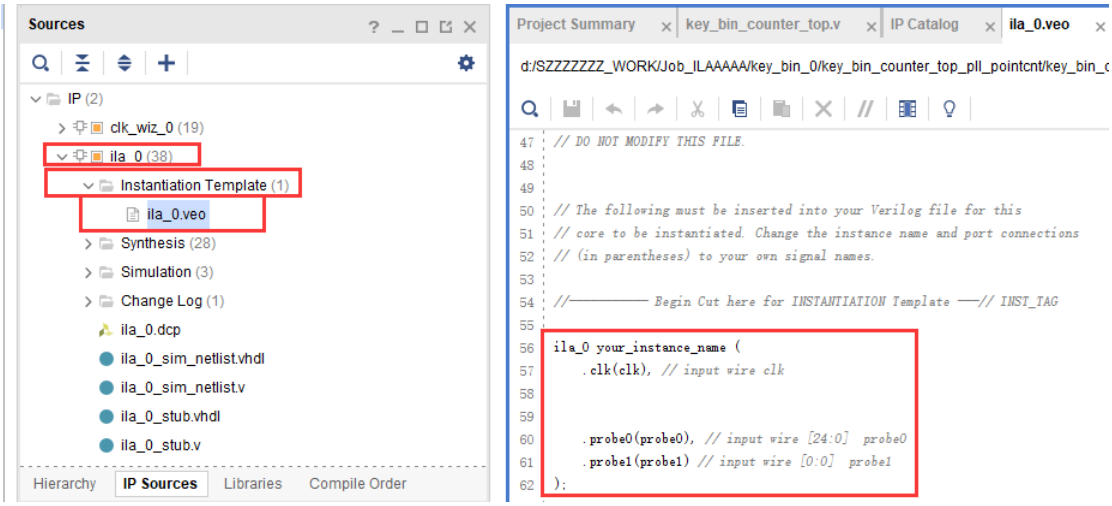


图 1.2-20 IP Source 窗口例化模板位置

该 ILA 模块有 3 个信号，一个时钟信号 clk，两个探测信号，时钟信号就是采样时钟，探测信号就是前面通过 IP 生成界面设置的探测信号，分别是 25bit 和 1bit 信号。将其复制粘贴到我们的工程文件的顶层进行例化，具体代码如下。

```
module key_bin_counter_top(
   clk,
   reset_n,

   key_in,
   led
    );
```

```
    input clk;
    input reset_n;
    input key_in;
    output led;

    wire key_flag;
    wire key_state;
    wire locked;
    wire clk_out1;

    ila_0 ila_0 (
    .clk(clk), // input wire clk


    .probe0(bin_counter.cnt), // input wire [24:0]  probe0
    .probe1(key_state) // input wire [0:0]  probe1
    );

   clk_wiz_0 clk_wiz_0
   (
    // Clock out ports
    .clk_out1(clk_out1),     // output clk_out1
    // Status and control signals
    .reset(~reset_n), // input reset
    .locked(locked),       // output locked
   // Clock in ports
    .clk_in1(clk)    // input clk_in1
    );

    key_filter key_filter(
    .clk(clk_out1),
    .reset_n(reset_n),

    .key_in(key_in),
    .key_flag(key_flag),
    .key_state(key_state)
   );

   bin_counter bin_counter
   (
    .clk(clk_out1),
    .reset_n(~key_state),
    .led(led)
    );
```

```
endmodule
```

从源码中我们可以看到，在 ILA 的使用过程中，Verilog 语法的“.”号也是可以正常的在例化端口的括号内使用和综合的。Verilog 中“.”的含义是：下一级文件中的内容。因此：

```
.probe0(bin_counter.cnt), // input wire [24:0]  probe0
```

这行代码，可以在顶层 ILA 中将探针信号和子模块 bin_counter 中的 cnt 信号关联。从而只需在顶层例化 1 个 ILA，就可以实现将其探针关联到任意层级的相关信号上这一现实需求。

完成以上配置后，就可以继续对工程实施管脚的绑定，从而进行 ILA 操作环节了。鉴于我们还需要讲解另外 3 种创建 ILA 的方法，操作环节的内容，将在完成 4 种创建 ILA 调试工具的讲解内容后，再进行进一步讲解。如果读者只对如何通过创建 ILA IP 核搭建调试环境感兴趣，学完本节内容后读者也可以直接跳到 1.6 节，继续本实验的学习。

1.3 使用 Debug 标记创建 ILA 调试环境

前面我们介绍了使用 IP 核例化的方法，完成 ILA 调试环境搭建的步骤和流程。在本节内容中，我们将介绍使用 Debug 标记的方法，创建 ILA 调试环境。

和创建 ILA IP 核方法进行抓线的步骤相比，直接使用 Debug 标记的方法创建 ILA 调试环境，能够弱化抓线信号的位宽对抓线调试的影响。

这里，我们继续以 key_bin_counter 工程进行举例，对这一刚刚在前面一节内容中使用 IP 创建 ILA 调试环境的工程，采用本节介绍的另外一种方法即 Debug 标记法，进行实例演示和讲解。

在前面的讲解中，我们已经对工程的架构进行了熟悉，这里我们回到工程最初状态，即如图 1.2-1，开始介绍我们的流程。

1.3.1 对需观察信号进行标记

如我们仍然以顶层的 key_state 和 bin_counter 模块中的 cnt 信号作为抓线采样研究对象。我们分别在顶层代码和 bin_counter 模块中对这两个信号进行标记。由于 ILA 需要参考时钟保证其工作，这里，我们以锁相环输出时钟 clk_out1 作为 ILA 的工作时钟。

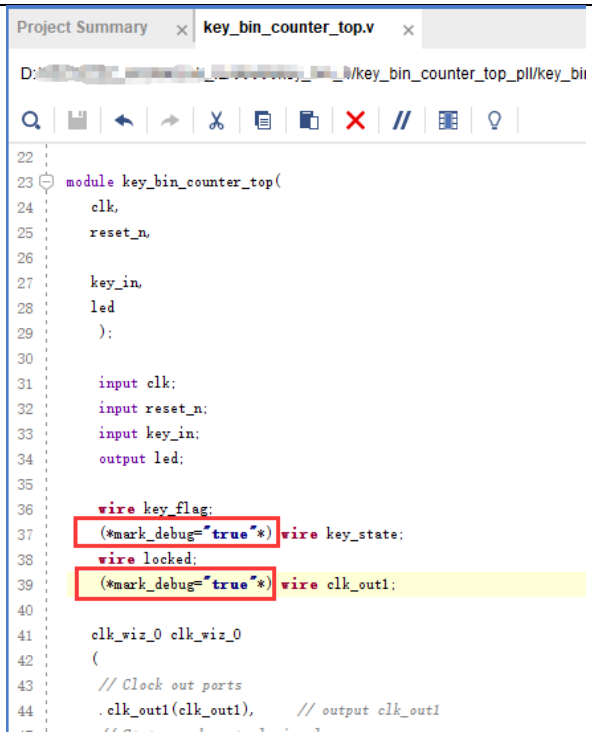


图 1.3-1 顶层模块信号标记

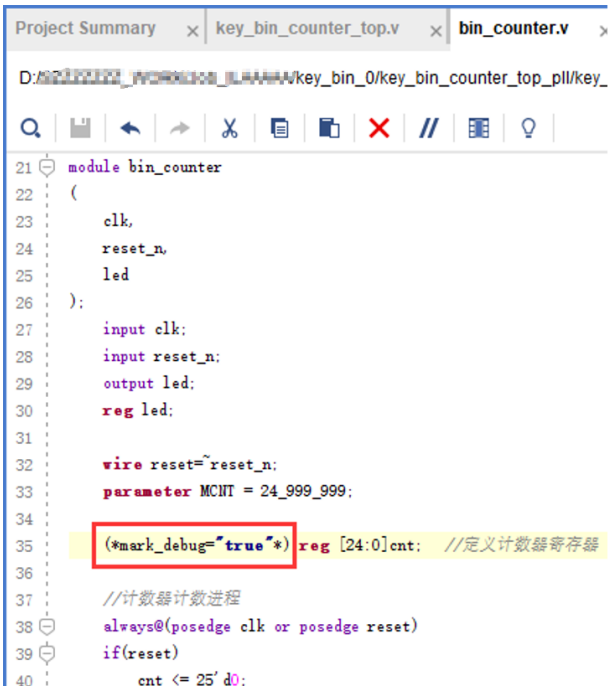


图 1.3-2 bin_counter 模块信号标记

1.3.2 对工程进行保存后重新综合

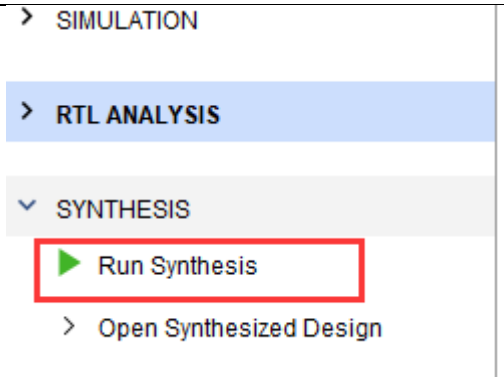


图 1.3-3 对工程进行综合

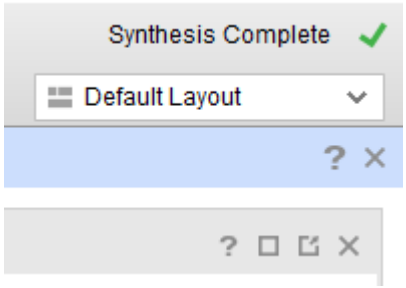


图 1.3-4 综合完成

1.3.3 Debug 信号和探针的关联

点击 Open Synthesized Design，即可进入 Debug 配置界面。

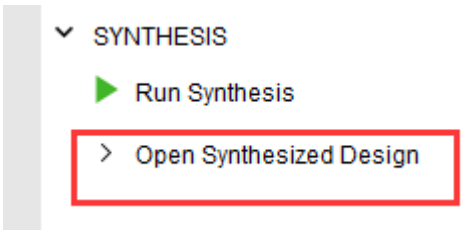


图 1.3-5 点击 Open Synthesized Design

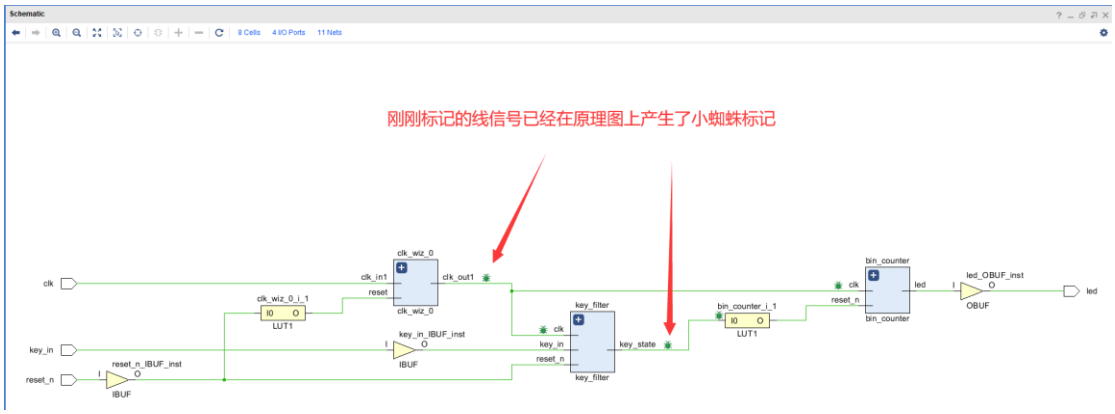


图 1.3-6 主窗口页面显示

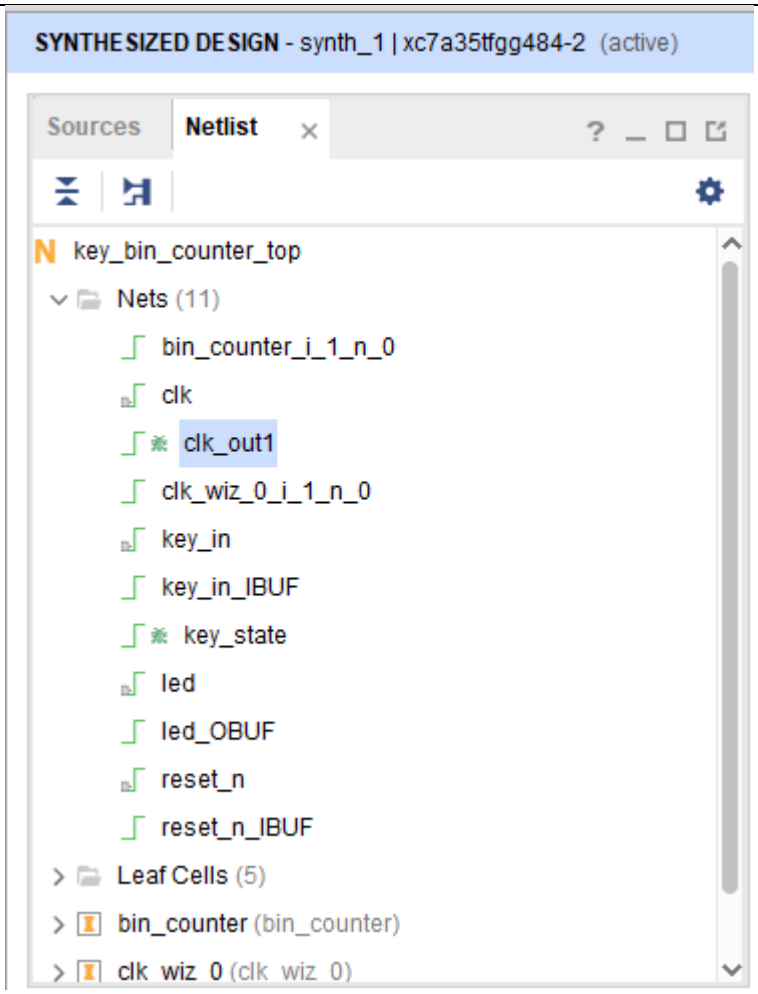


图 1.3-7 网表管理窗口的标记信号前方也有小蜘蛛图标

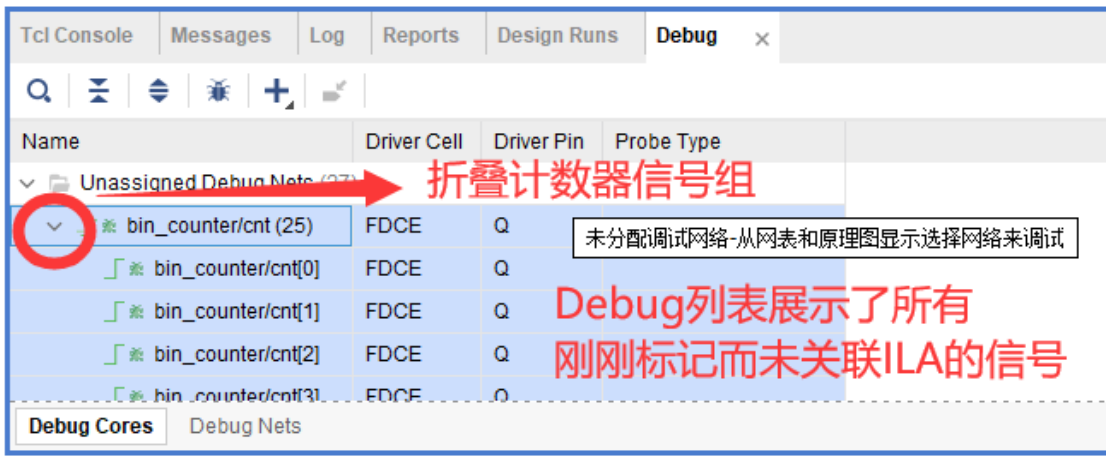


图 1.3-8 折叠信号组

右键点击还未关联的标记信号，在菜单中选择 Create Debug Core。

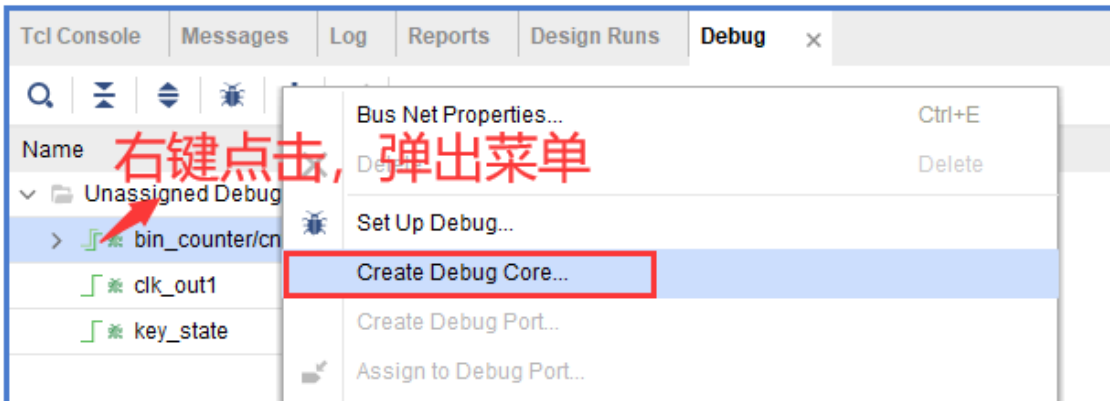


图 1.3-9 创建 ILA IP 核

新的窗口可以设置 ILA 的名称，选择 ILA 的类型，同时也可以进行采样深度设置等操作。和我们在上一节讲解的内容进行相同的设置，我们也将采样深度设置为 4096。

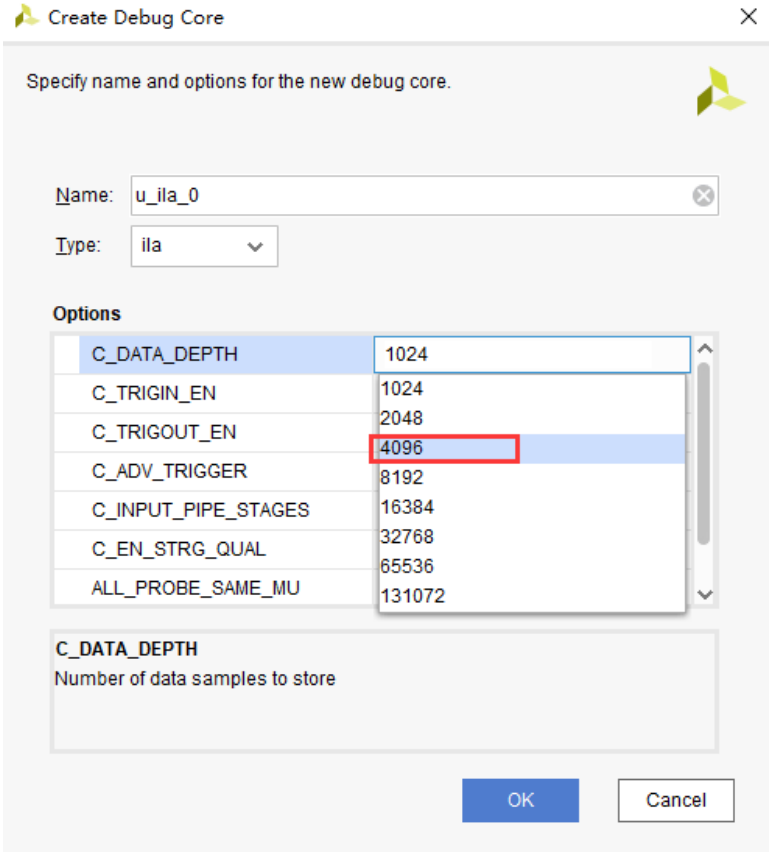


图 1.3-10 选择采样深度

完成 ILA 的设置后，呈现如下界面：

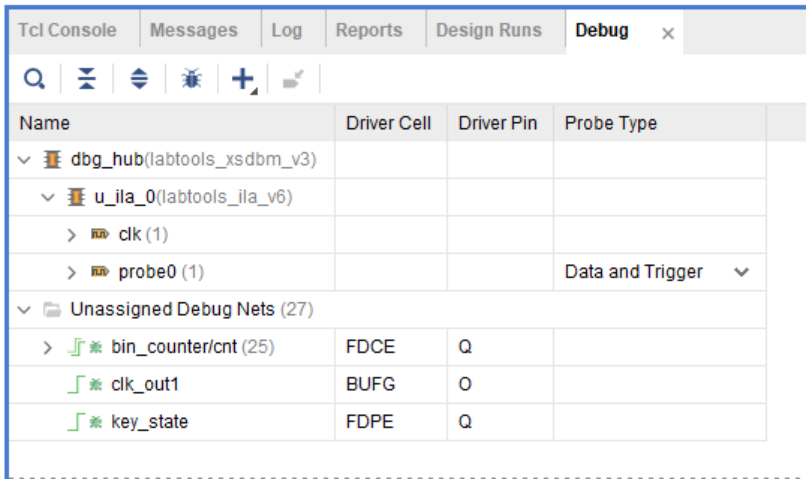


图 1.3-11 完成 ILA 核的设置

我们看到，我们需要对包括时钟信号在内的 3 个信号进行关联绑定，而刚刚创建的 ILA 工具，包括时钟绑定信号在内只有 2 组可以进行绑定的探针。这时，需要结合采样信号的实际组数，将不足的探针组数补齐。

右键点击 probe0，弹出菜单后点击 Create Debug Port。

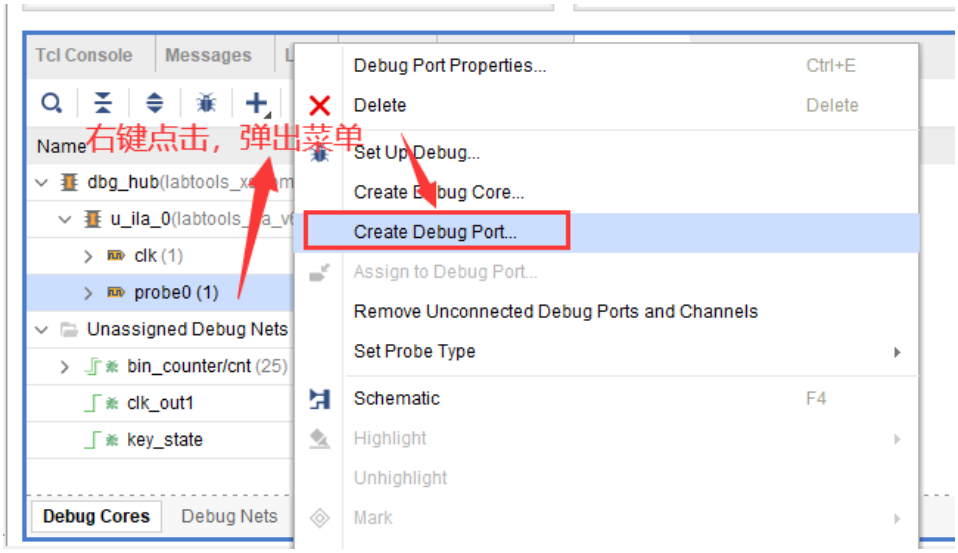


图 1.3-12 点击 Create Debug Port

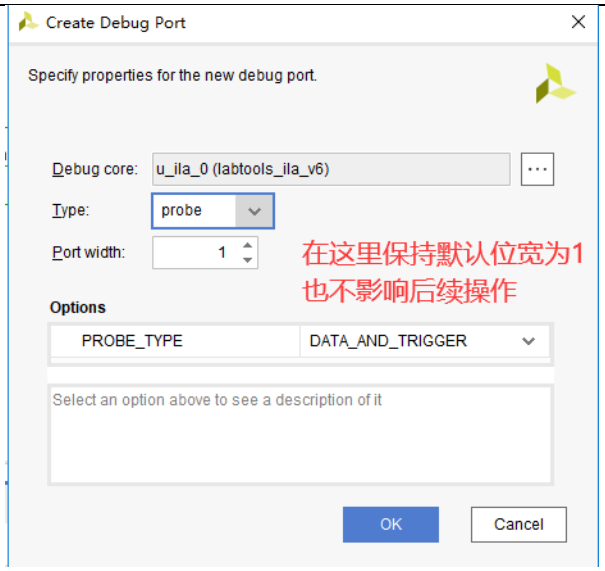


图 1.3-13 填写 ILA 属性

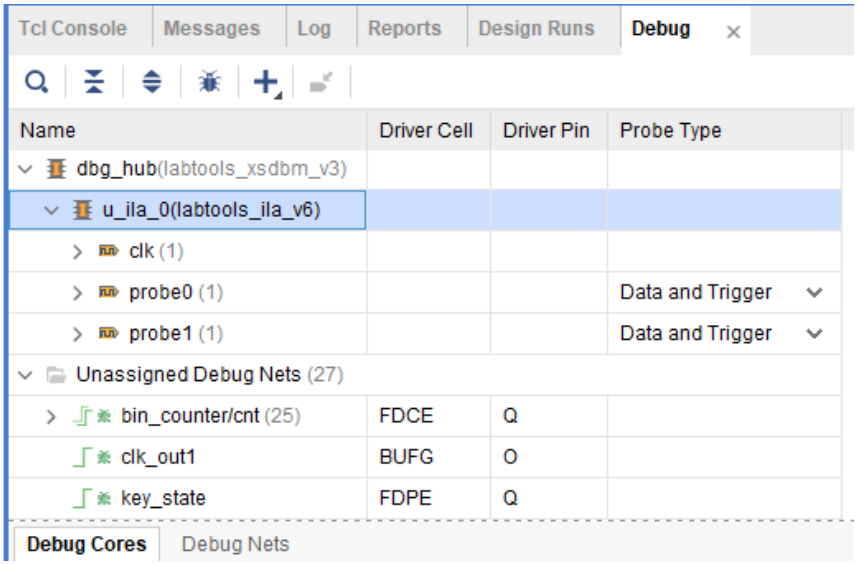


图 1.3-14 生成 ILA 后页面状态

这样，3 组未绑定的信号端口，就有了 3 组 probe 探针。虽然此时的探针位宽都默认为 1，但是并不影响我们绑定探测信号后的结果。接下来，即可开始完成探针和探测信号的绑定。

绑定前，先观察点击时钟信号的下拉选项，发现时钟信号的绑定标号为灰色圆圈实心点。绑定信号名称为默认未绑定的 Ch0。

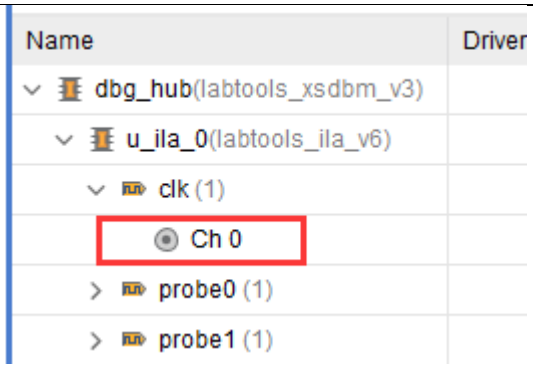


图 1.3-15 绑定前信号标号

先以时钟信号为例，右键点击 clk_out1,在弹出的窗口菜单中选择 Assign to Debug Port。

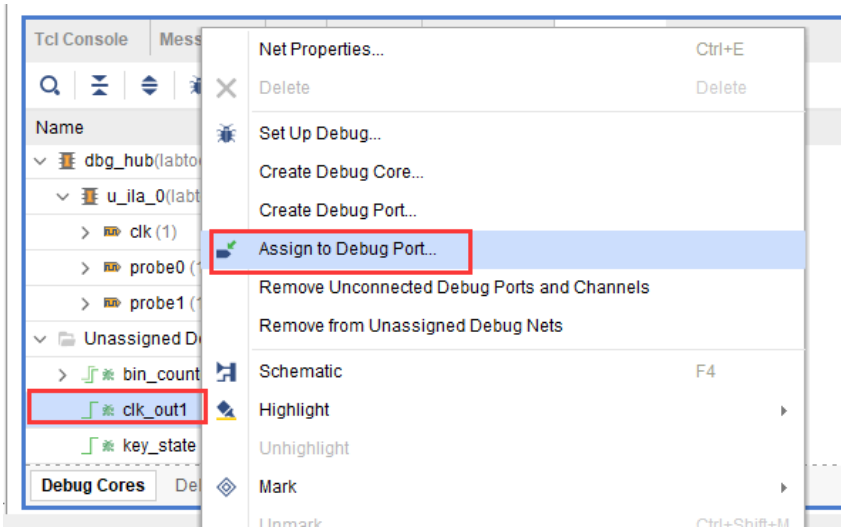


图 1.3-16 将 clk_out1 信号和调试端口进行关联

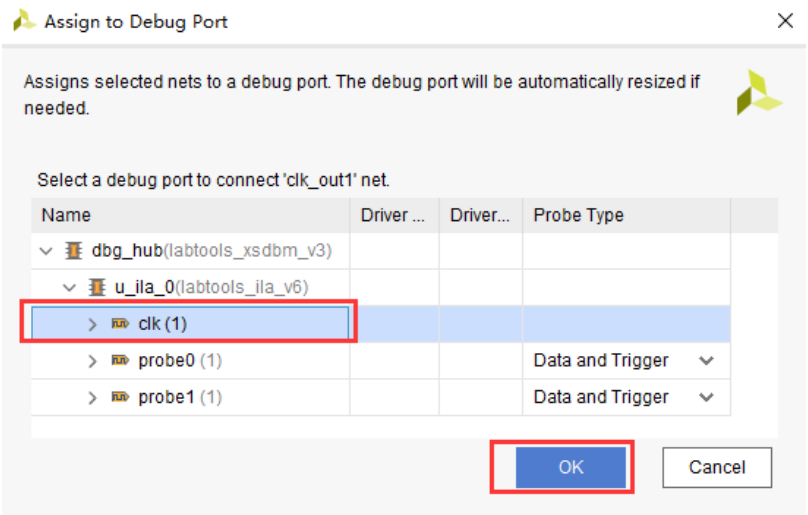


图 1.3-17 选定绑定信号 clk

再来查看 clk 的信号下拉菜单的内容。可见，其灰色实心圆点的标号已经变

成了黄色实心圆点的标号。而与 clk 关联的信号名称 clk_out1，已经成功标注到该标号的后方，说明时钟信号 clk_out1 和 ILA 工作时钟已经关联成功。

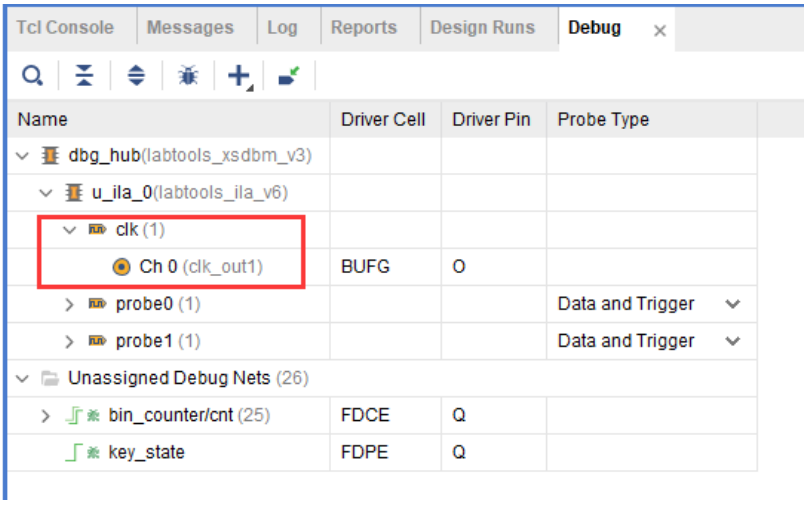


图 1.3-18 将时钟信号 clk_out1 和 ila 信号关联后效果

使用同样的方法，完成另外两组观测信号的关联和绑定，绑定后效果如下。

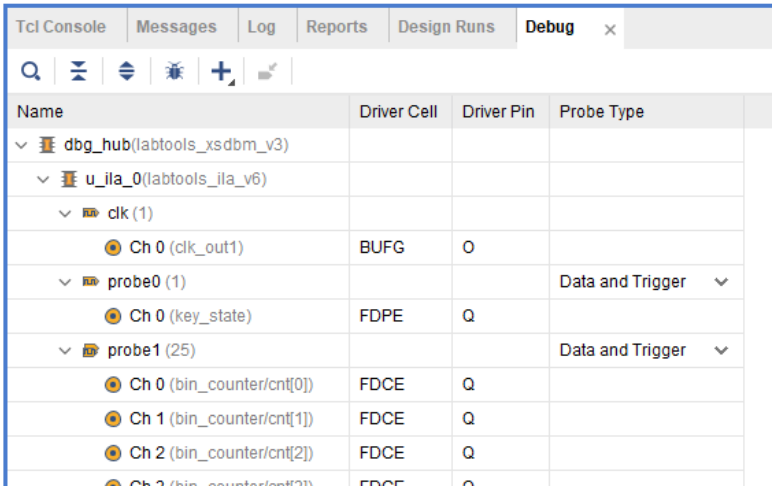


图 1.3-19 完成绑定后效果图

可以看到，我们需要观测的 cnt 信号位宽虽然为 25 位，而绑定探针在绑定前位宽为 1 位。但完成绑定后，探针位宽会自动识别为 25 位，这样，和 IP 核创建 ILA 的方法比起来，可以淡化位宽的定义操作，减小调试时发生人为错误的概率。

有时候，由于操作不慎，信号探针在绑定时放出了多余的位宽，例如：

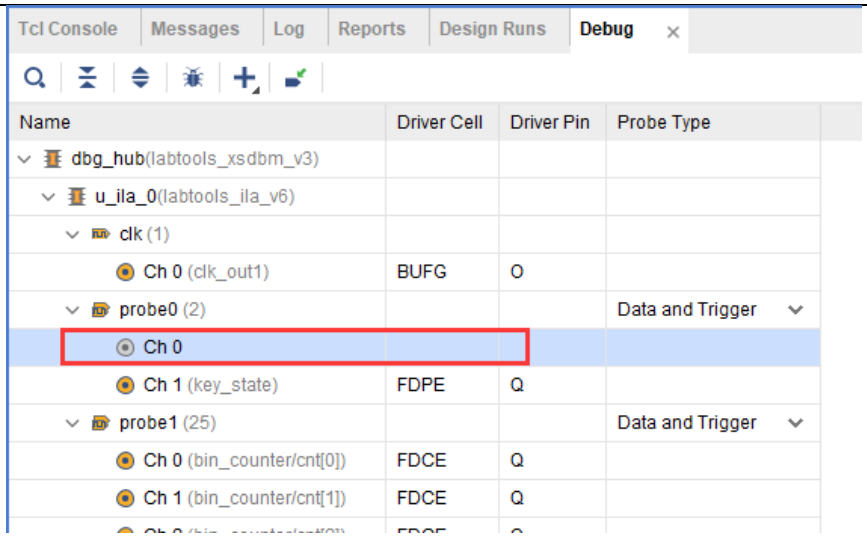


图 1.3-20 探针信号放出了多余的位宽

这时也不用担心，只需从右键菜单将这一无效位宽进行移除即可。

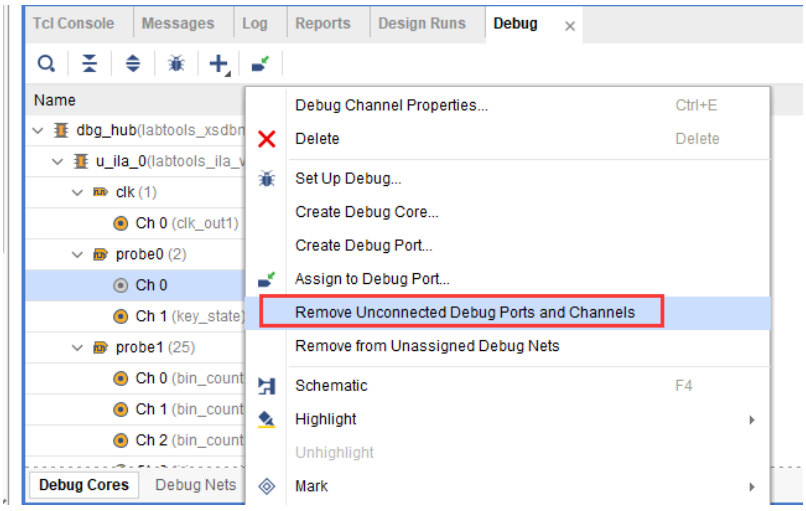


图 1.3-21 移除无效的探针端口

1.3.4 完成关联后的原理图

到这里，我们再来仔细观察一下原理图，发现原理图右上角多出了 ila 和 dbg 模块。同时，我们期望进行观察而关联到探针的信号组，在原理图中已经用小蜘蛛进行了标识。

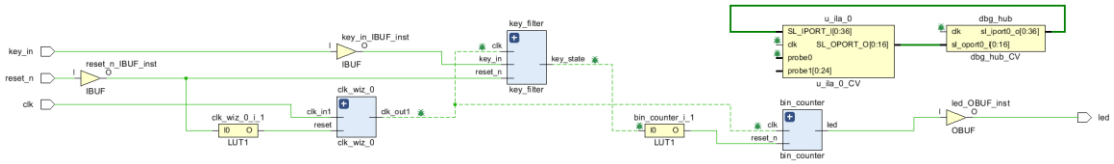


图 1.3-22 经过 ILA 关联后的原理图

经过以上操作，使用 Debug 标记创建 ILA 调试环境的工作已经准备完成了，接下来对工程进行全编译即可进一步进行板级验证。

值得注意的是，在使用 mark_debug 方法抓线完成调试工作后，要记得删除源码中的 mark_debug 标记。否则，不但代码会留下杂乱的调试痕迹，而且被标记的信号会保持占用工程的逻辑资源，导致工程的总逻辑资源使用量增大。

1.4 使用路径标记和 Set up debug 菜单创建 ILA 调试环境

我们在前面介绍了使用 mark_debug 的方法，标记抓线信号的操作流程。在实际使用中，我们还可以通过在综合后的原理图上直接进行调试标记，而简便的锁定需要观察的信号。这样，我们就不用在代码中进行标记，也不需要承担在调试完成后忘记在代码中删除 mark_debug 标记的风险了。

还是回到工程原本的初始状态，首先对工程进行分析和综合。

图 1.4-1 对原始工程进行分析和综合

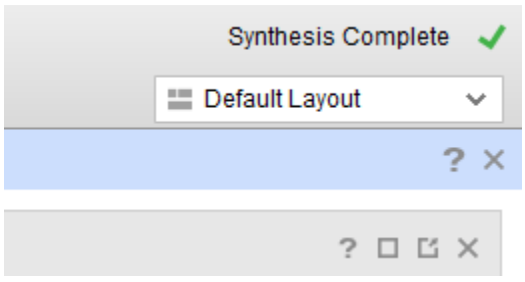


图 1.4-2 分析和综合完成

完成分析和综合后，点击 Open Synthesized Design

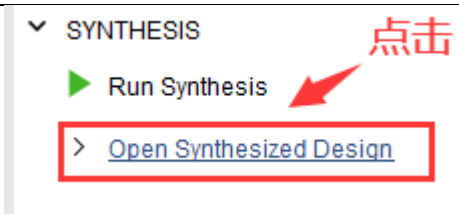


图 1.4-3 点击 Open Synthesized Design

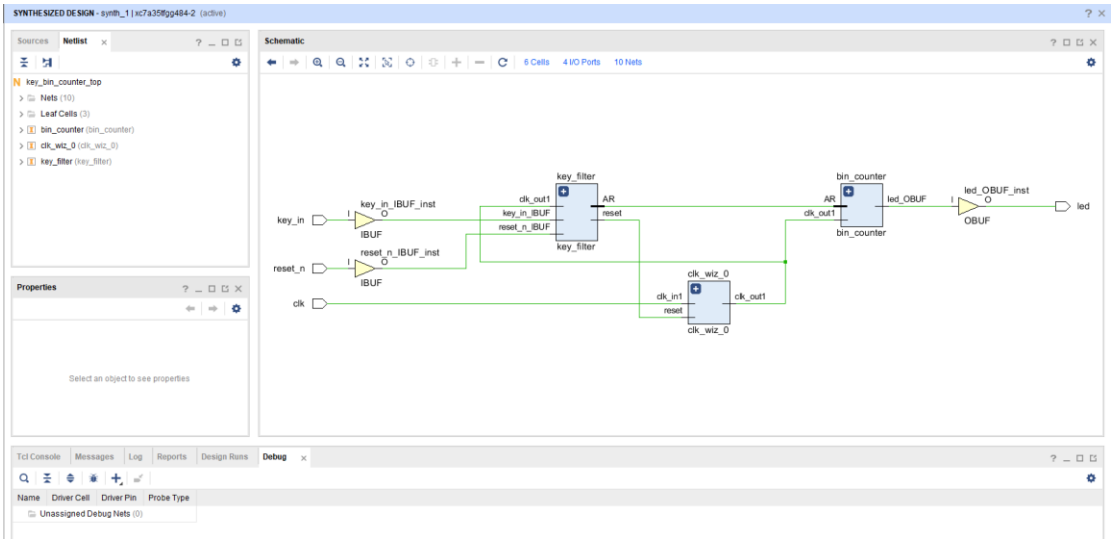


图 1.4-4 综合完成后的主窗口页面

可以看到，综合完成后，如果选择 Open Synthesized Design，则默认在主窗口页面会有 Netlist、Schematic、Properties 和 Debug 这样几个小窗口分布。在这里，选择我们需要观察的 key_state 信号和 cnt 信号可以有非常灵活的方法。

这里，我们直接从 Schematic 原理图中，寻找并标记 key_state 信号。仔细观察原理图，我们并未发现 key_state 字样。由于 VIVADO 会按自己的规则给编译后的信号命名，所以我们推测，key_state 信号就在我们当前的原理图中，只是命名方式发生了变化。

结合我们对工程的理解，加上我们在寻找信号时的注释提示，如果把鼠标放在 AR 连线上，则显示该网络为 key_state。

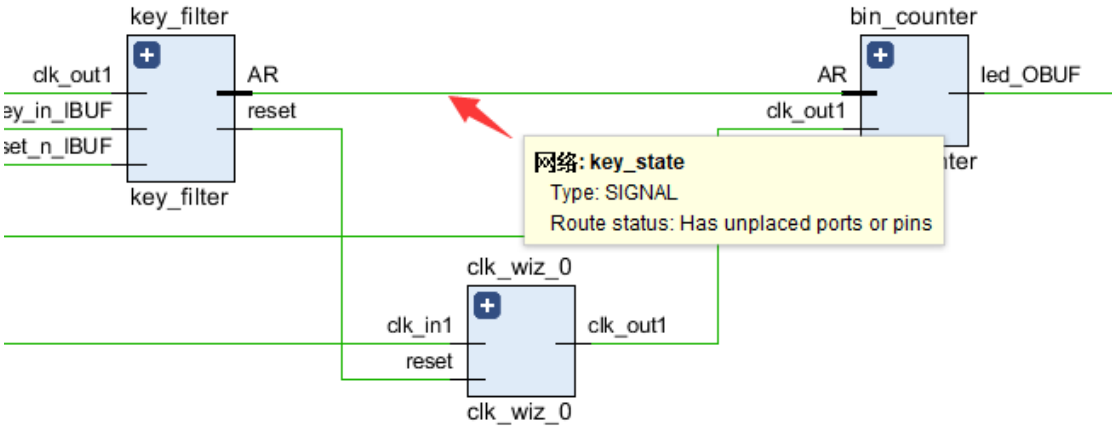


图 1.4-5 显示 AR 连线为 key_state

右键点击 AR 网络，随后点击 Mark Debug 菜单。

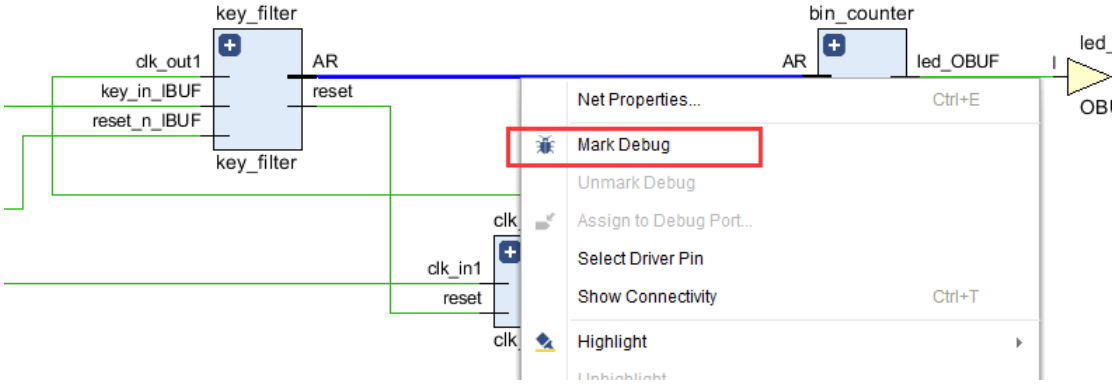


图 1.4-6 标记 AR 网络

这时，该网络即被标记为待观测信号。

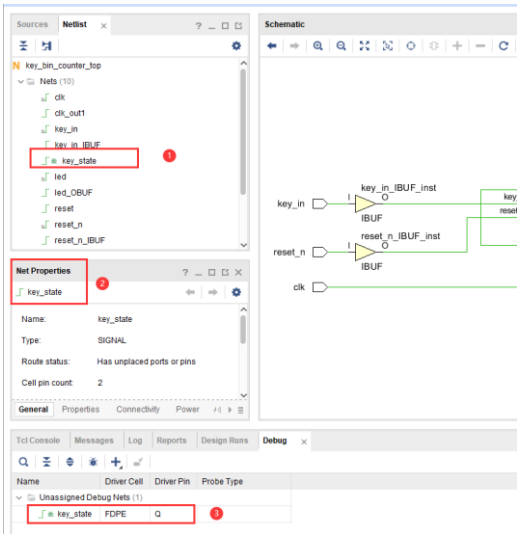


图 1.4-7 3 个小窗口的显示内容发生变化

这样，我们在 Schematic 原理图中标记信号的演示，就完成了。接下来我们演示在 Netlist 列表中，标记 cnt 的方法。

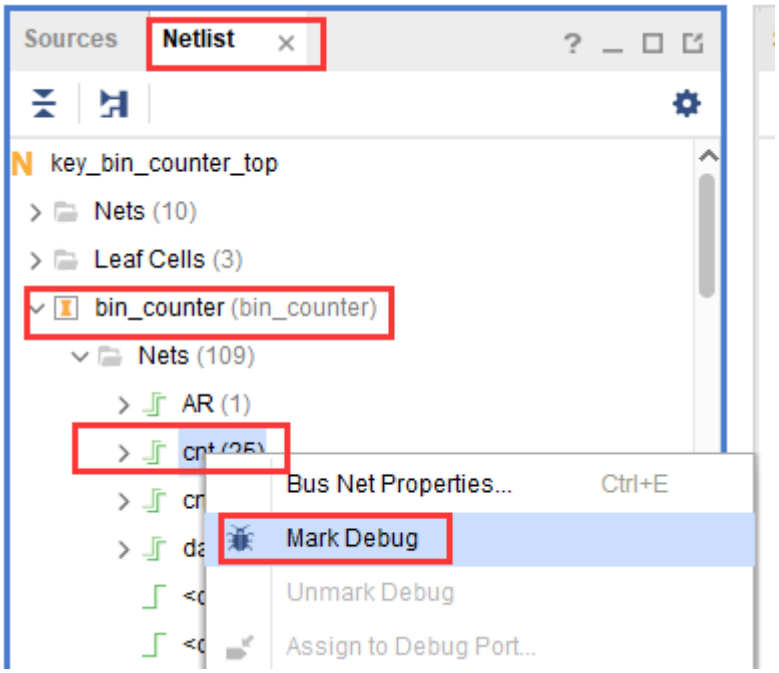


图 1.4-8 在 netlist 中标识 cnt 信号

标记完成后，3 个小窗口的内容再次发生变化，未分配调试网络的列表中，多出了 cnt 信号组。

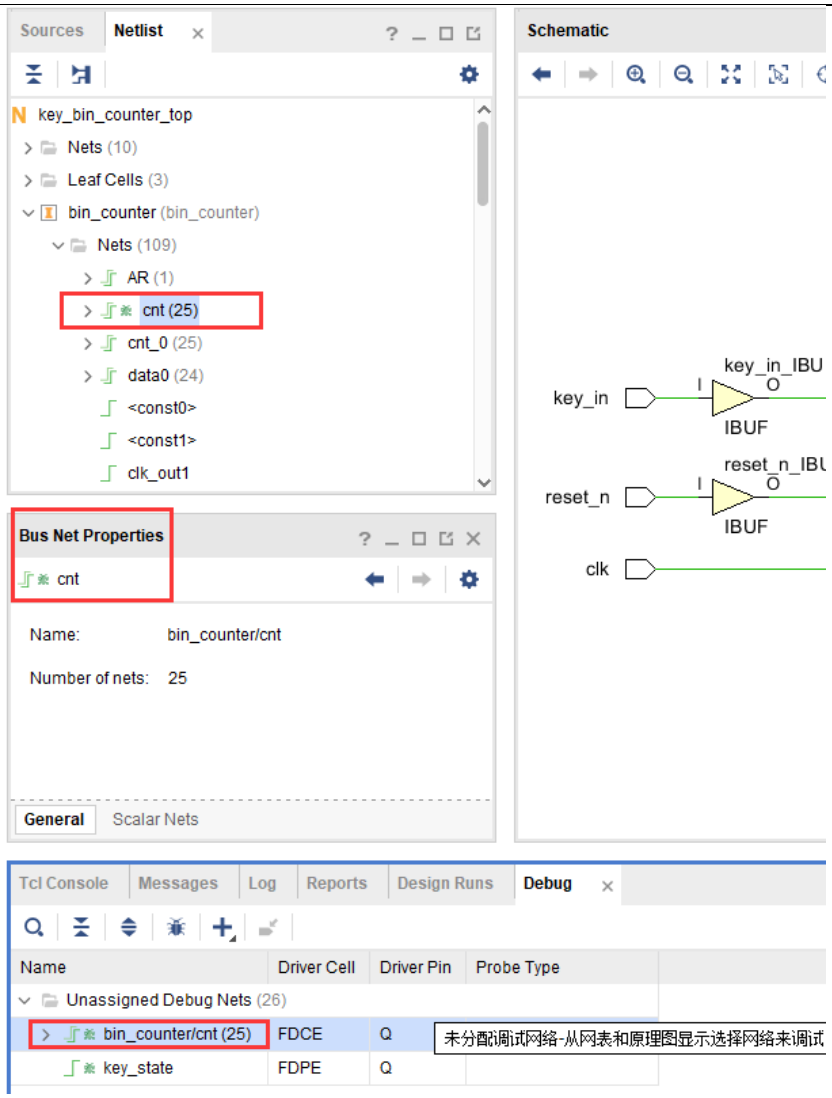


图 1.4-9 3 个小窗口的显示内容再次发生变化

使用同样的方法，我们对 clk_out1 这一时钟信号进行标记。

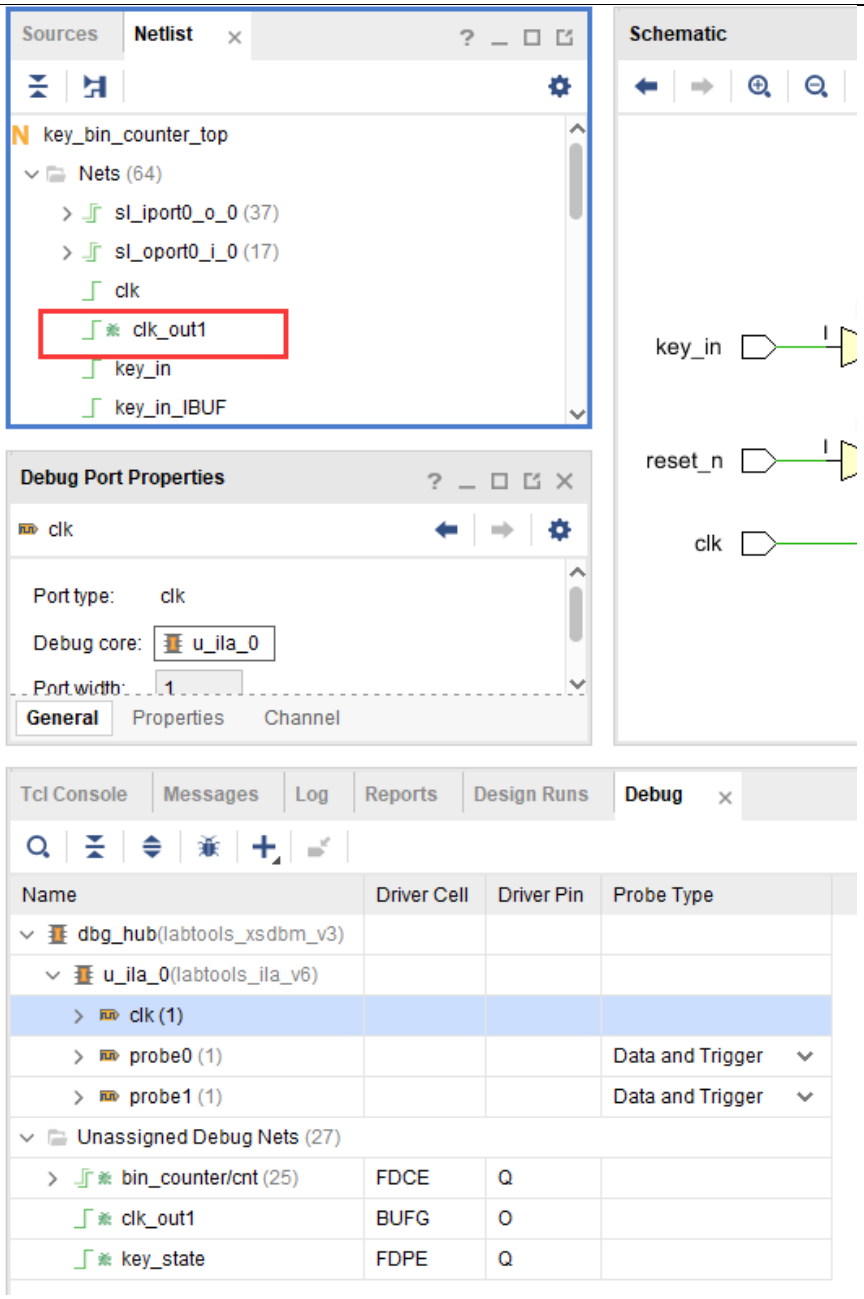


图 1.4-10 将 clk_out1 和时钟 clk 进行关联

参考前面介绍的图 1.3-9 生成 ILA dbg_hub 的方法，我们也同样可以回到前面的流程，继续完成 ILA 调试工作。

在这里，我们还可以通过 Set Up Debug 的方法，利用新建建调试的引导流程，进入 ILA 设置。

鼠标右击 Unassigned Debug Nets，然后点击 Set Up Debug。

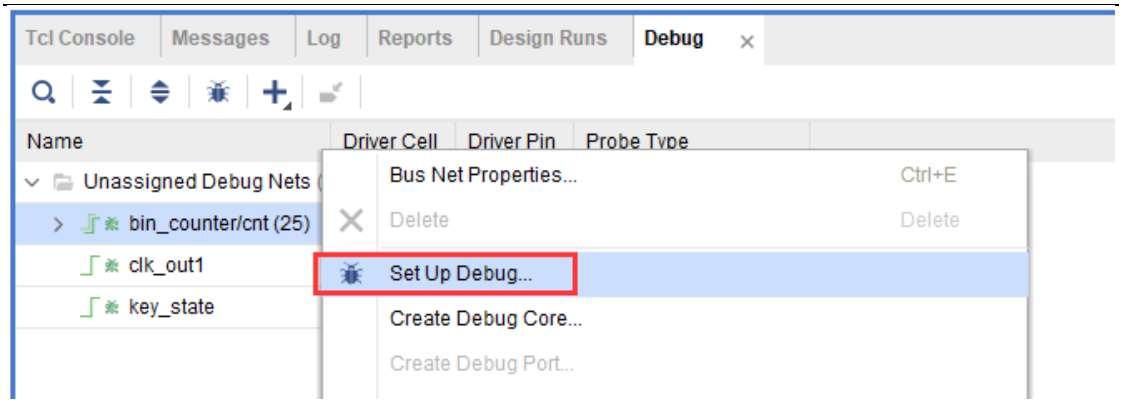


图 1.4-11 新建 Debug

在第一步，直接点击 Next，会进入第二步。

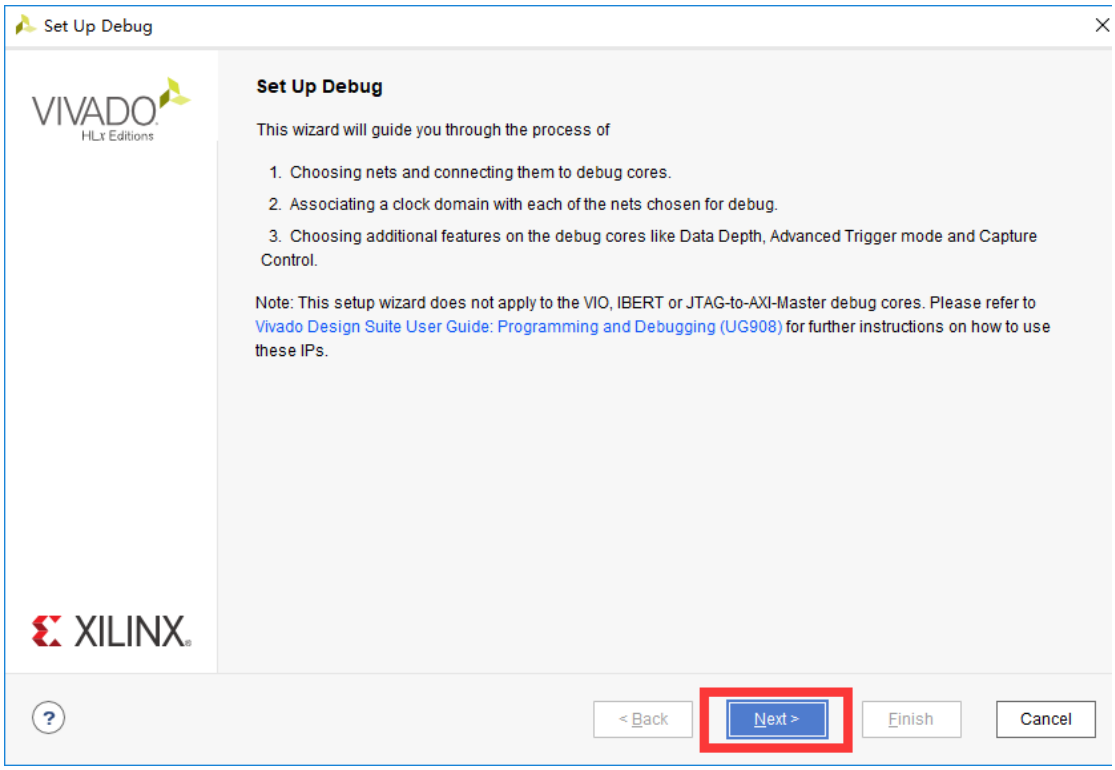


图 1.4-12 Set Up Debug 第一步

Set Up Debug 第二步会列出所有的未关联的调试信号，同时给出这些信号默认的时钟域。

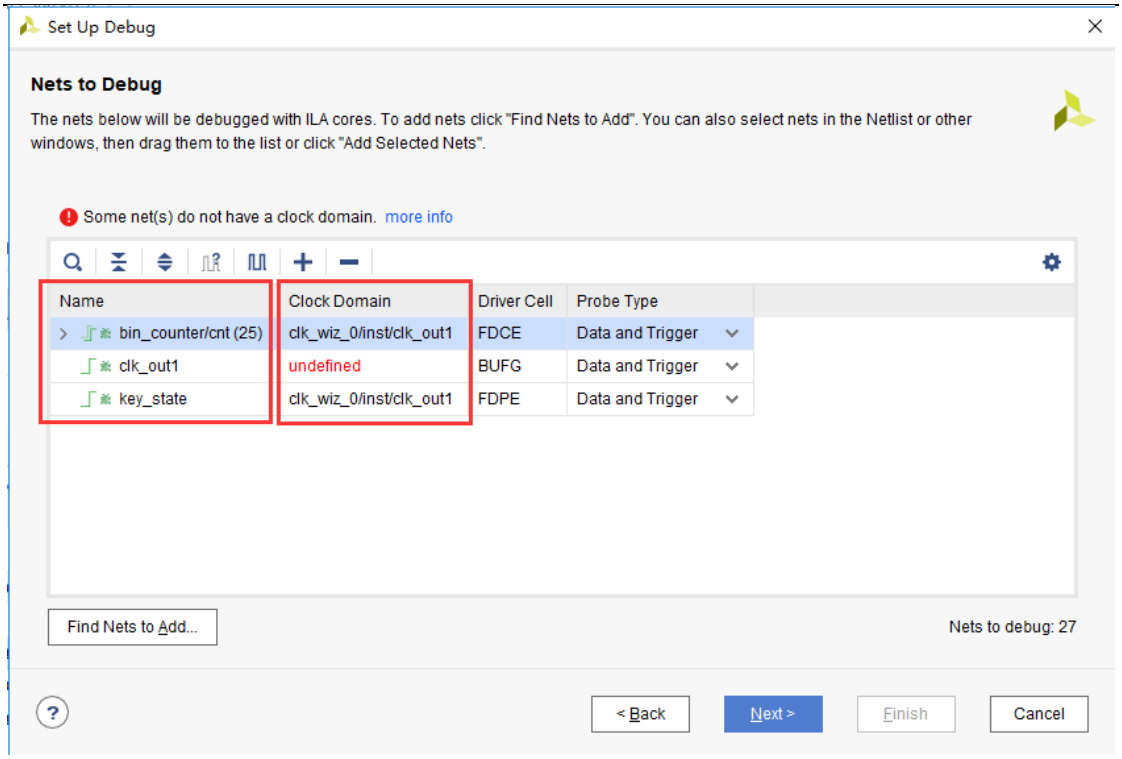


图 1.4-13 在列表中可以选择或修改未关联信号的时钟域

如果默认给出的时钟域并不是你期望的时钟域，则可以通过右键菜单重新对时钟域进行选择。在这里，针对每组待观察信号，时钟域甚至可以有不同的选择。这也给工程调试带来更大的自由度。

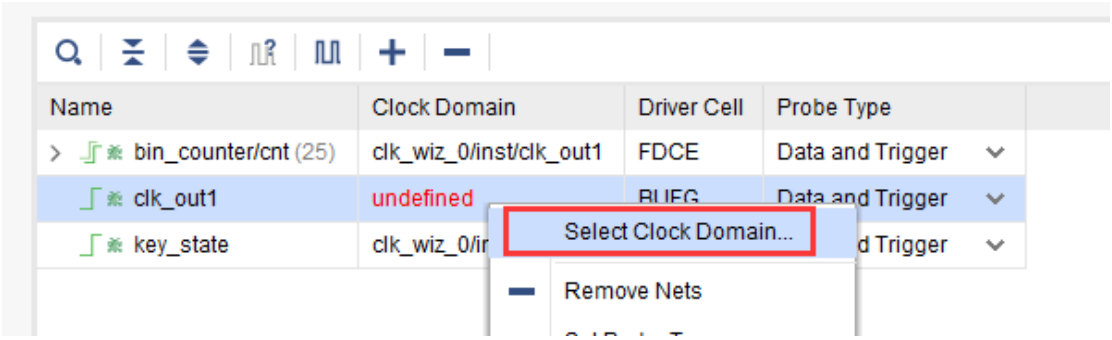


图 1.4-14 选择时钟域

在选择时，一般不太大的工程，可以在图 1.4-15 中的下拉菜单中，选择 ALL_CLOCK 以展示全部时钟。

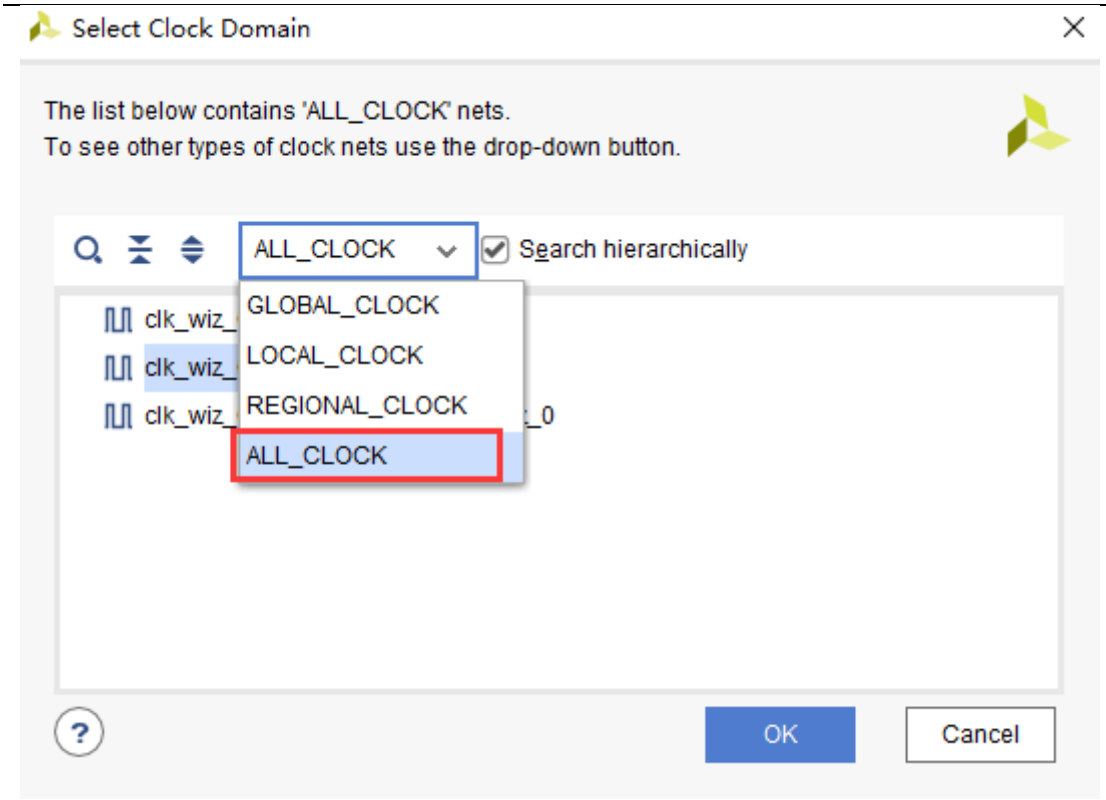


图 1.4-15 时钟域的下拉菜单

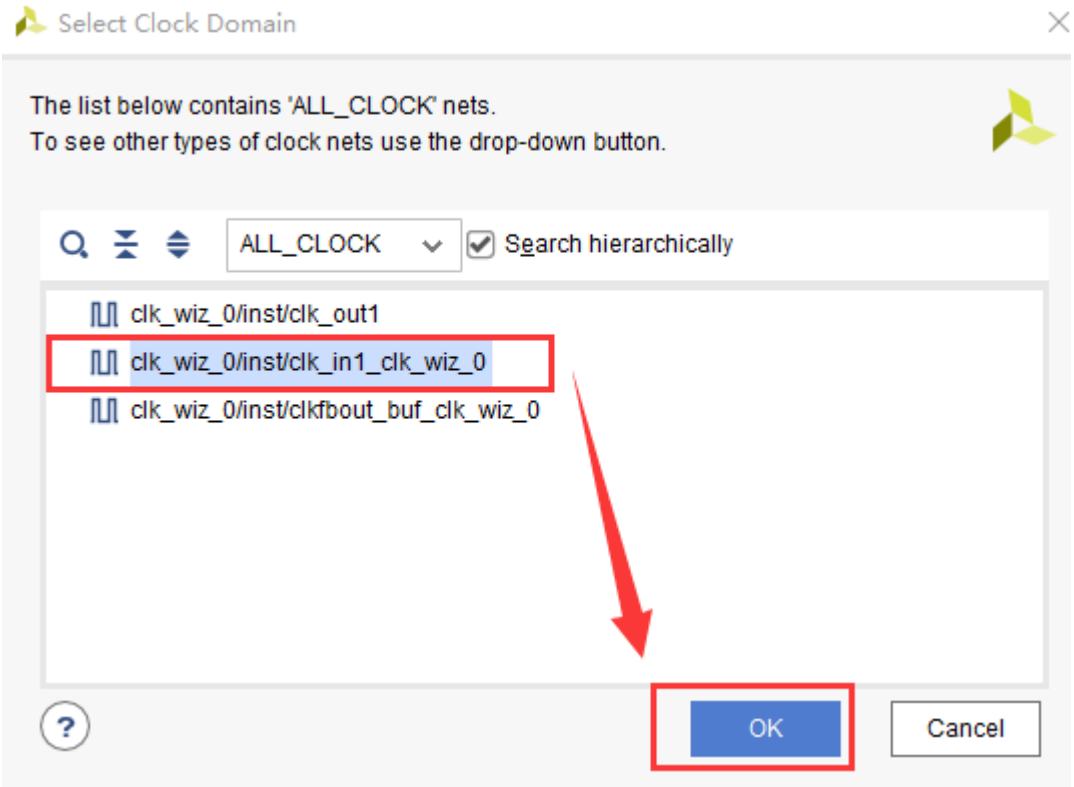


图 1.4-16 在列表中选择新的时钟信号

由于 clk_out1 本来就是一个时钟信号，所以我们也可以将其从列表中删除。

这样并不会影响后续 ILA 时钟的生成。

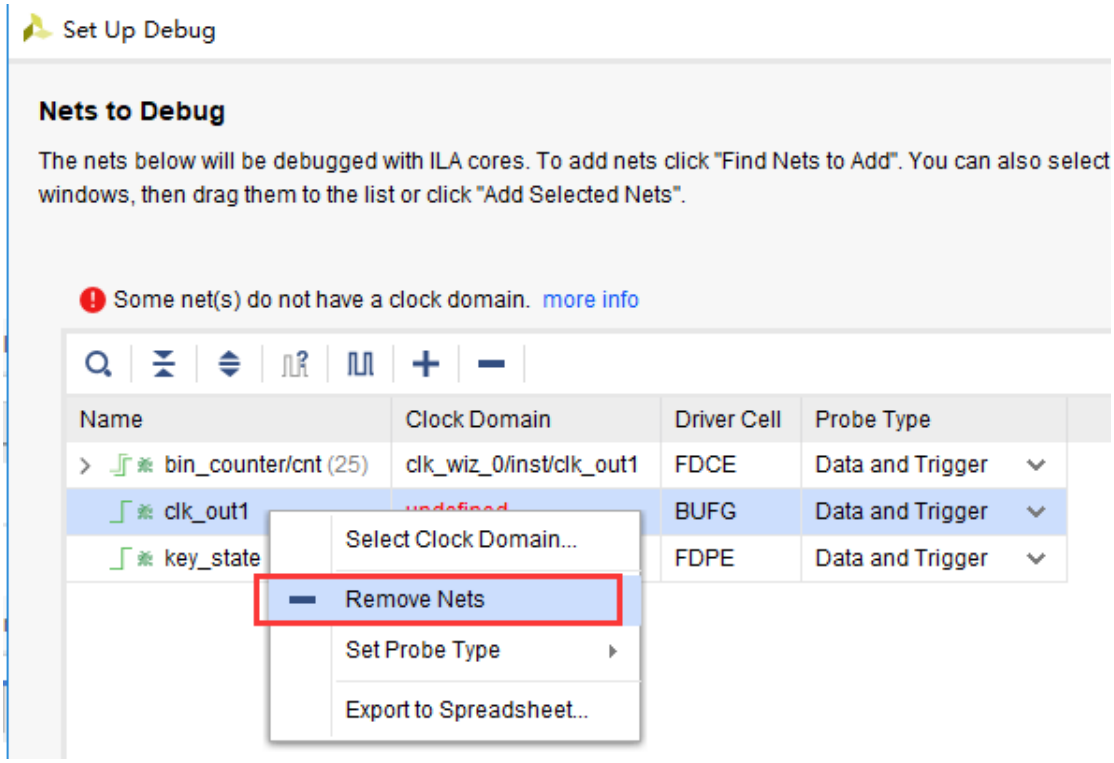


图 1.4-17 删除未关联信号

进入下一步，选择采样深度，这里，我们和前面保持一致，仍然选择 4096 个采样点。

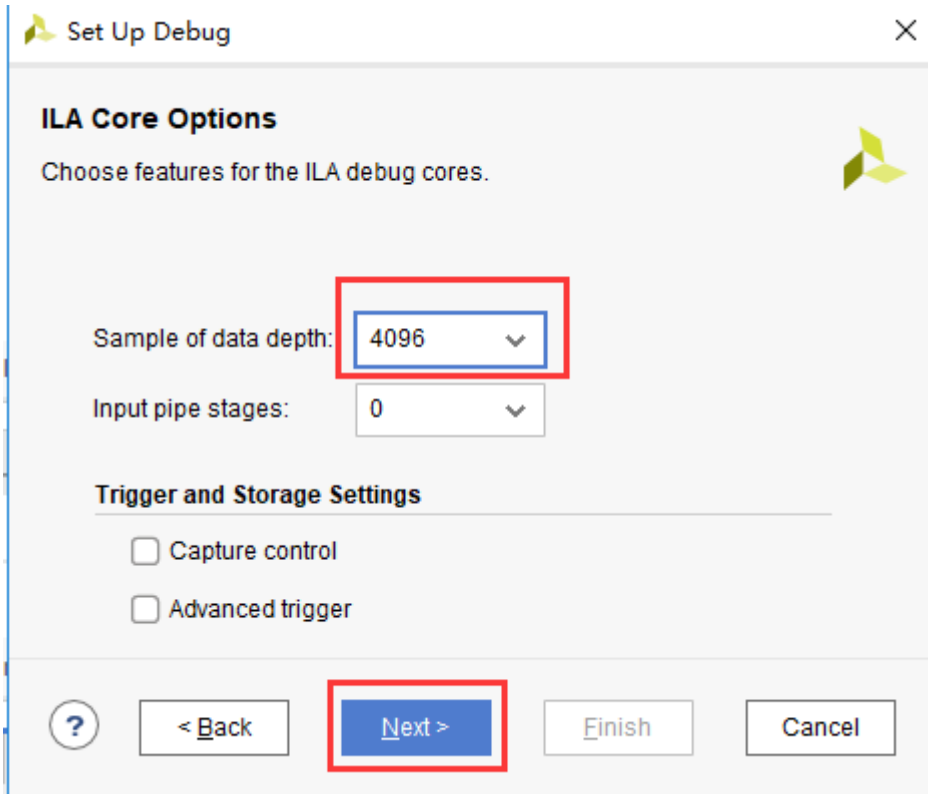


图 1.4-18 选择采样深度

经过一段时间的等待，Debug cores 就重新生成了。

图 1.4-19 生成 Debug cores

到这里，调试环境的搭建，就又顺利完成了，可以看到，和前面图 1.3-19 的状态，又是一致的了。

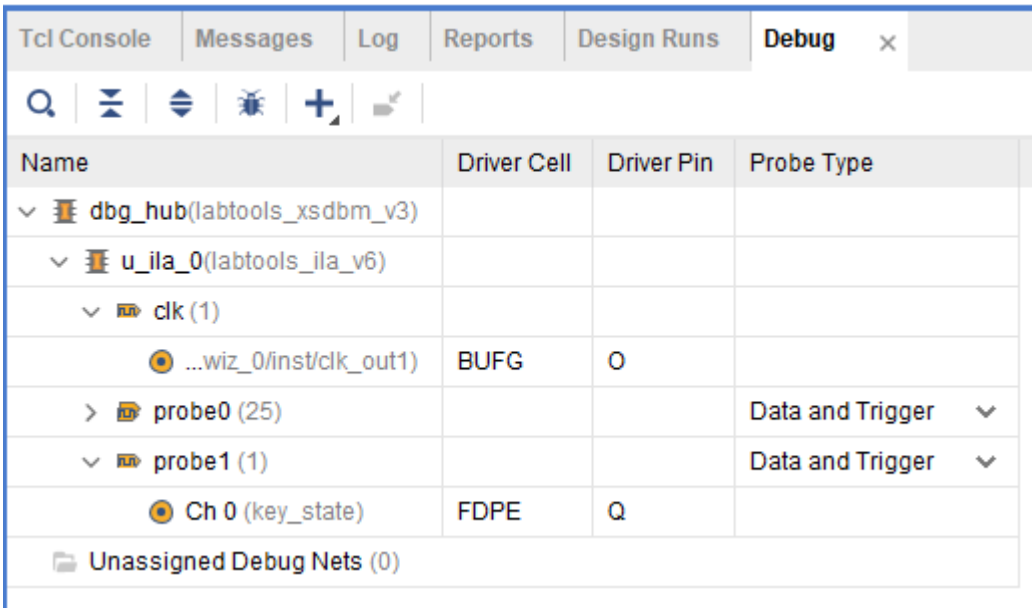


图 1.4-20 使用本节方法完成调试信号关联后效果

到这里，ILA 调试环境的搭建工作，我们就讲了 3 种切实可行的方法。值得提出的是，有时候，主时钟信号（如本工程中的 clk 信号）无法被 Debug 工具准确识别，这时候，可以通过我们最后介绍的 Set up debug 方法，在时钟域列表中查找其在工程中其他管脚上被重新定义的名称。

1.5 使用 IP 核在 Block Design 界面创建 ILA 调试环境

前面我们介绍了逻辑代码设计中 ILA 调试工具的使用方法。而有时候，一个工程的设计并不仅仅只包含纯逻辑设计。典型如 ZYNQ 系统中，往往会引入 CPU

代码设计。在这种条件下，是否也能发挥 ILA 调试工具的用武之地呢？答案也是肯定的。

由于使用 VIVADO 进行 PS-PL 联合设计的实战工程中，PS 部分（即 CPU）设计部分是以一个独立而特殊的封装模块而体现在设计中的，而我们进行的 PL 设计，也可以很容易的导入到图形化界面设计工具中，因此，我们把在前面介绍过的已经熟悉的举例工程，以 Block Design 封装的形式作为演示内容，在这里呈现给各位读者。

由于部分读者有可能还未接触过 Block Design 设计方法，我们也是先搭建好了 Block Design 设计平台。对于这部分读者，可以直接打开我们提供的文件夹：key_bin_counter_top_pll_Block_start 即可。至于如何搭建 Block Design 工程，不在我们本次课程的讨论范围之中，有兴趣的同学可以通过学习我们 ZYNQ 开发板的相关课程进行了解和掌握。

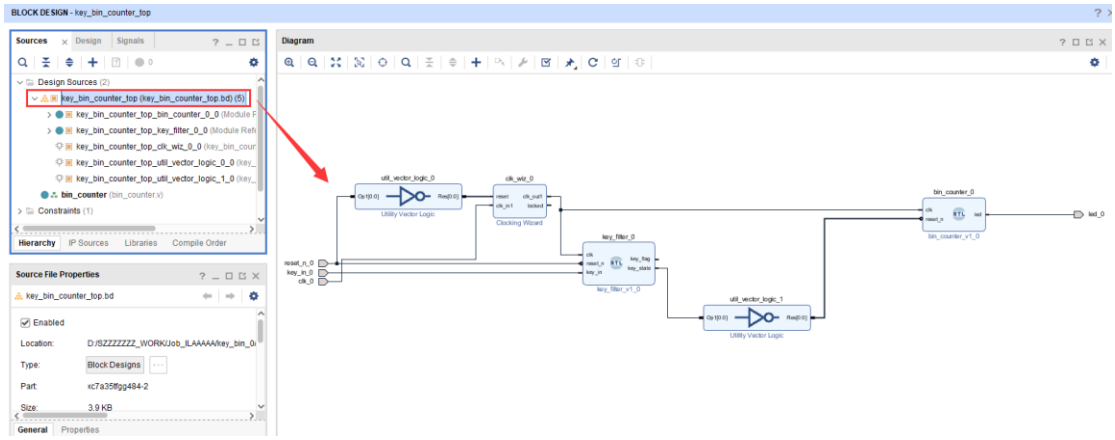


图 1.5-1 工程原始架构

对于我们需要标记的信号，我们既可以通过创建 IP 的方法对这些信号进行关联，又可以直接在期望观测的信号连线上点击鼠标右键，创建 Debug 调试工具。但值得注意的是，使用 Block Design 的方法，对于如本工程中我们前面作为被抓线信号的计数器 cnt 这种设计中的子模块内部信号是难以进行捕捉的。所以，我们在本工程中，仅以捕捉 key_state 线信号的下降沿为实验任务，进行环境搭建方法的演示。

借鉴纯逻辑代码添加 ILA IP 核的设计方法，我们也可以在 Block Design 主窗口页面，添加 ILA IP 核。

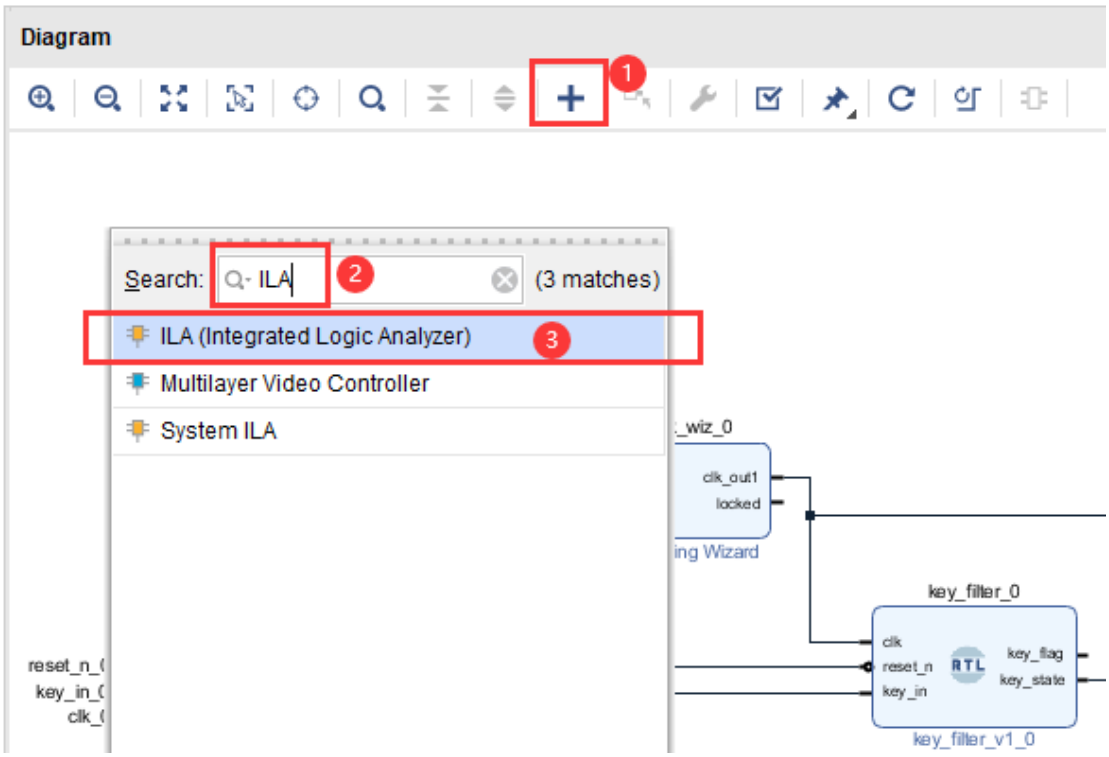


图 1.5-2 添加 ILA IP 核

完成 IP 核添加后，Diagram 页面多出了 ILA 封装模块图示。这时引入的 ILA 工具，默认是给出的 AXI 配置模式。通过双击该封装，可以进入前面介绍过的 IP 配置界面，对该 IP 核进行配置。

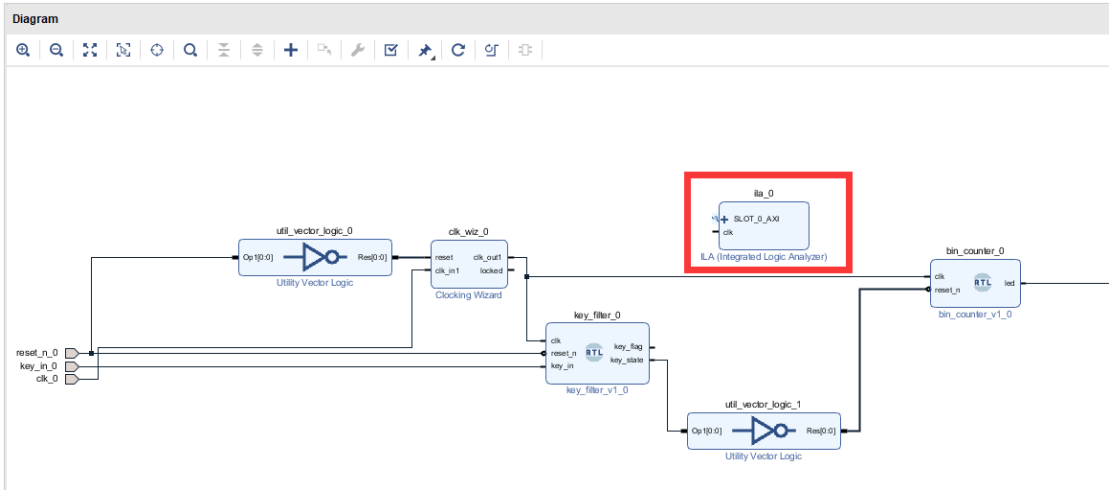


图 1.5-3 主窗口页面多出的 ILA 封装

我们在这里，只需要配置 1 个 1 位的探针，就可以探测 key_state 信号。另外，给 ILA 加上时钟信号的关联，我们很容易完成该 IP 的抓线设置。

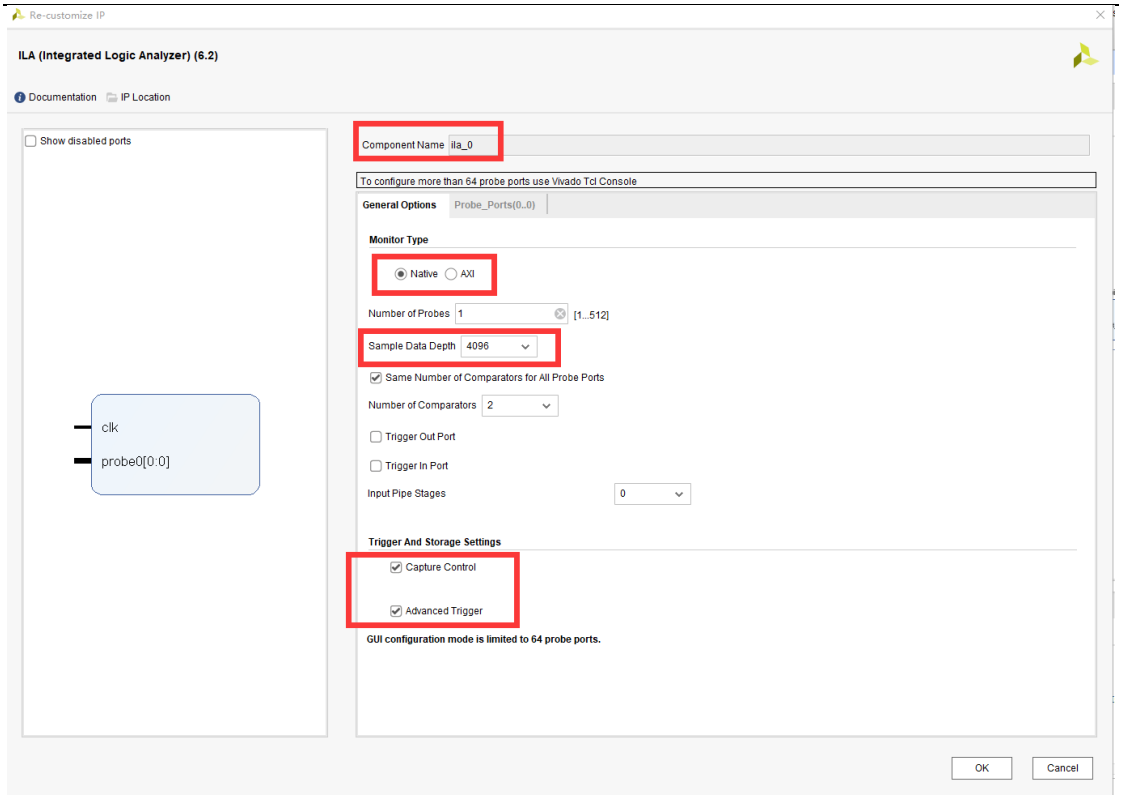


图 1.5-4 配置该 IP 核

完成设置后，我们给 ila 封装的信号完成连线。

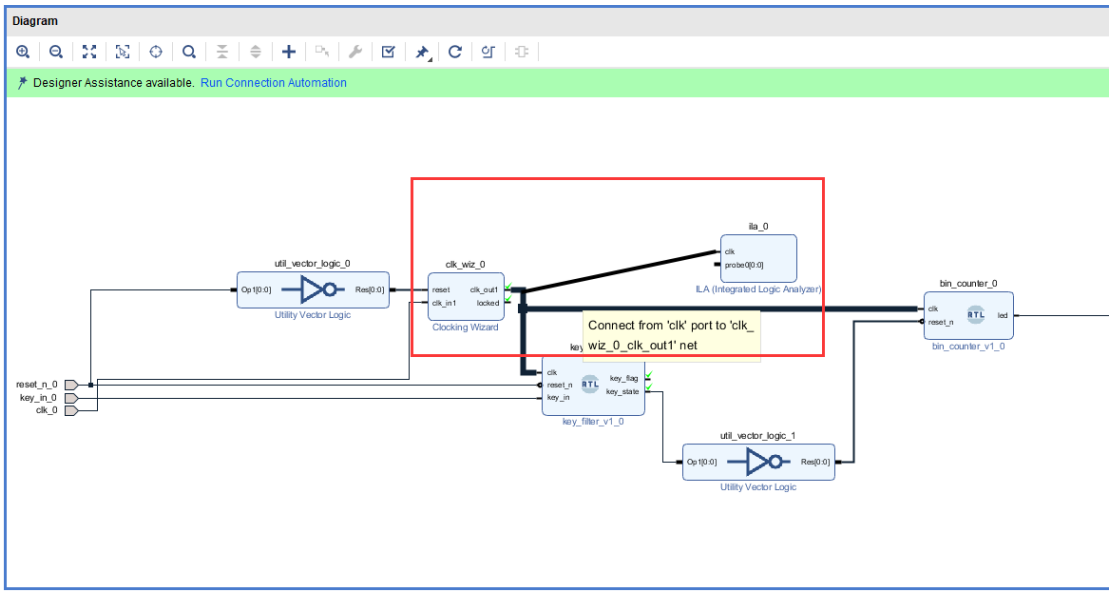


图 1.5-5 连接 ILA 时钟信号

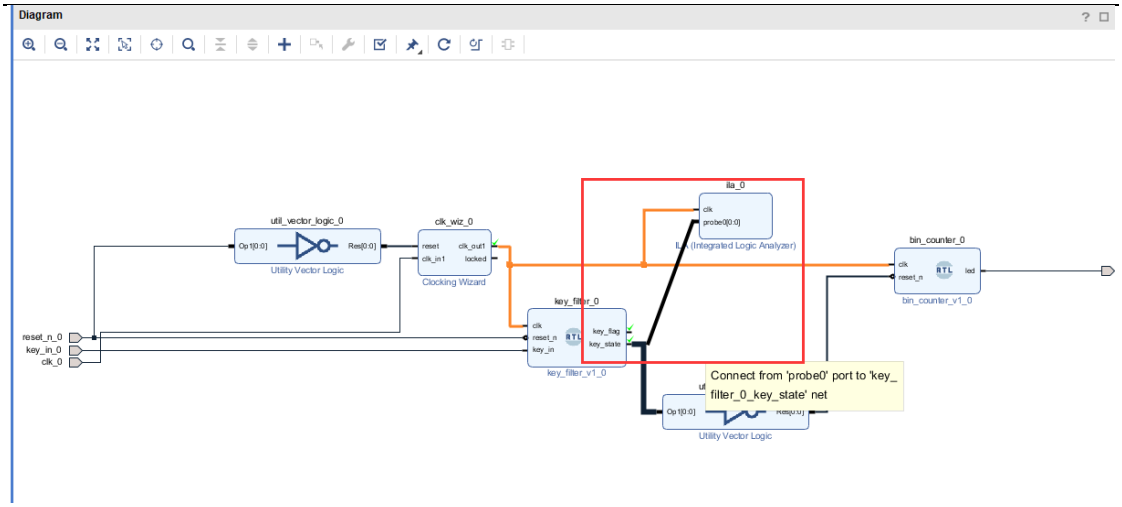


图 1.5-6 连接 ILA 对 key_state 的探测信号

连接完成后，在 Block Design 环境下的 ILA 抓线环境搭建，就完成了。

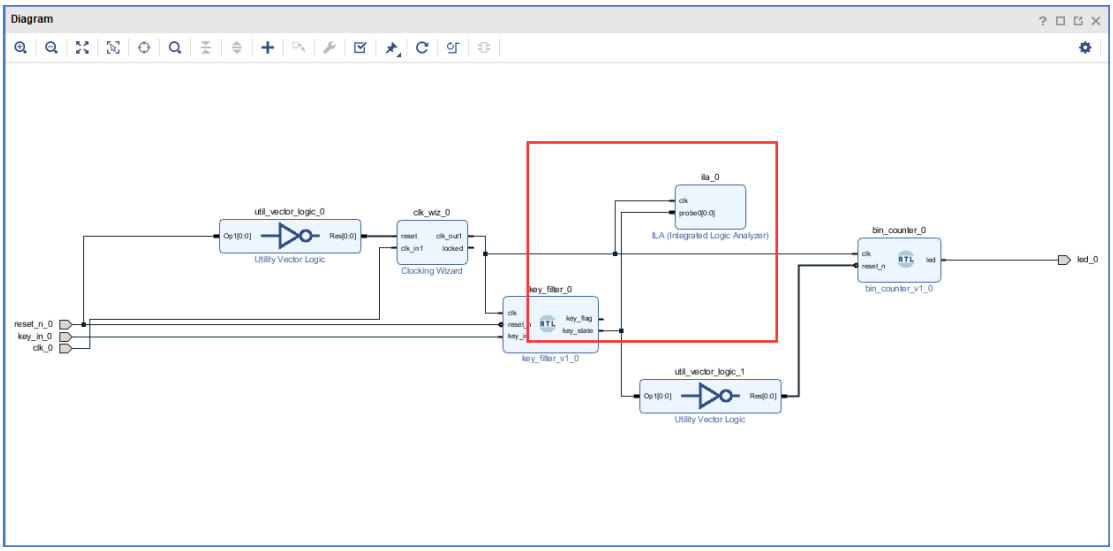


图 1.5-7 完成连接后的 ILA 配置图示

完成环境搭建后，我们给该 Block Design 环境生成工程顶层，绑定管脚，随后在 VIVADO 左侧菜单中直接点击生成 bit 文件，就可以利用该 ILA 进行抓线调试了。

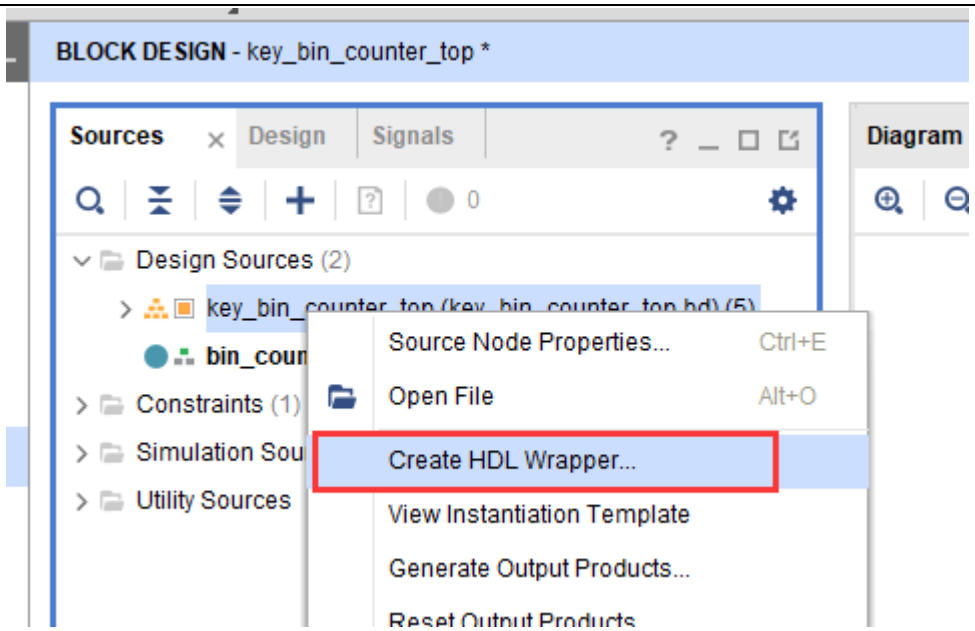


图 1.5-8 生成顶层文件

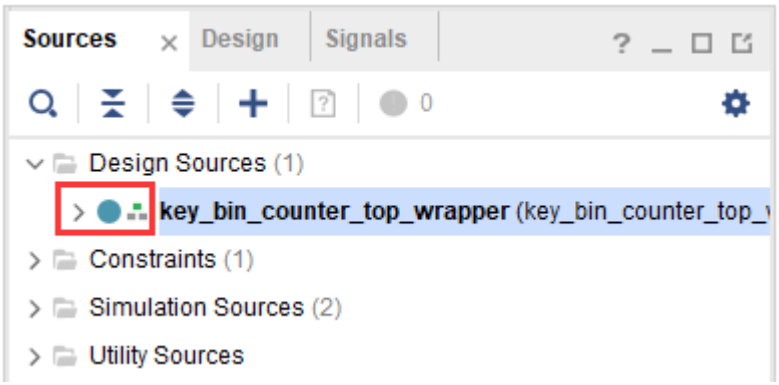


图 1.5-9 确认新生成的文件被置为顶层

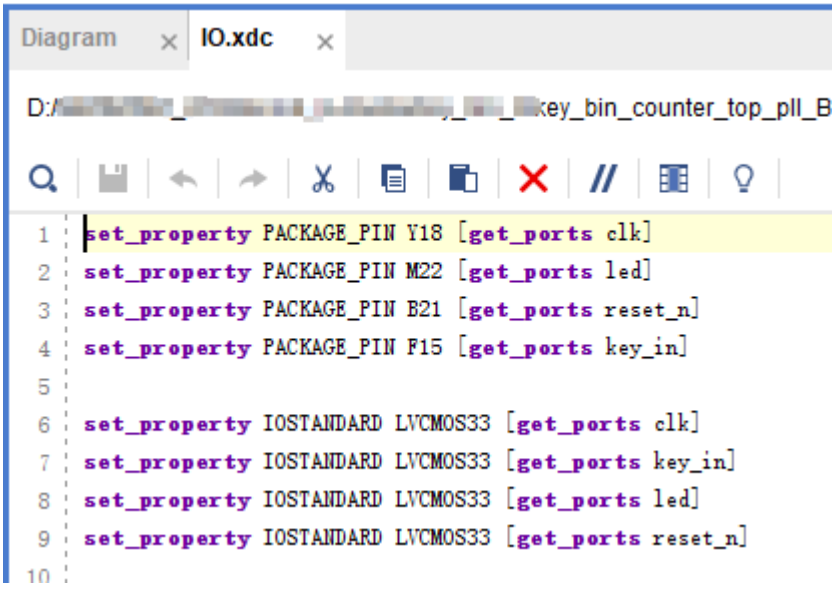


图 1.5-10 工程原始管脚分配

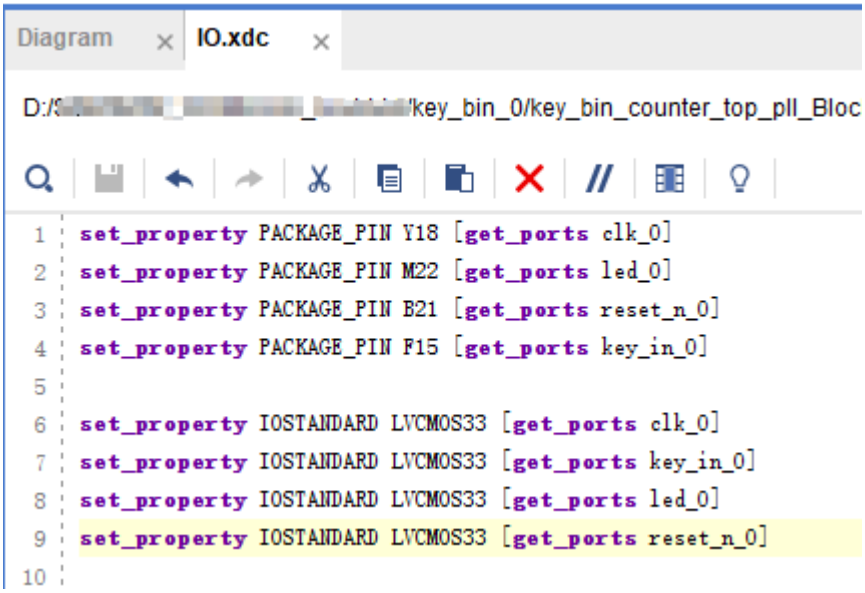


图 1.5-11 根据工程框图引出管脚名称修改后的管脚分配文件

完成管脚分配后，直接生成 Bit 文件。

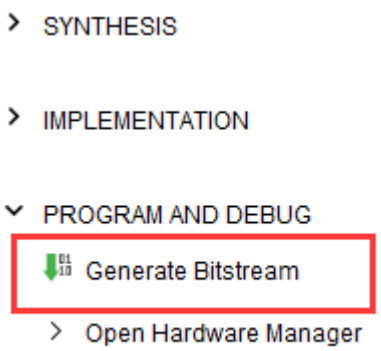


图 1.5-12 生成顶层文件

通过以上介绍,使用 IP 核在 Block Design 界面参与调试的方法就全部讲解完毕了。前面我们也说过，介绍该方法的最终目的，是给设计师在 ZYNQ 等嵌入式开发过程中提供一种有力调试武器。我们在这里也只用逻辑代码代表了各种类型的 Block 封装，当添加 ZYNQ CPU 核，添加其他 IP 核以及自定义 IP 核等 Block 封装后，其调试方法，和这里仍然是一致的。

1.6 生成 bit 文件和 ltx 文件并下载到开发板

前面我们已经介绍了 4 种 ILA 应用环境以及对应应用环境的配置方法。

在对工程进行分析综合、编译直到没有错误，然后生成 bit 文件后，在如下

图所示窗口将生成文件下载到开发板。

图 1.6-1 Bit 文件下载窗口

在下载程序窗口可以看到，除了下载 bit 文件，还同时需要下载 ltx 文件，这个文件生成位置和 bit 文件是在同一个目录下，下载的时候，软件会自动关联到 ltx 文件和 bit 文件一起下载到板子。

1.7 ILA 调试及板级验证

这里，我们以使用 IP 核创建 ILA 调试环境的方法举例。程序下载完成后会自动出现如下图所示界面。该界面即 ILA 调试界面。

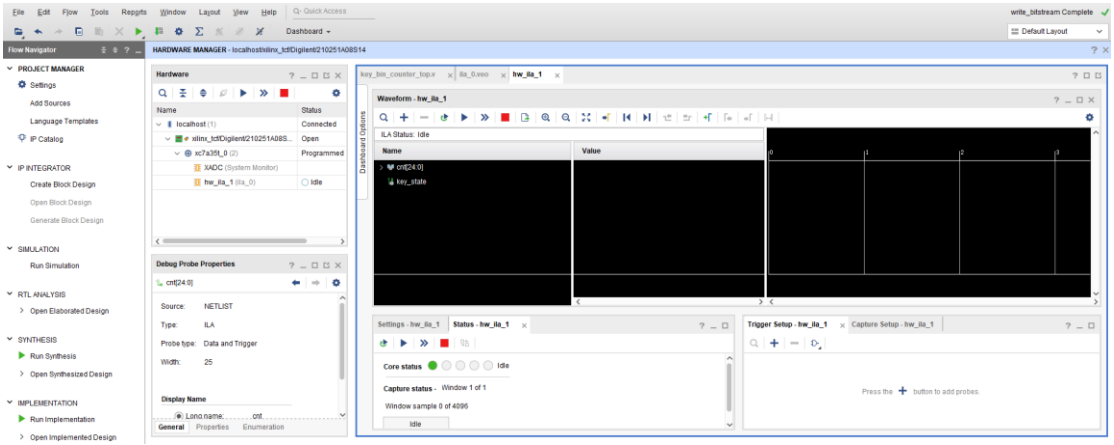


图 1.7-1 下载完 bit 文件后调试窗口

如果有时候 ILA 的采样波形窗口界面没有自动弹出，可以点击左侧 hardware 窗口中带 ila 特征的硬件选择按钮，如本例中 hw_ila_1 以重新启动 ILA 采样波形

窗口界面。

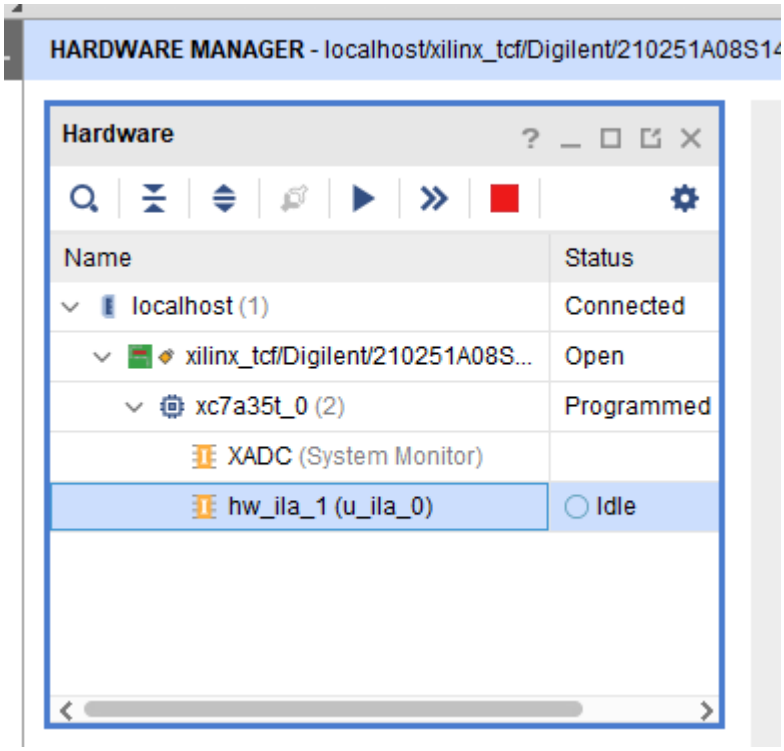


图 1.7-2 在 HARDWARE MANAGER 界面点击重新启动 ILA 窗口

启动窗口后，可以给本次 ILA 调试面板重新进行命名。

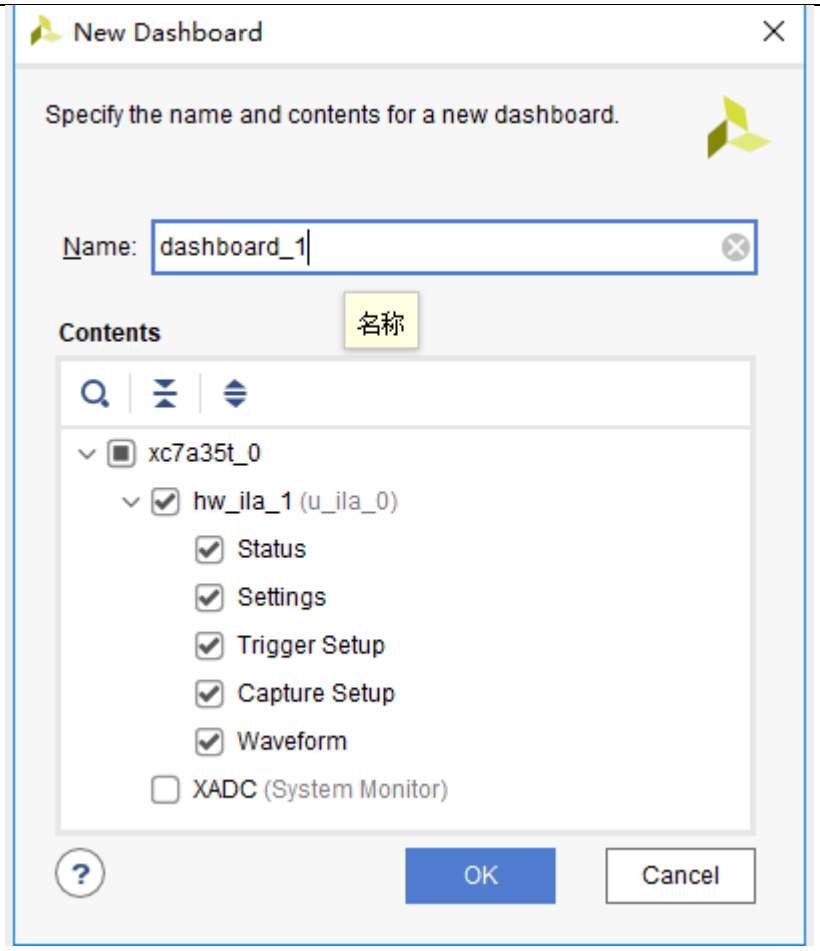
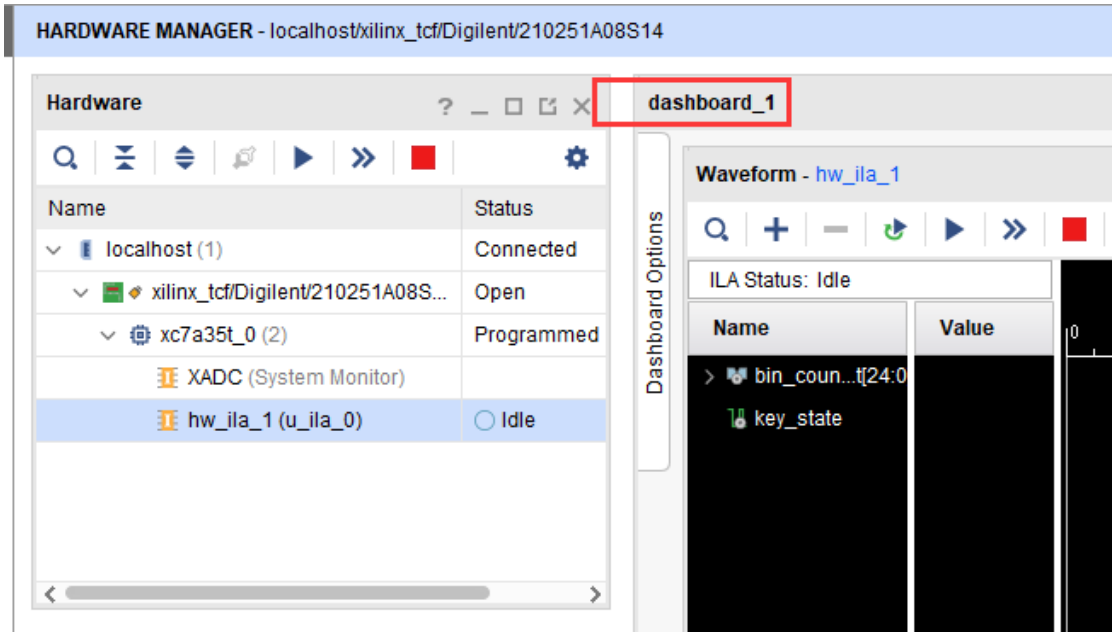


图 1.7-3 自定义本次 ila 窗口命名

此时，右侧窗口重新出现，并且主窗口面板名称被设置为刚刚未修改的默认名称：dashboard_1。



接下来，我们逐一介绍 ILA 调试的各个窗口。如下图所示，是在线调试的波

形窗口。

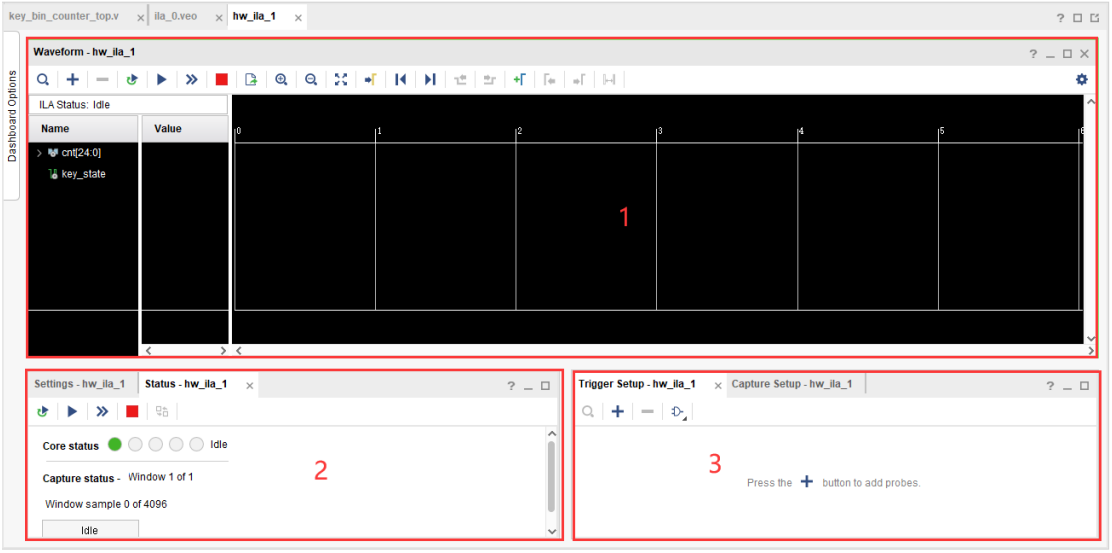


图 1.7-4 在线调试波形窗口 1

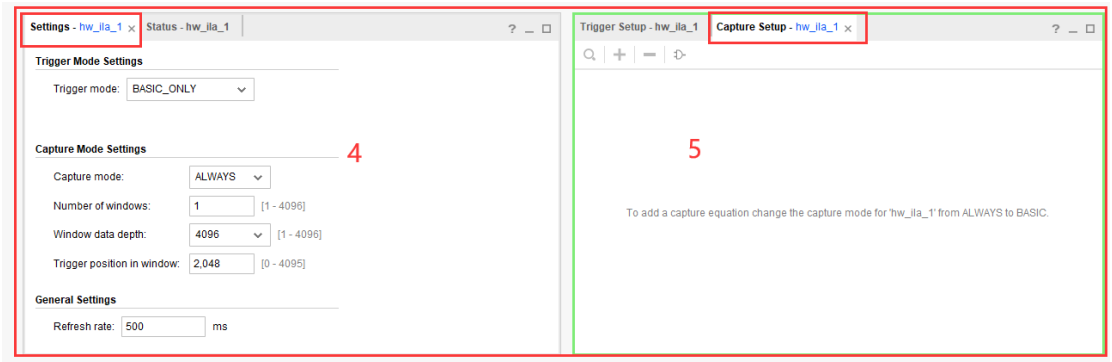


图 1.7-5 在线调试波形窗口 2

上图中的各个子窗口的的功能作用如下。

➢ 窗口 1：波形显示窗口，可以通过点击＋添加想查看的波形的信号。因为在 IP 设置中对探针设置的时候，cnt 和 key_state 信号的 Probe Trigger or Data 属性设置为了“DATA AND TRIGGER”，所以这里可查看的信号有 cnt 和 key_state。当然，通常在下载完程序后，ILA 能够非常智能的识别出设计师期望观察的信号，同时，在右键菜单也可以找到删除观察信号的指令。这样，有限的窗口面积就可以被充分利用，自由调配和增减所需观察信号的条目。

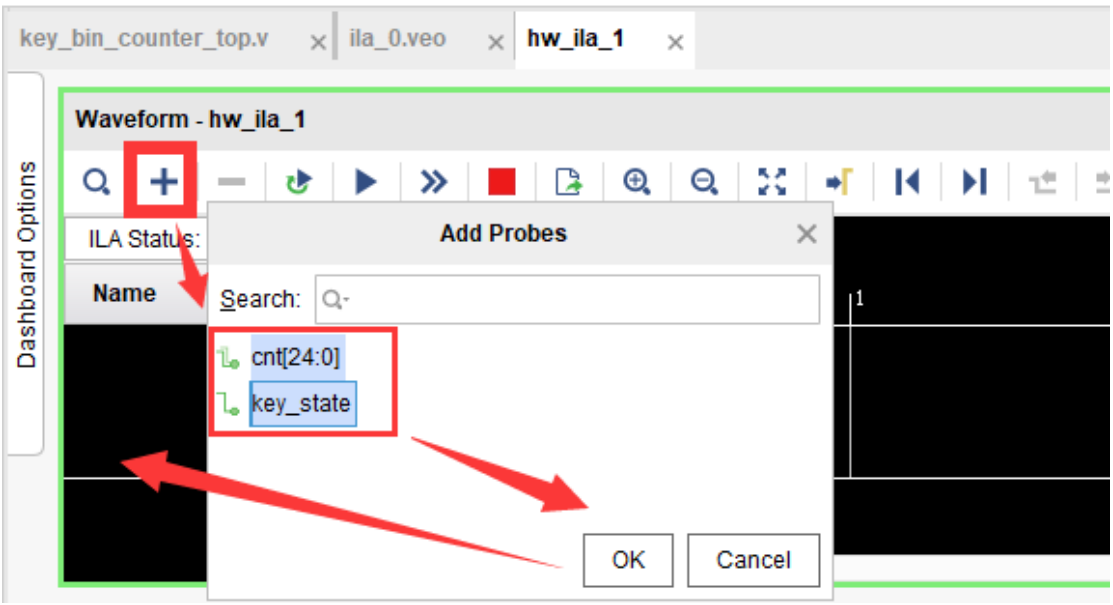


图 1.7-6 添加查看波形窗口

➢ 窗口 2：ILA Croe 的状态控制和显示窗口。

图 1.7-7 窗口 2 面板(1)

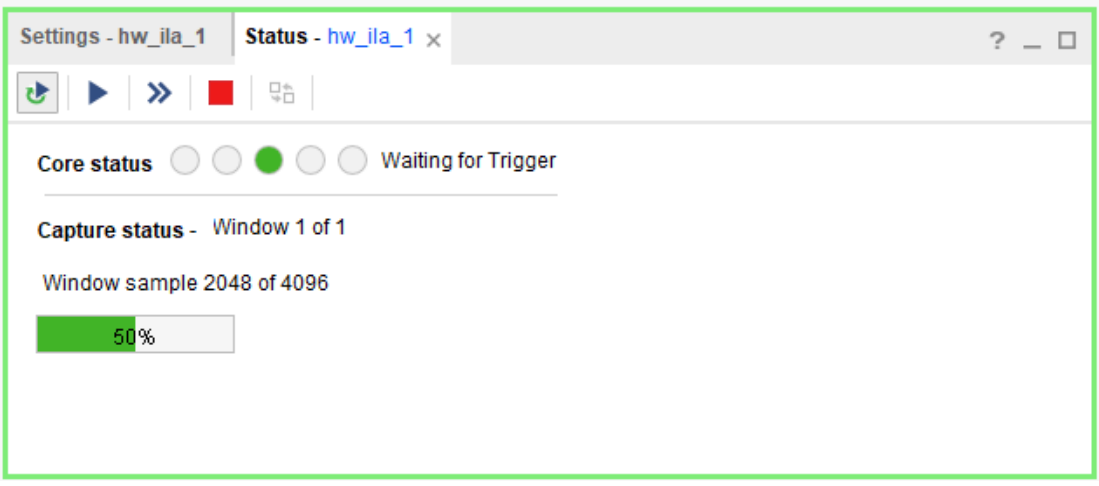


图 1.7-8 窗口 2 面板(2)

如图 1.7-7，从该窗口中，可以对 ILA 采样过程进行控制，并观察 ILA 的执行状态和步骤。窗口中：

按钮①：设置采样执行过程为循环采样。如果该按钮按下，则其外框为灰色线条，且框内颜色微深于面板背景色。(观察比较图 1.7-7 和图 1.7-8)

按钮②：启动采样按钮。按下后将启动 ILA 采样。根据是单次采样还是循环采样，根据是有条件触发还是无条件触发，启动后会呈现 ILA 状态的周期性变化或等待触发条件的到来。

按钮③：无条件执行 ILA 采样。

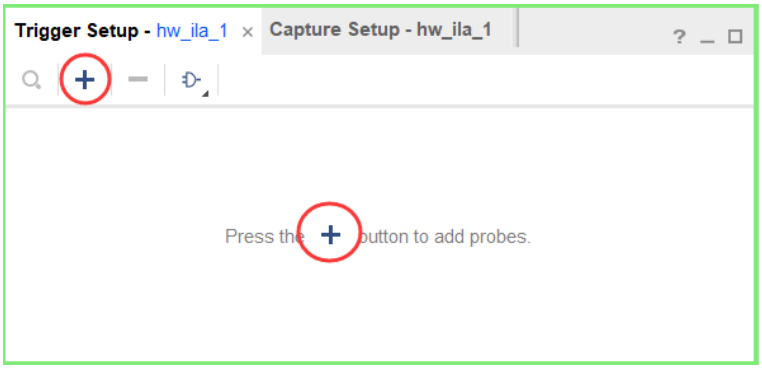
按钮④：停止采样按钮。如果处于循环采样状态，按下此按钮后，本轮正在执行的循环结束则停止采样。

状态栏⑤：状态栏⑤由 5 个空心圈组成，唯一的实心绿色点所处的位置表征 ILA 的运行状态。

状态栏⑥：已采集的点数占总采集点数的关系。由于图 1.7-8 中 TRIGGER 设置的在波形窗口 2048 点展示，而 TRIGGER 还未到来，所以采样完成了 2048 个点而正等待 TRIGGER。

状态栏⑦：以进度条的方式结合状态名称，标明当前所处的状态以及采样过程中执行状态的百分比。

➢ 窗口 3：添加触发信号窗口，如下图所示，点击窗口内 ＋ 添加产生触发条件的信号（两个地方的 ＋ 都可以进行信号的添加）。



点击后，在弹出的 Add Probes 窗口选择需要添加的信号。因为在 IP 设置中对探针设置的时候，cnt 和 key_state 信号的 Probe Trigger or Data 属性设置为了“DATA AND TRIGGER”，所以这里可查看的信号有 cnt 和 key_state。

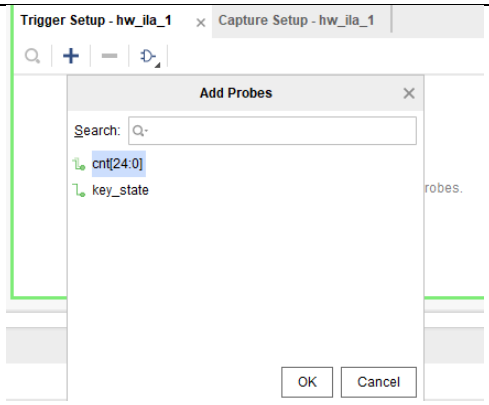


图 1.7-9 添加触发信号窗口

添加触发信号后的窗口如下图所示。

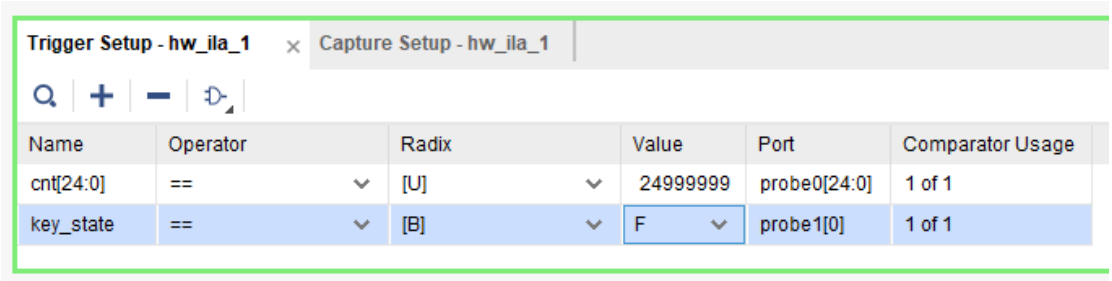


图 1.7-10 添加触发信号后的窗口

添加好触发信号后，需要设置触发条件，触发条件有多个地方可进行设置。假设现在我们想设置在满足 cnt == 24999999 并且 key_state 为时触发进行一次信号的抓取，设置如下：

① 设置触发情况，下面是设置多个信号产生触发信号时的条件，与、或、同或、异或，这里根据需求设置“Set Trigger Condition to ‘Global AND’”表示。

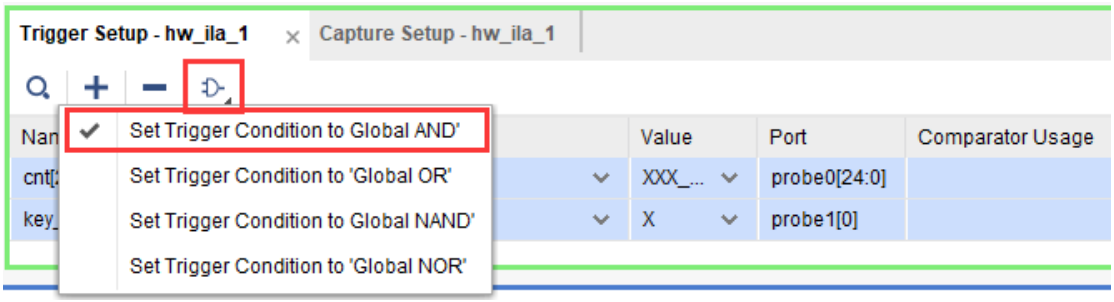


图 1.7-11 设置多个触发条件同时判断时的逻辑规则

② 设置操作类型

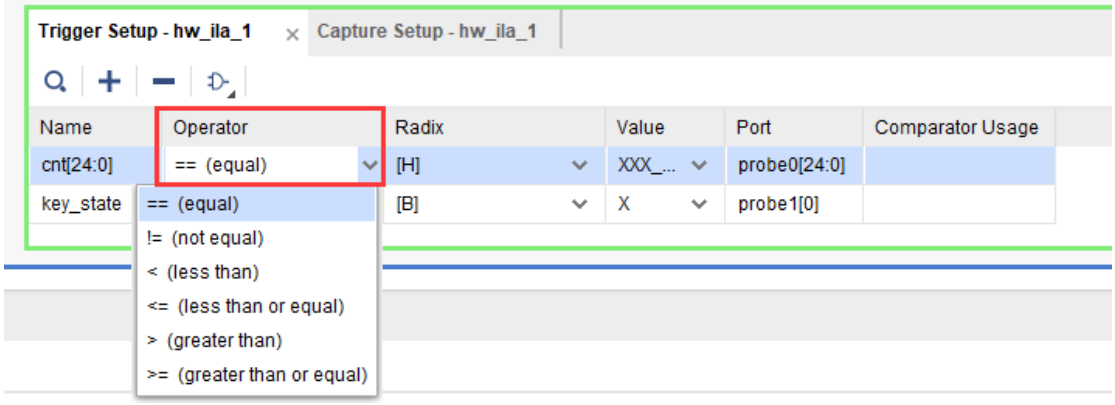


图 1.7-12 设置操作类型

③ 设置信号数值的进制，这里对计数器 cnt，可以选择无符号的十进制数，

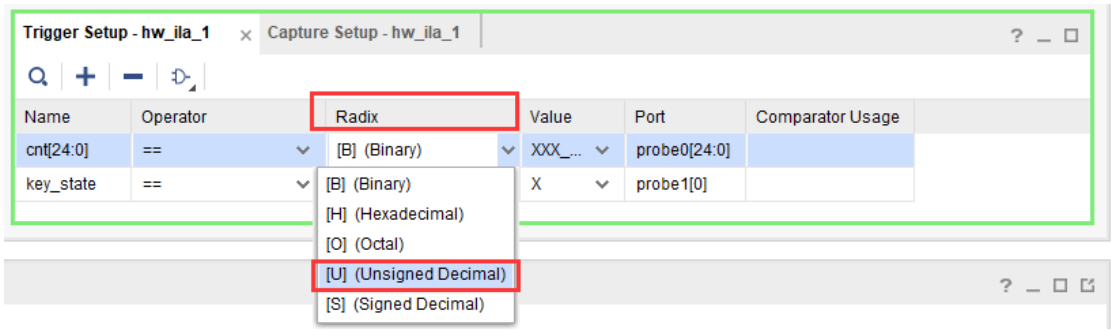


图 1.7-13 设置信号数值的进制

④ 设置数值 24999999

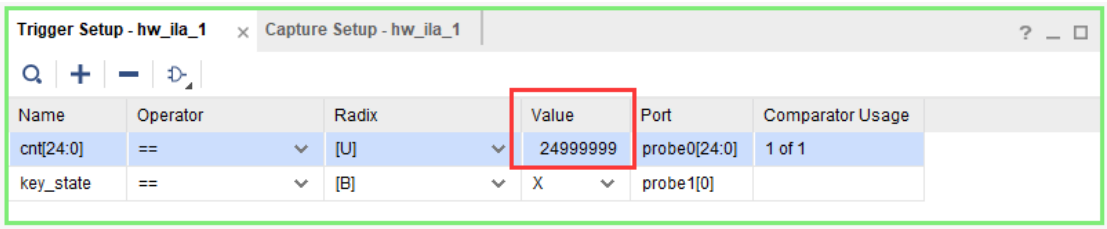


图 1.7-14 设置信号数值

⑤ 同样的方式也可以对信号 key_state 进行设置，使之低电平作为触发条件。

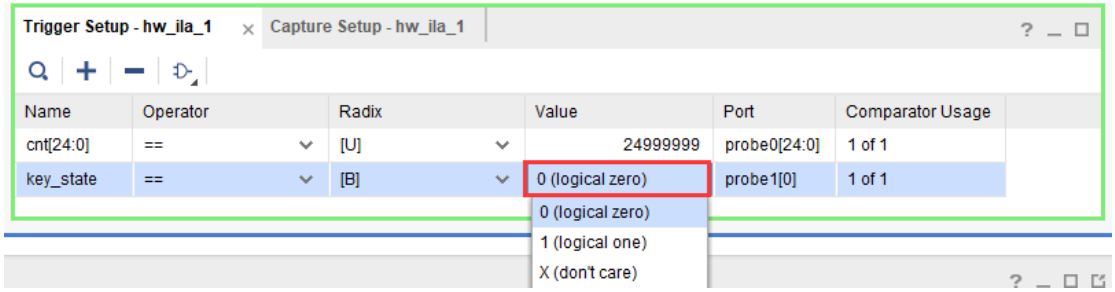


图 1.7-15 同上法设置信号 key_state

触发条件设置完后，点击 Run。

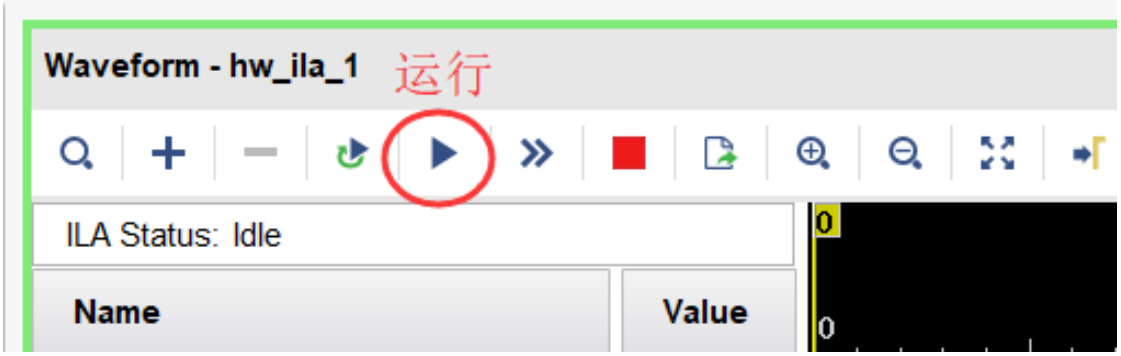


图 1.7-16 启动 ILA

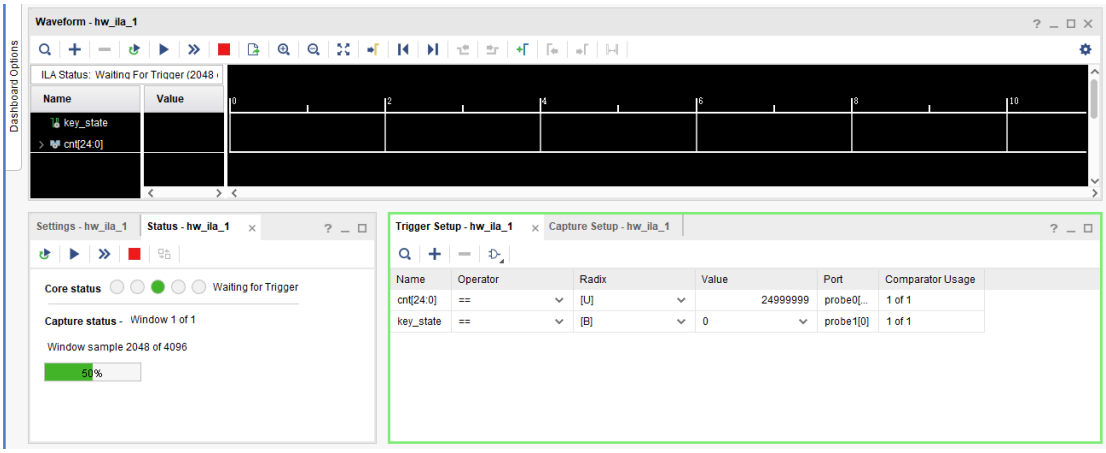


图 1.7-17 等待触发条件

这时，可以看到 ILA 正在等待触发条件。由于在该例程中，我们设置了开发板按键 S0 按下经过消抖后则计数器开始工作，此时，我们只需按下 S0 即可触发 ILA 采样。

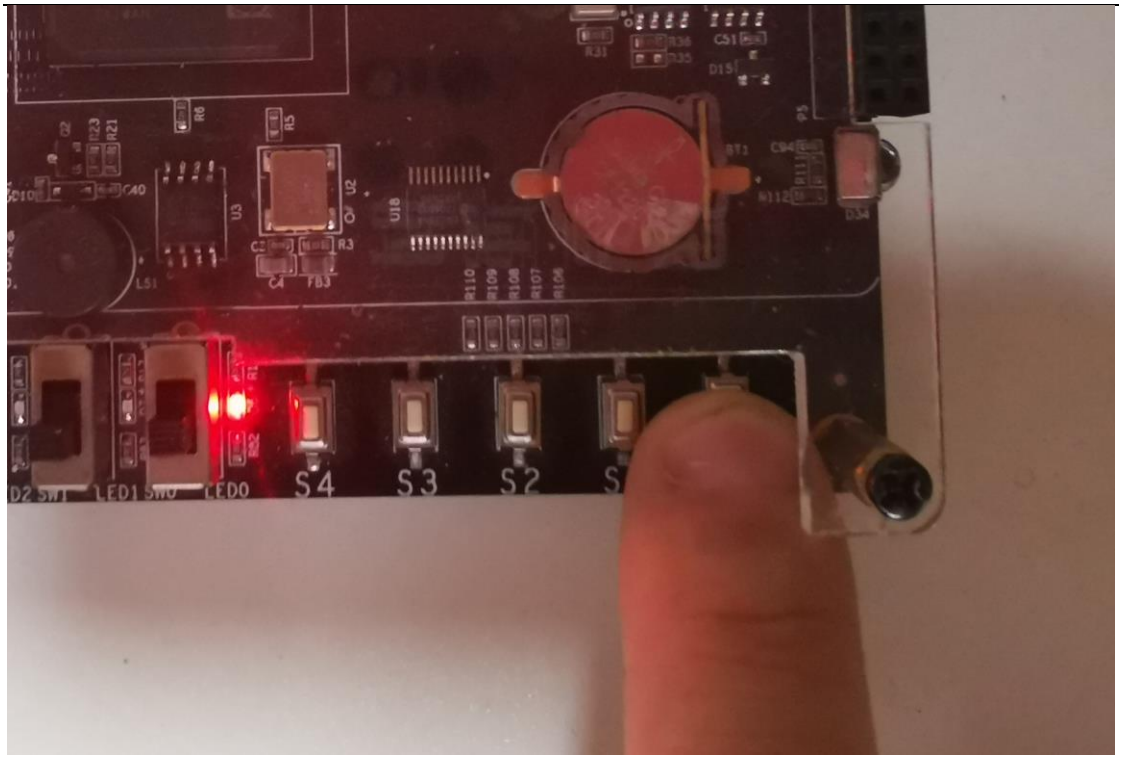


图 1.7-18 点击管脚绑定的按键触发启动条件

在这里，我们设置的采样模式为单次采样，因此，主窗口页面会出现一次采样波形并不会再刷新。

可以看到满足触发条件后的波形窗口。波形中有红色 T 标记的符号，正好处在第 2048 个采样点处，即我们预设的 4096 个采样点的中点。

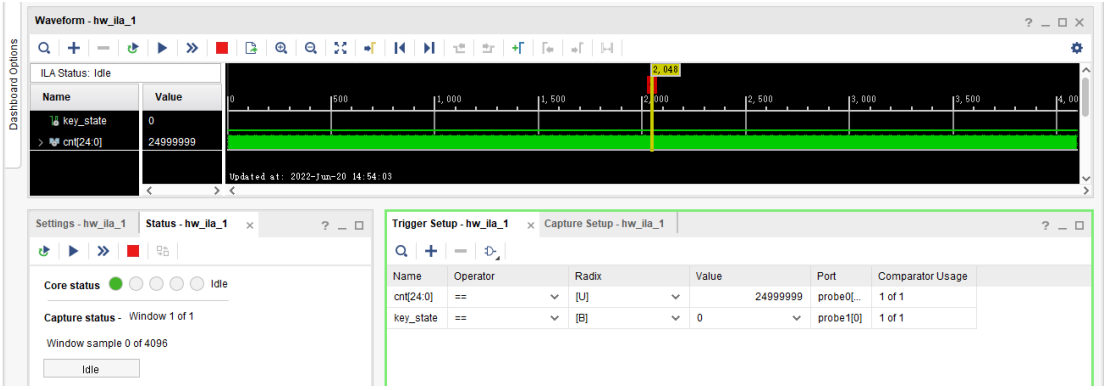


图 1.7-19 采集完成后总图

点击波形放大，可以清楚的看到触发处的波形。

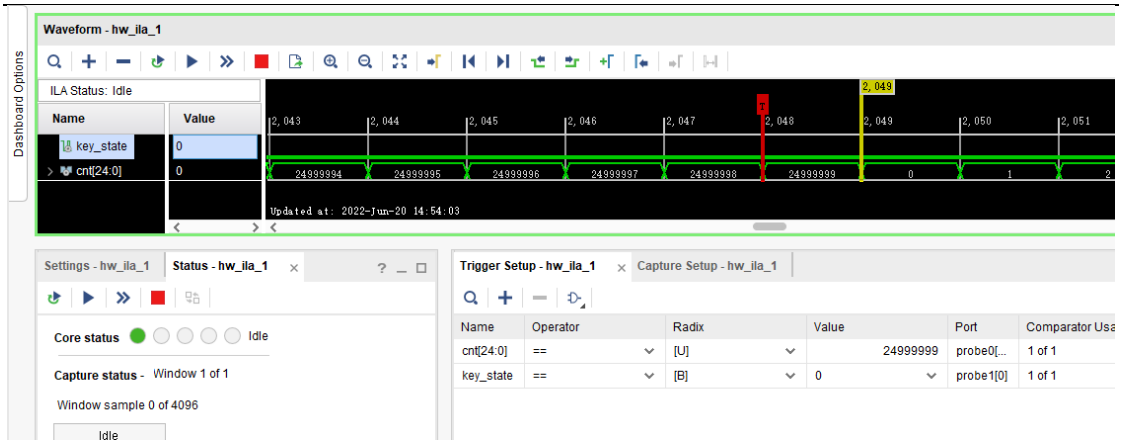


图 1.7-20 触发点处的波形

和仿真波形数值显示的数制调整方法一致，我们可以在数值列表中选择右键调整数制，以使波形中展示的数值便于我们的观察。例如，如果您先前设置的波形信号的数值采用的是 16 进制显示，在这里可以更改设置让其采用 10 进制显示，具体设置如下。

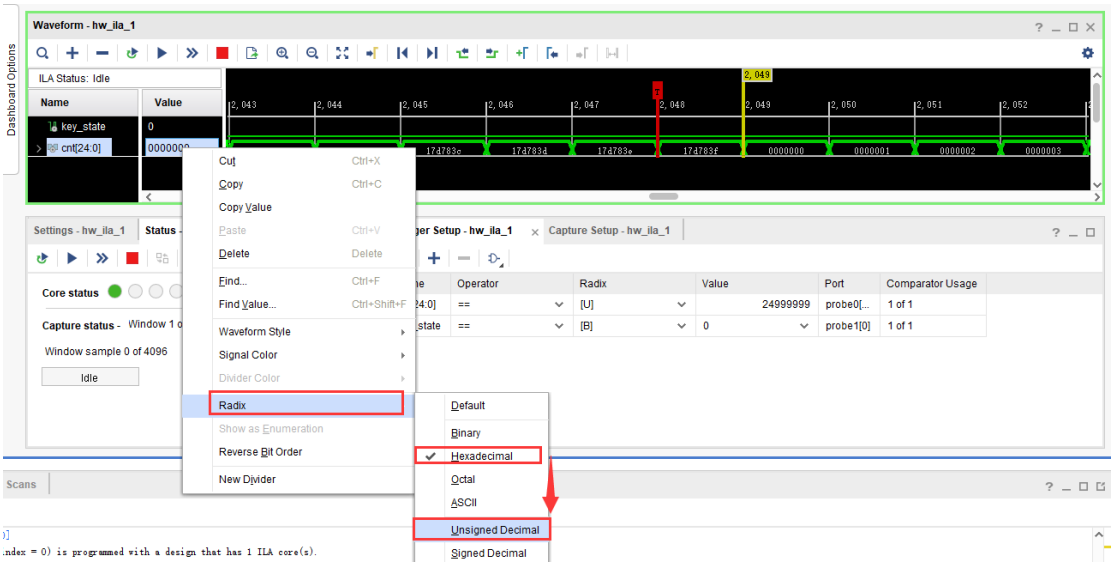


图 1.7-21 设置波形数据的进制

更改数值显示为十进制后的波形显示如下：

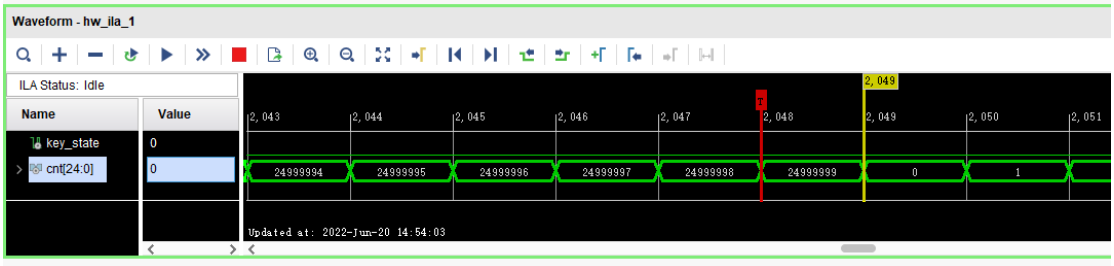


图 1.7-22 更改数制后的波形显示

虽然完成了本次 ILA 抓线的既定任务，但作为学习教程和内容，我们有必要对一些深层次的设置作一些了解。所以，我们继续沿着介绍 ILA 窗口的思路讲解。

窗口 4：设置触发显示窗口，点击窗口右上角的方框放大这个窗口，可以看到这个窗口全部的设置参数。

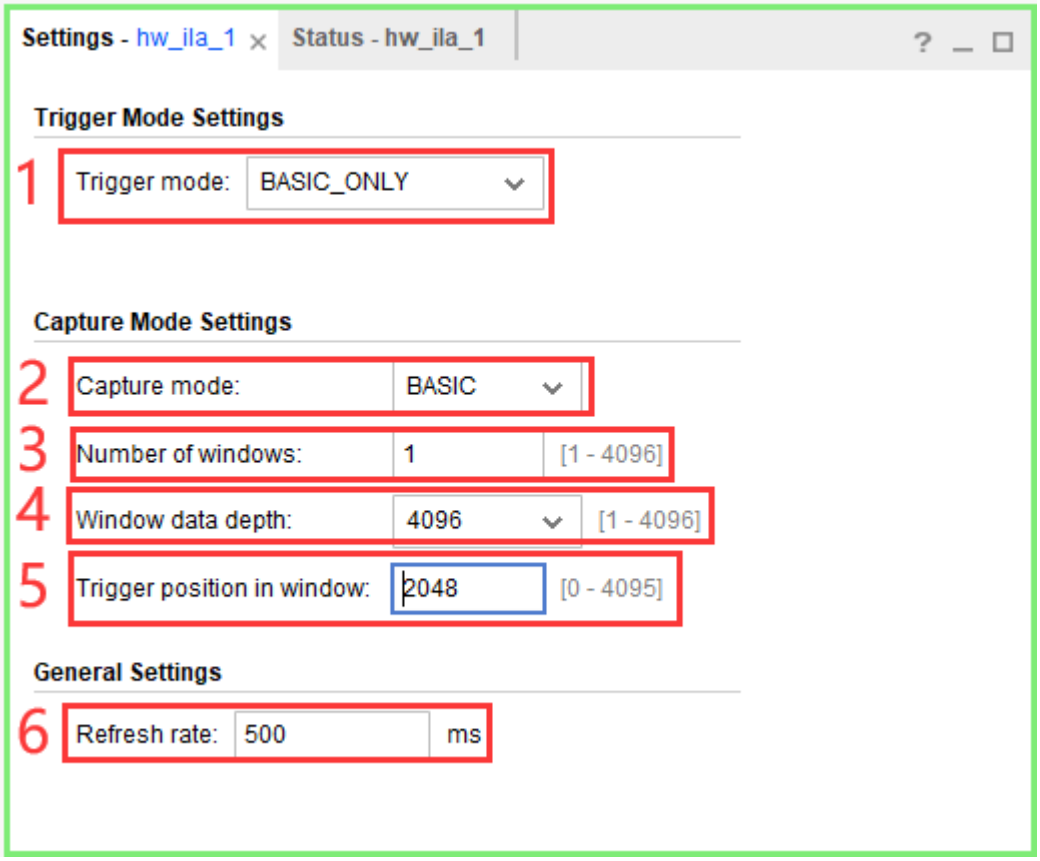


图 1.7-23 设置触发显示窗口的参数

（1）Trigger mode：触发模式设置，选择默认的 BASIC_ONLY 即可，有关 ADVANCED_ONLY 模式的具体使用，可查看 IP 手册。

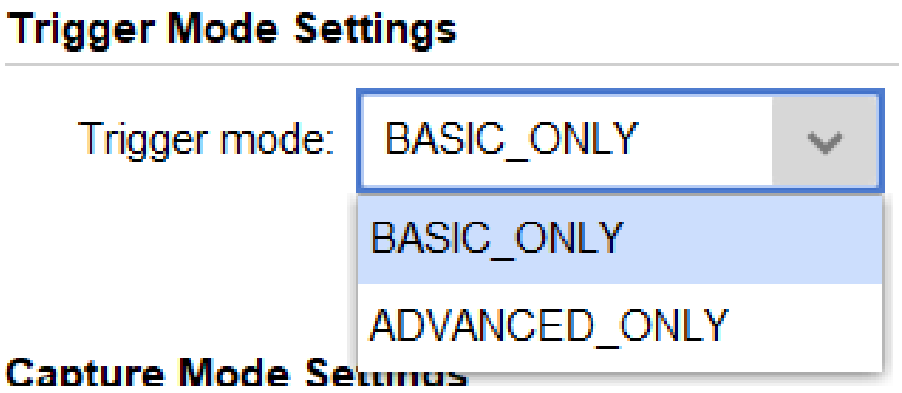


图 1.7-24 触发模式设置

（2）Capture mode：捕获模式设置，选择默认的 BASIC 即可，有关 ALWAYS

模式的具体使用，可查看 IP 手册。

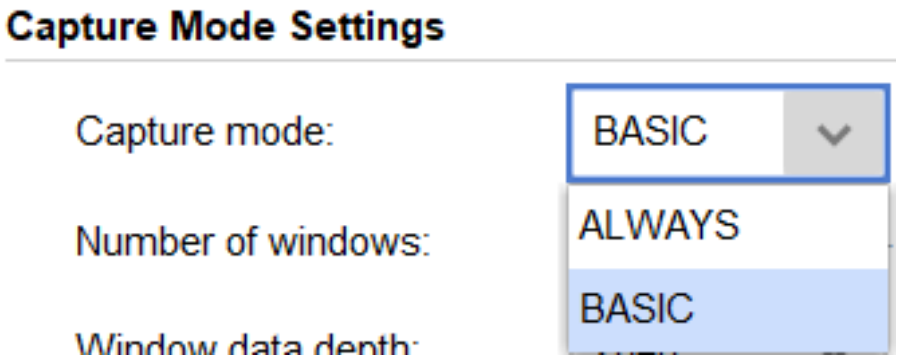


图 1.7-25 设置捕获模式

注：有关上述（1）（2），如果在前面的 IP 创建的时候不勾选 13、14 处的选项，上述（1）（2）是不可设置的。

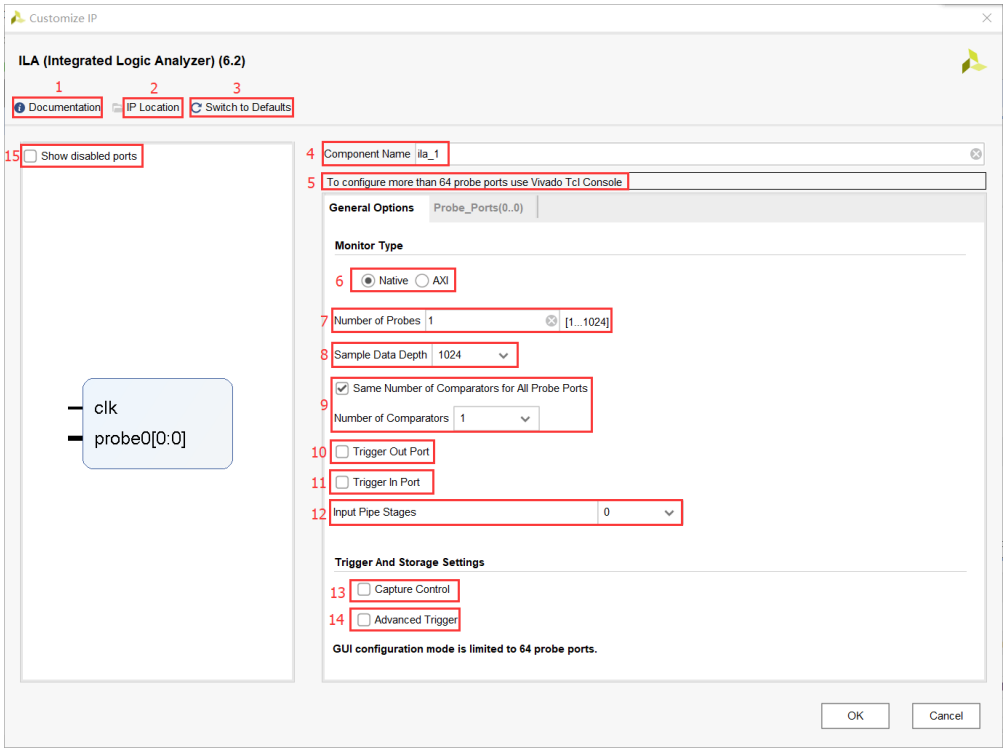


图 1.7-26 回顾前面 IP 创建时的设置界面

（3）Number of windows：显示窗口个数，默认为 1，更改这里为 2。窗口个数的设定值可以设定在同样的触发条件下，连续多次的触发显示在不同的窗口中。随着窗口数量的增加，势必会使得单个窗口的采样深度变小。所有窗口采样深度的总和不能大于总的采样深度。这里更改后下面 Windows data depth 的最大值会自动变为 2048（4096=2*2048=4*1024=……）。

（4）Windows data depth：窗口显示数据深度，上面设置为 2 后，这里最大可设置为 2048，设置 2048 即可。如果仅设置 1 次采样，则其最大值为 IP 核配置、抓线设置中采样的最大值。

（5）Trigger position in windows：设置触发位置在窗口波形中显示的位置，一般根据需求设置，这里设置波形窗口的中间位置，即设置为 1023

（6）Refresh rate：连续触发模式下，相邻触发之间的刷新时间，这里保持默认即可。

设置完这些参数后的如下图所示：

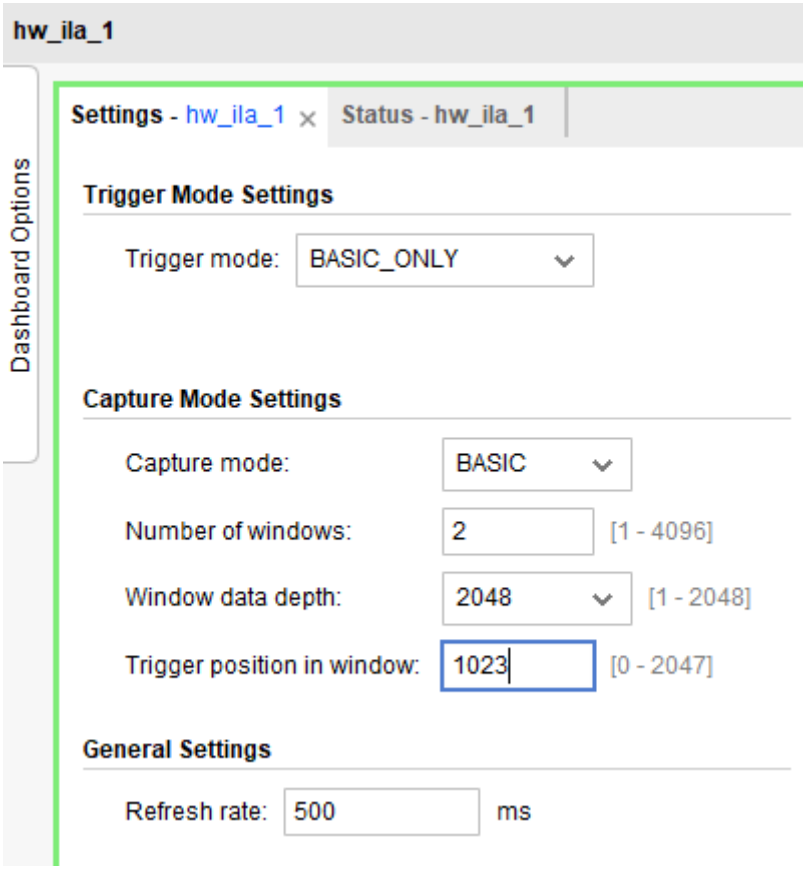


图 1.7-27 触发条件参数设置完成后界面

设置完后，点击 Run，同时给出按键按下的触发条件。

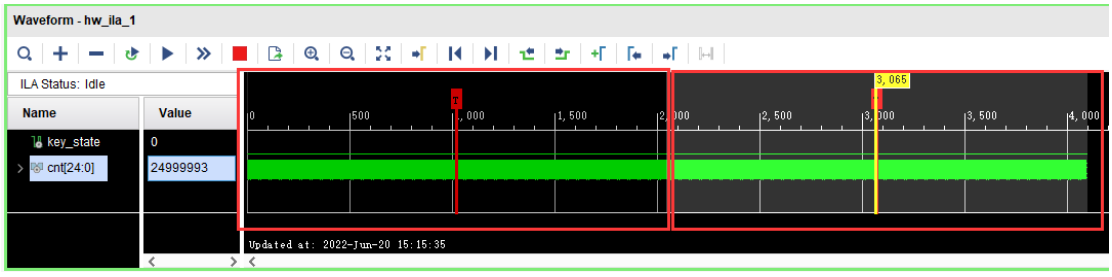


图 1.7-28 使用新设置的触发条件参数启动 ILA 工具得到波形

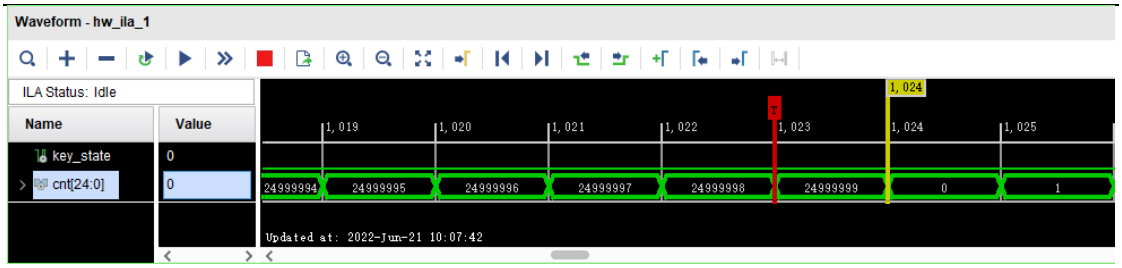


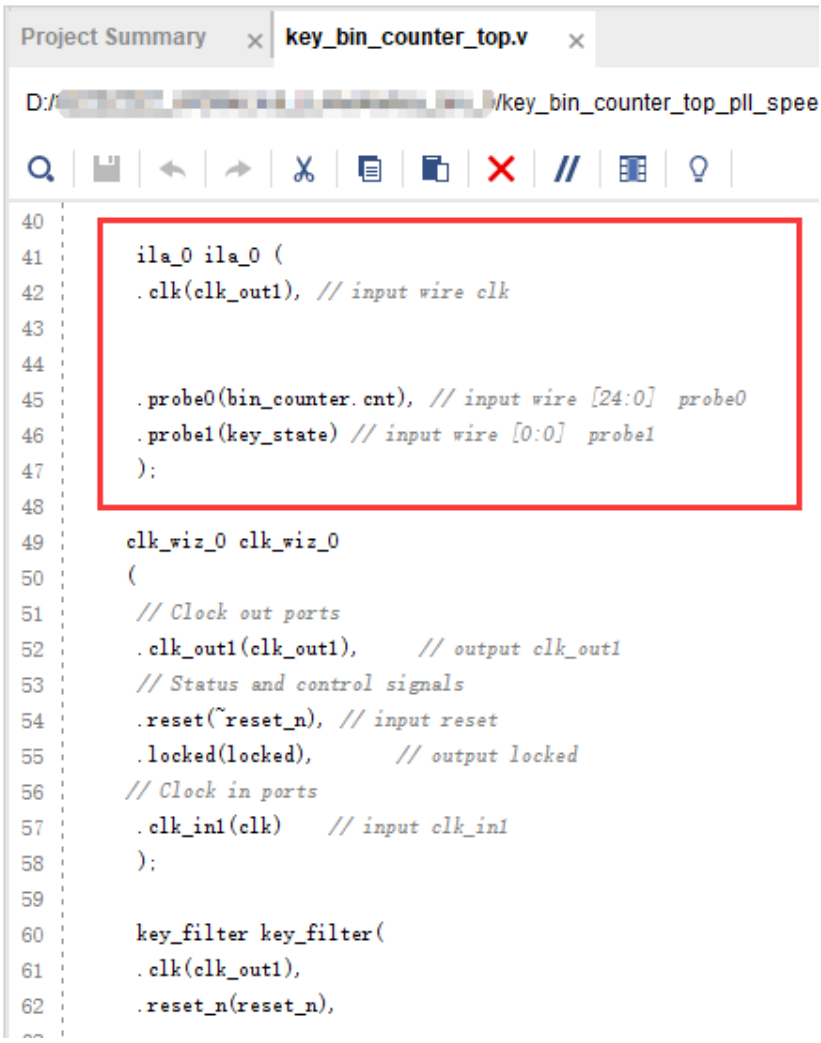
图 1.7-29 放大第一个窗口波形

从触发的波形可以看出，与之前波形相比，现在是出现两次触发波形，与我们上面设置的窗口个数是对应的。同时，红色触发标记点，正好在第 1023 个采样点出现，和我们刚刚在图 1.7-27 的 Settings 中设置的触发点位置是一致的。ILA 工具为了区分两次采样的不同阶段，使用了纯黑色和淡灰黑色的底色，作了细微的划分。

整个在线调试工具 ILA Croe 的使用基本就讲完了，如果需要了解更多有关 ILA 的使用，可以进行更多的尝试和查看 IP 手册。

1.8 对 ILA 的工作时钟频率进一步探究

前面，我们探讨的范围，都限于固定频率。为了研究 ILA 和频率的关系，我们将外部晶振 50M 输入接入锁相环，使之按同频率同相位输出。输出的时钟信号，直接提供给 ILA 及工程的功能模块。



事实上，ILA 的工作频率，是有范围要求的。得益于我们通过锁相环可以在一定范围内设置任意设置时钟频率，我们接下来，调整锁相环不同输出频率，测试 ILA 的工作频率范围。这里，我们以使用 IP 核创建 ILA 的纯逻辑代码工程，进行举例。

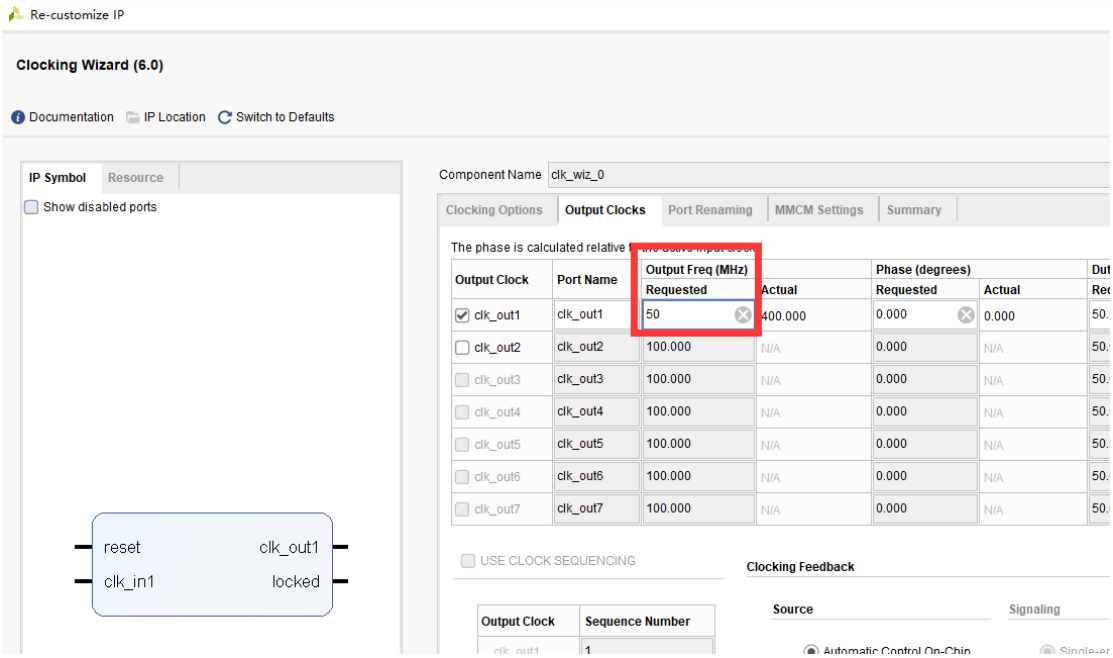


图 1.8-1 修改锁相环频率

我们进行了一系列测试，并得出了不同频率的测试现象。ILA 可正常运行的频率区间，大致为下载器时钟频率信号的 2.5 倍至 VIVADO 报告工程时序违例的时钟范围区间。

其中，有如下报错给我们的 ILA 工作频率设置范围提出了一个非常重要的要求：

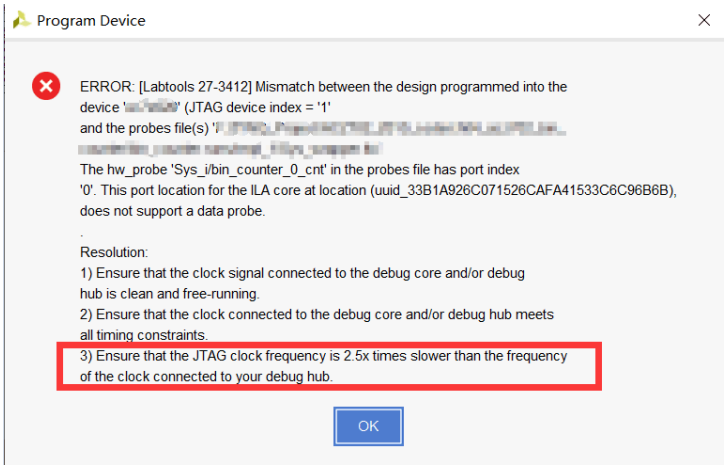


图 1.8-2 ILA 工作频率设置范围提示

这条要求指出，ILA 的时钟频率，不得低于 JTAG 下载器的时钟频率的 2.5 倍。事实证明，过低的 ILA 时钟频率，会导致 ILA 工作不正常，无法启动，状态跳转不正常或报错等非特定现象的发生。当然，ILA 的工作频率也不是无线高的，过高的频率，会导致工程时序违例，从而发生无法抓取波形或者 VIVADO 无法识别到 ILA 等问题发生。

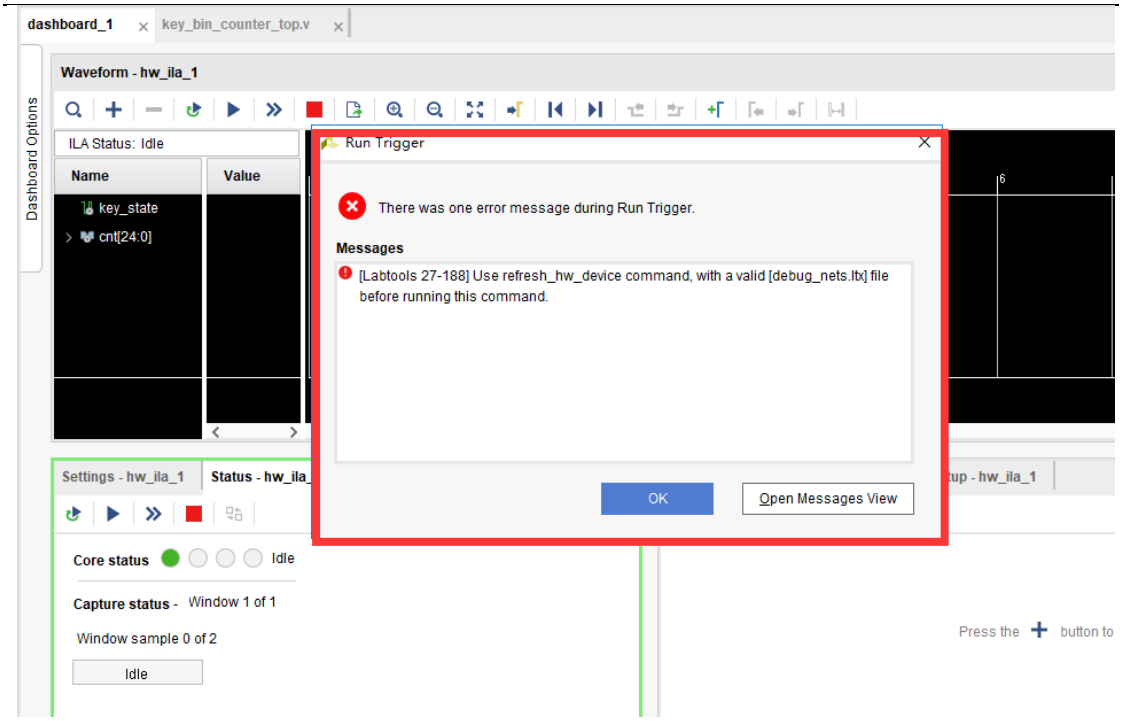


图 1.8-3 将 ILA 工作时钟设置为 5M 时发生意外报错

有时候，我们需要测试的工程，其 ILA 时钟频率非常低，并且不允许调整。遇到这种问题，我们这里，提供另一种解决该问题的思路，即降低 JTAG 下载器的频率。

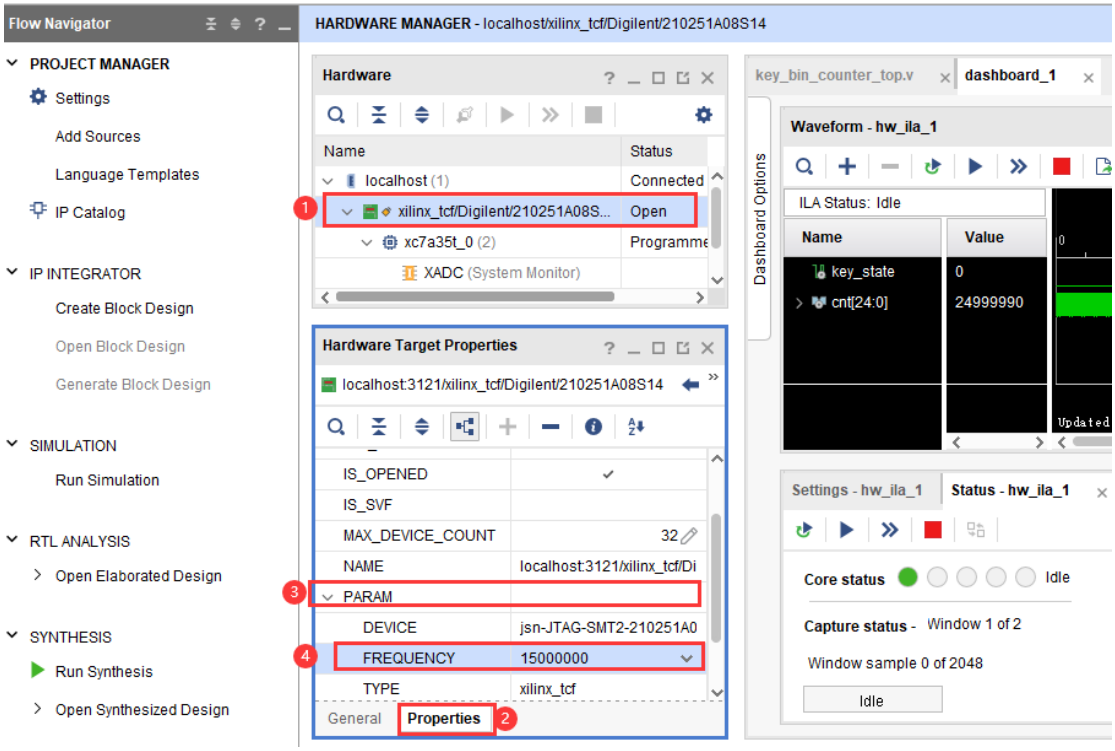


图 1.8-4 设置 JTAG 下载器工作频率

根据前面讲解的 JTAG 频率和 ILA 频率的倍数关系，我们选择合理的 JTAG 下载器频率，按前面 VIVADO 提示的频率关系，在一定程度上迁就 ILA 的工作频率，这样又可以使 ILA 达到正常工作的频率范围了。

1.9 总结

本章节讲解了 4 种 ILA 工具的生成和使用方法，以完成 ILA 的配置和工作环境搭建。在板级验证环节，又详细介绍了 ILA 窗口的各个面板按钮的功能和设置用途。最后，又进一步分析了 ILA 时钟频率和下载器 JTAG 调试时钟频率的关系，并介绍了如何通过降低 JTAG 时钟频率来一定程度满足 ILA 的工作时钟频率的方法。希望读者能跟随本文档讲解的内容，结合应用实际，完整掌握 ILA 工具的使用方法。