# 实验三 时序逻辑实验

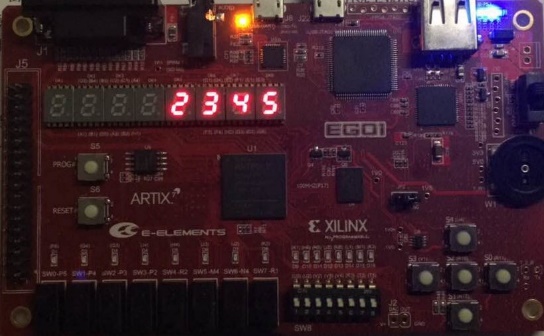
## 实验目的

本实验的目的是学习时序逻辑模块在数字系统中的综合应用；掌握实验平台的外部功能模块在数字系统设计中的应用。

## 实验内容

1. **实验3.1——在七段数码管上滚动显示学号**：按照“4 实验步骤——在七段数码管上滚动显示学号” 完成本实验。以学号40123456为例：
2. 首先将学号中的8个数字存储在一个32位的寄存器msgArray中，每4位存放一个十进制数字；
3. 4个数码管始终显示寄存器的高16位数据；
4. 用频率为3Hz的时钟控制7段数码循环显示：在时钟的上升沿进行向左循环移动4位，并显示。注意：记得要把msgArray中的内容，即msgArray [31:28]的内容移到msgArray [3:0]中。
5. 复位时，寄存器恢复原始存储状态，7段数码管显示第一组4位字符（寄存器中的高16位），即4012；

下图显示了通过4个数码管滚动显示学号（40123456），初始时刻和经过3个周期以后的效果：

t(0)时刻 t(3)时刻

图1 数码管滚动显示学号例

1. **实验3.2——通过按键输入学号，并滚动显示**：

电路功能描述：通过Ego1上的按键输入自己的学号（8位10进制数），并存储在32位的寄存器中；8位10进制数输入完成后，实现类似实验3.1中的滚动显示效果。

除了要求实现上述功能外，还包括如下具体实现上的要求：

1. 整个电路具有**复位功能**；
2. 8个数码管中，4个数码管用于输入数据**后**的循环显示，另外4个用于显示当前**正在输入**的数据。
3. 输入过程可控、实时可视，比如可采用下面逻辑：
   * 用按键或者开关控制输入开始和结束，输入开始后用于循环显示的4个数码管停止循环显示，用于显示输入数据的4个数码管开始显示输入数据；输入结束后用于循环显示的4个数码管开始循环显示输入的学号，用于显示输入数据的4个数码管停止显示输入数据。
   * 用按键切换当前要输入的数据位，切换结果通过用于显示输入数据的4个数码管展示出来，即用户能够通过显示输入数据的4个数码管看出来当前正在输入的是哪一位10进制数。
4. 输入数字的时候进行**按键消抖**（关于按键防抖的原理，请参考本实验手册的“5.1 实验3.2 相关说明”部分）；
5. **设计具有开放性**，图2仅仅为参考的一种实现效果，**鼓励设计实现自己的输入输出控制逻辑和显示效果**。
6. 数码管、按键、开关的管脚和控制方式等信息请参考本文件“5.1 实验3.2 相关说明”以及EGo1用户手册和EGO1电路原理图。

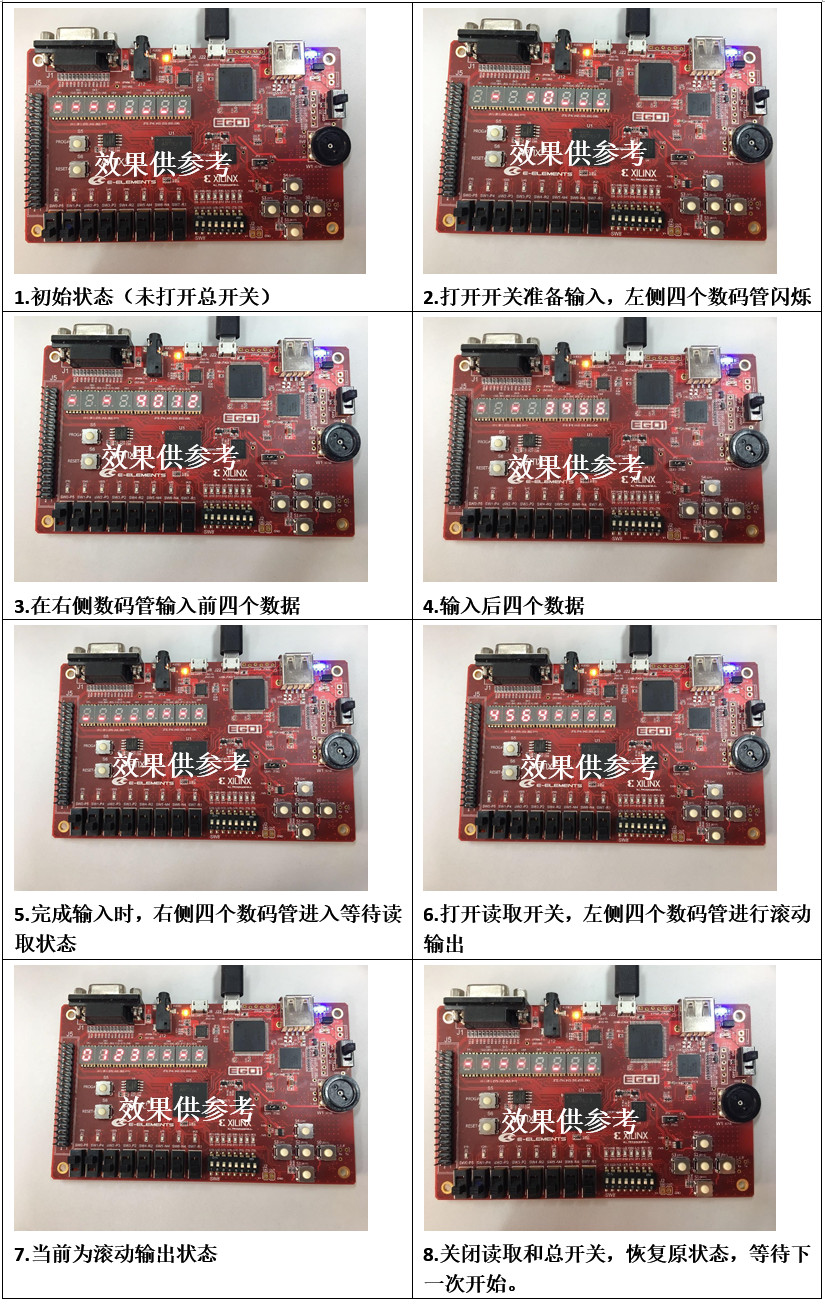
****

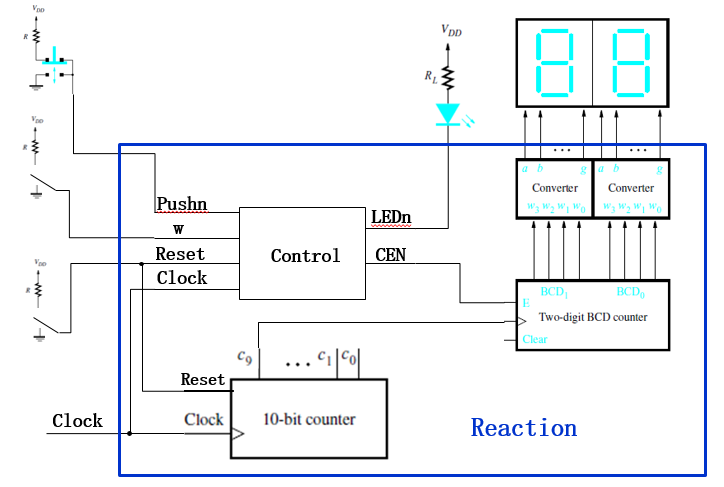
图2 按键输入及显示例

**说明：对实验3.2，验收时间靠前（前15%）的同学具有优先加分机会。**

## 实验要求

1. 在实验报告中提交系统级设计模块图、设计代码、激励程序（不必须包含所有模块的）、仿真波形结果截图（与激励配套）、板级实测验证结果照片。

其中，系统级设计模块图要求给出整个系统的数据输出信号，系统内各个子模块的输入输出信号和模块间的连接关系（图3仅做示意，不需要画出数码管/LED灯/按键等，给出其信号名及位宽即可）。



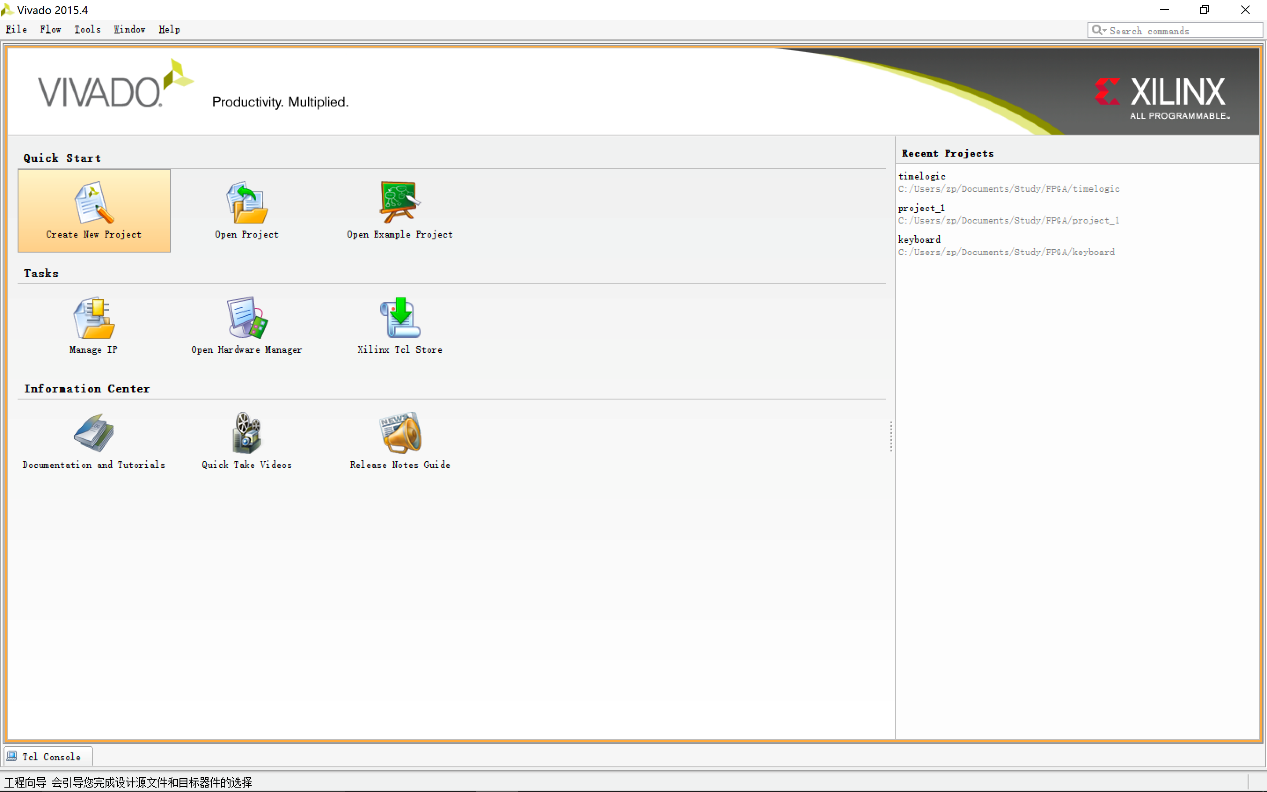
**图3 系统级设计模块图（示意图）**

1. 提交实验报告和所有源程序文件的压缩包。

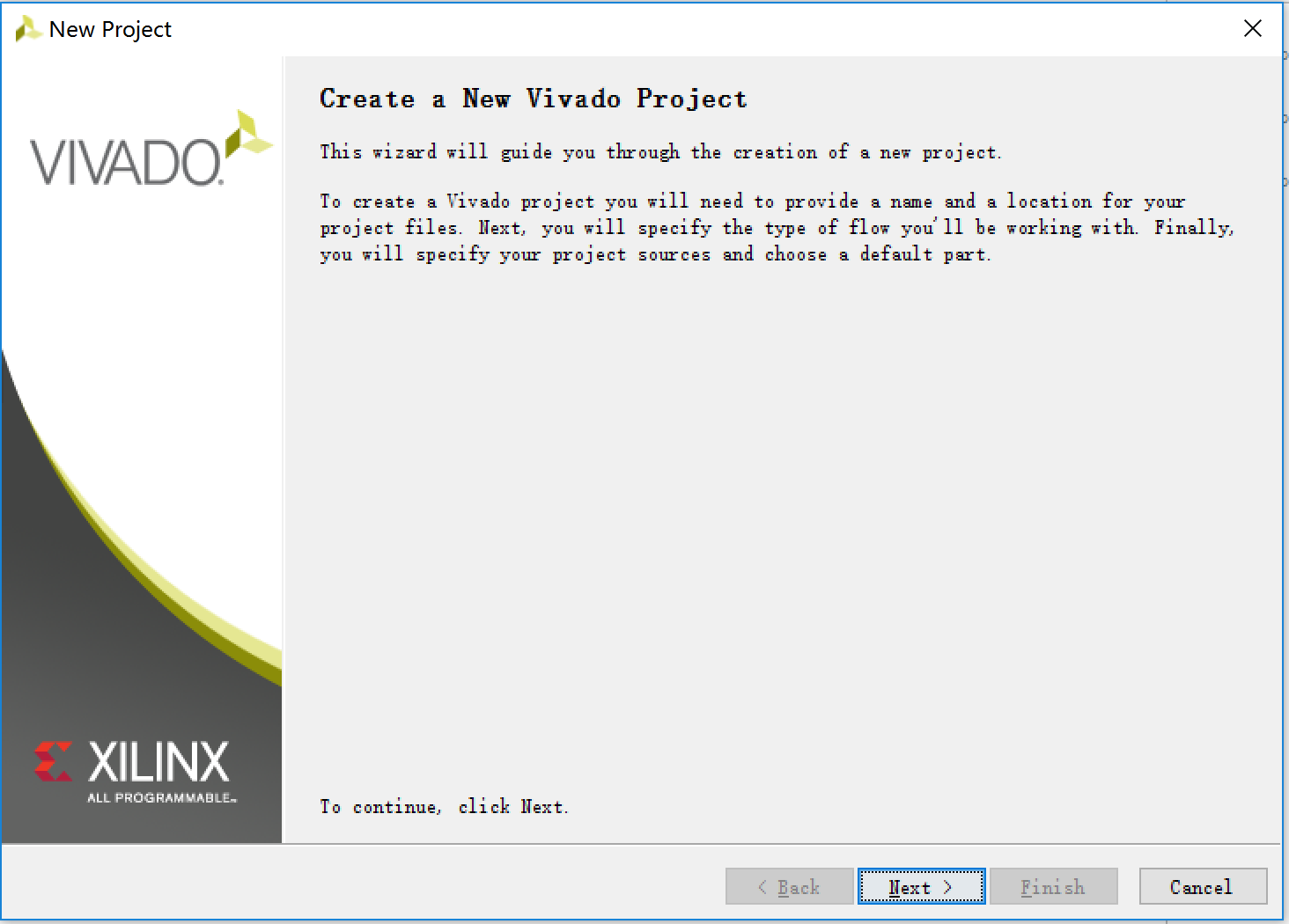
## 实验步骤——在七段数码管上滚动显示学号

#### 4.1创建工程

1. **打开Vivado，选择“Create New Project”，创建一个新的工程。**

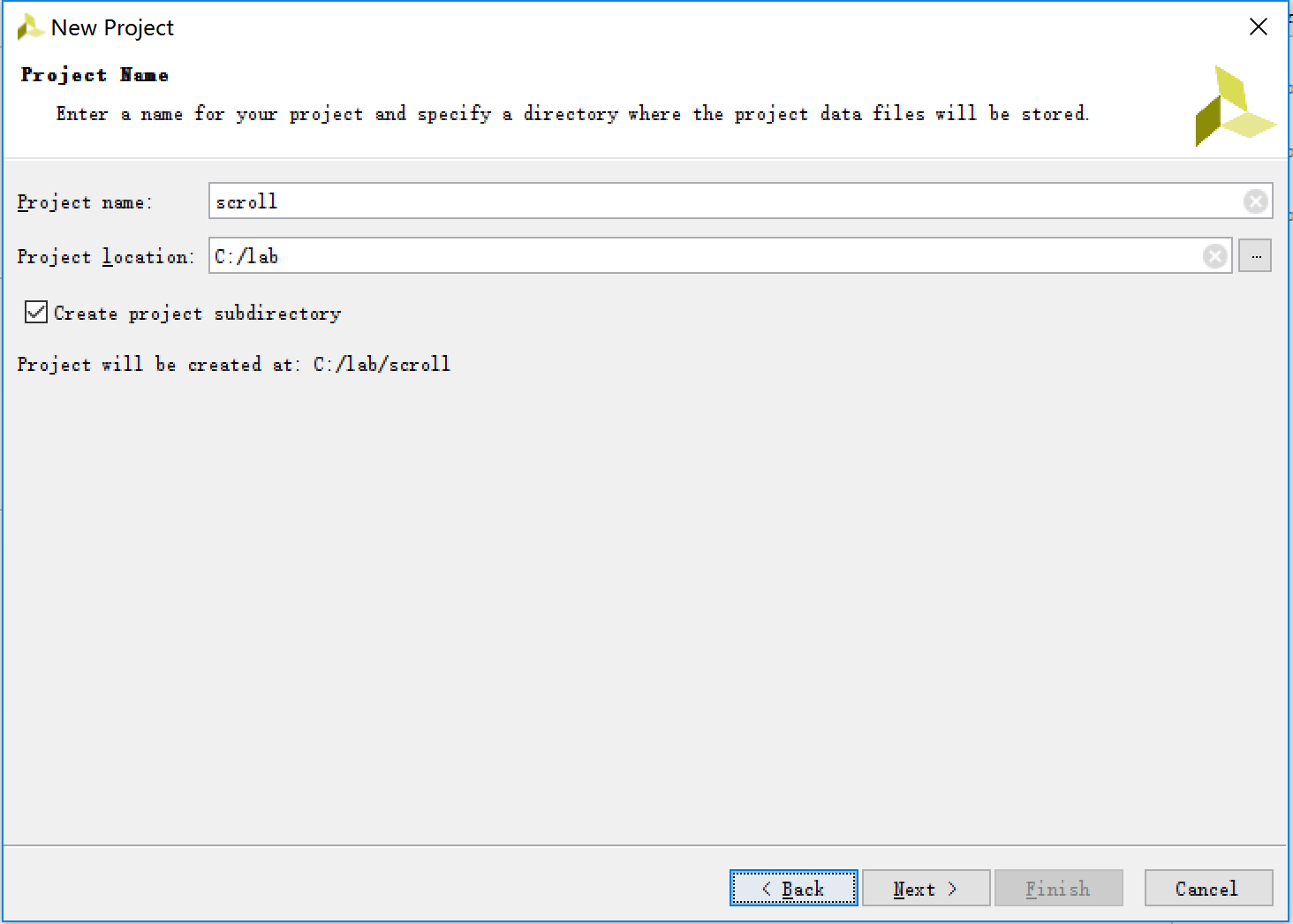


1. **进入新建工程向导。**

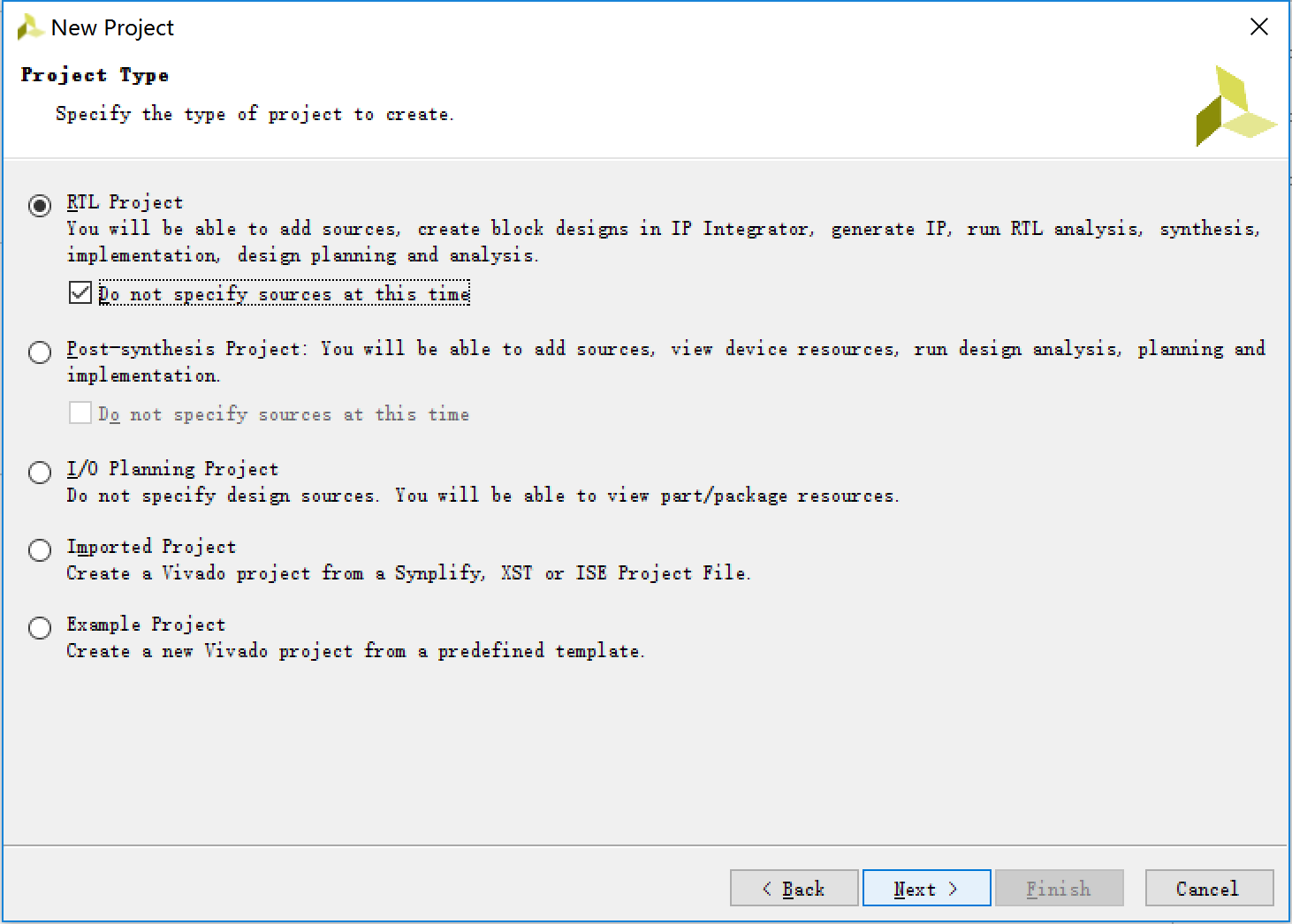


1. **点击“Next”， 输入工程名并指定工程所在的目录，确认勾选 “Create project subdirectory”。**

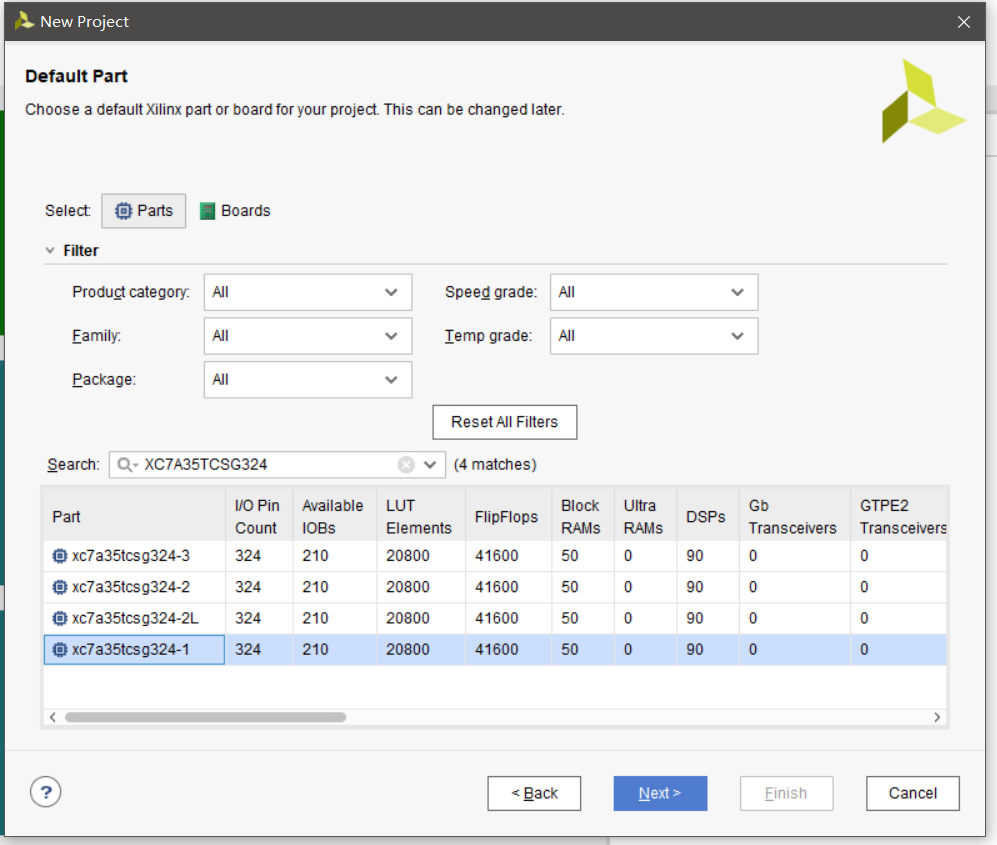
**注意工程名及工程所在的路径中只能包括数字、字母及下划线，不允许出现空格、汉字以及特殊字符等。这里将工程名命名为 “scroll”，工程目录选择 “c：/lab”。**



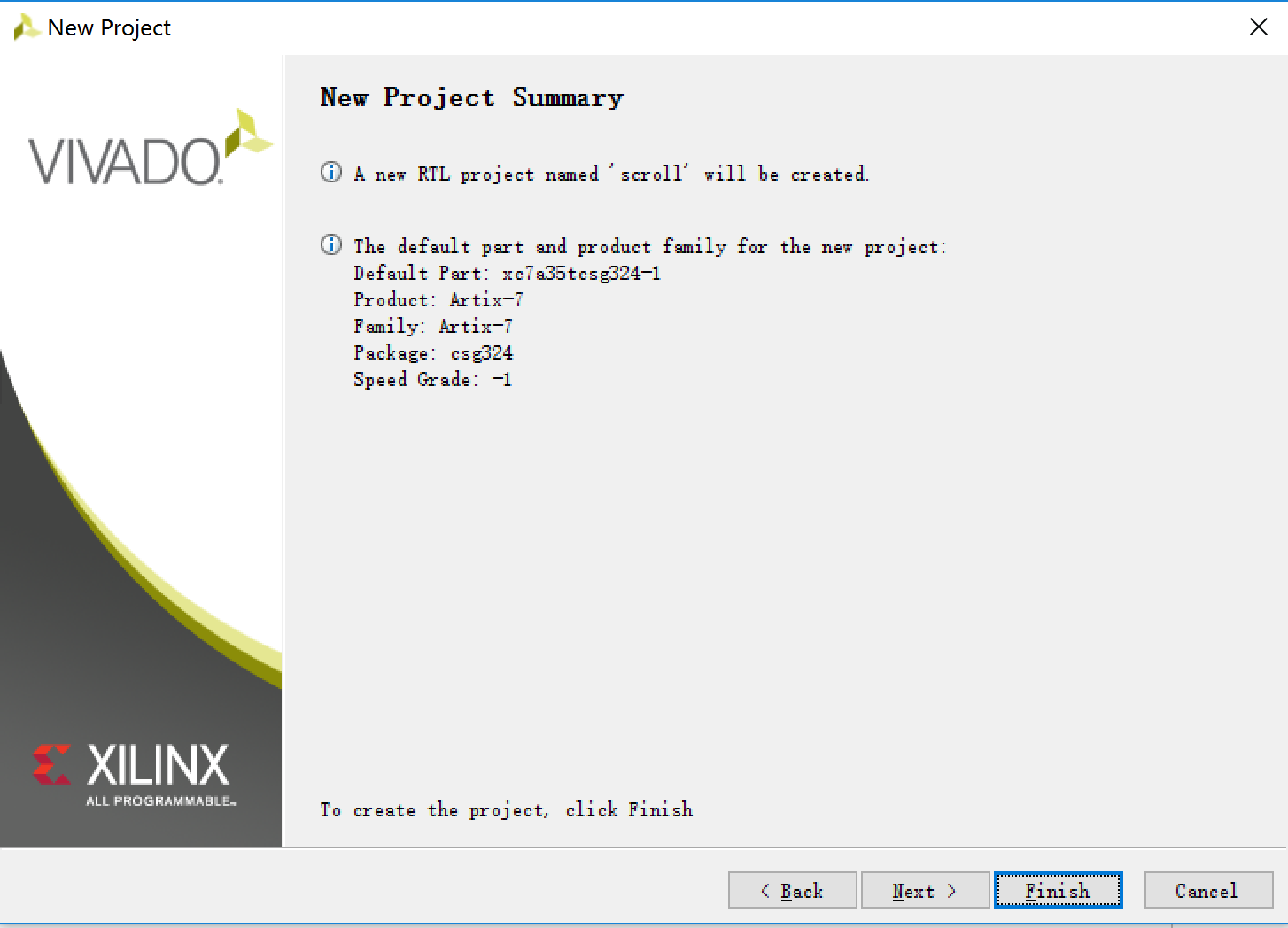
**4. 点击 “Next”, 指定创建的工程类型，选择 “RTL Project”，勾选“Do not specify sources at this time”。**



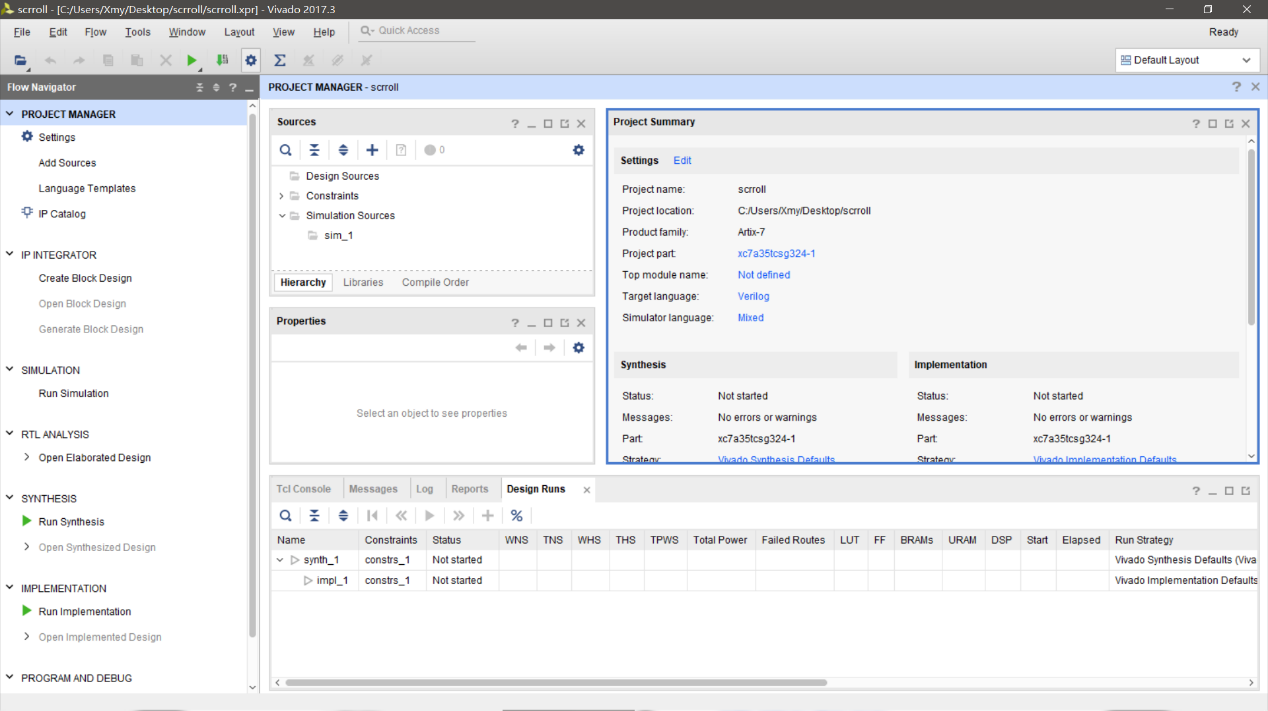
1. **点击“next”。**
2. **进入器件选择界面，搜索芯片型号为XC7A35TCSG324‐1。**
3. **在符合条件的器件中选中板卡对应的芯片。**



1. **点击“Next”，进入Summary界面查看所创建工程的相关信息。**

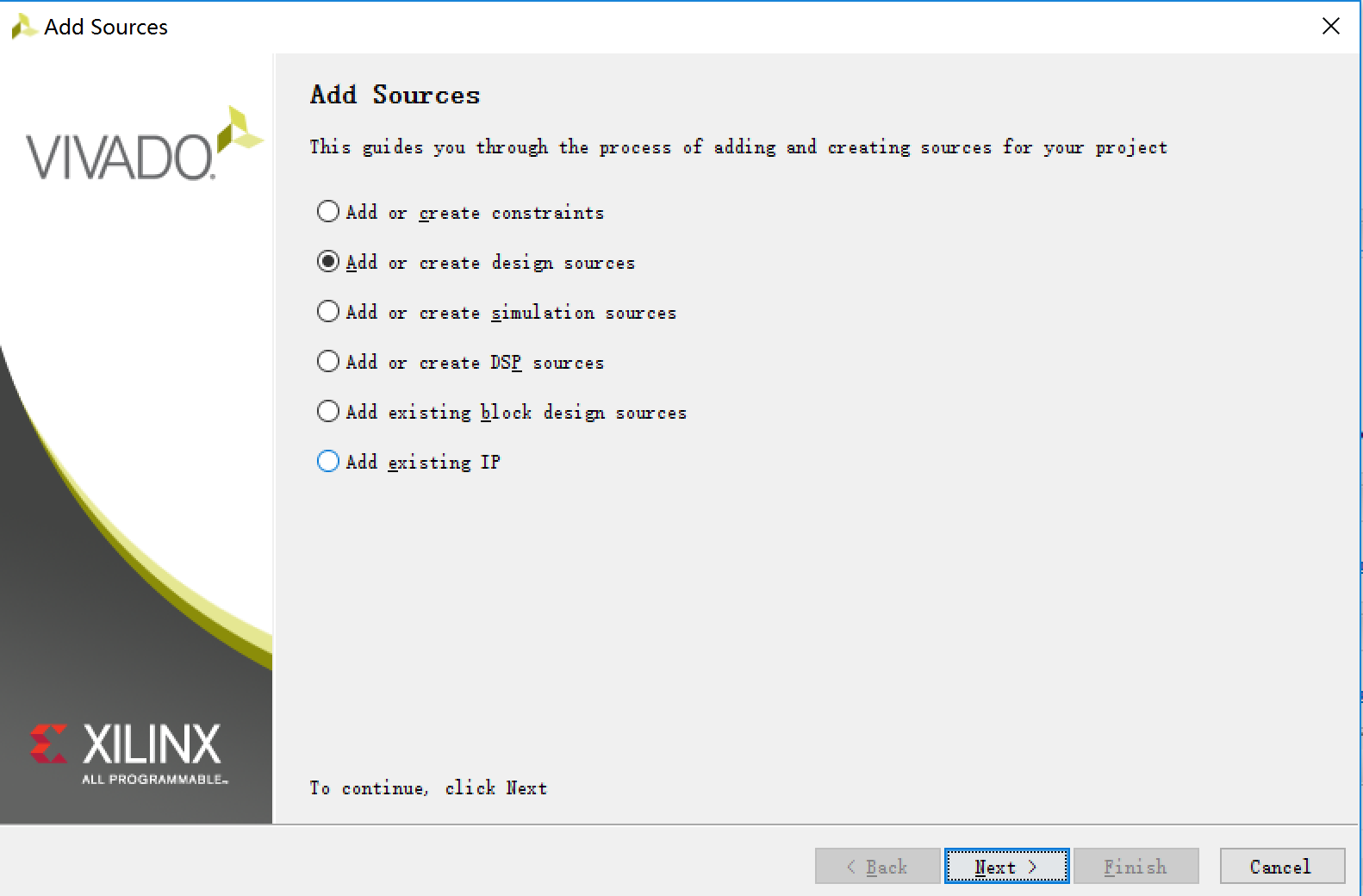


1. **点击“finish”打开创建的工程。**

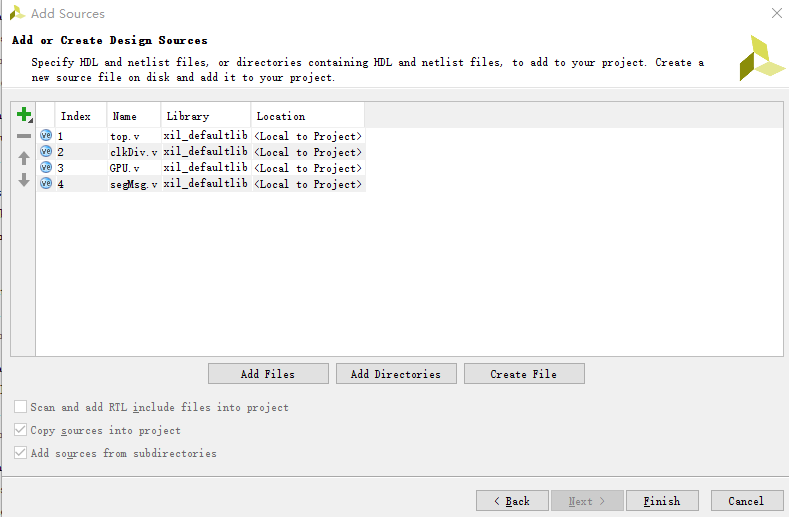


#### 4.2创建源文件

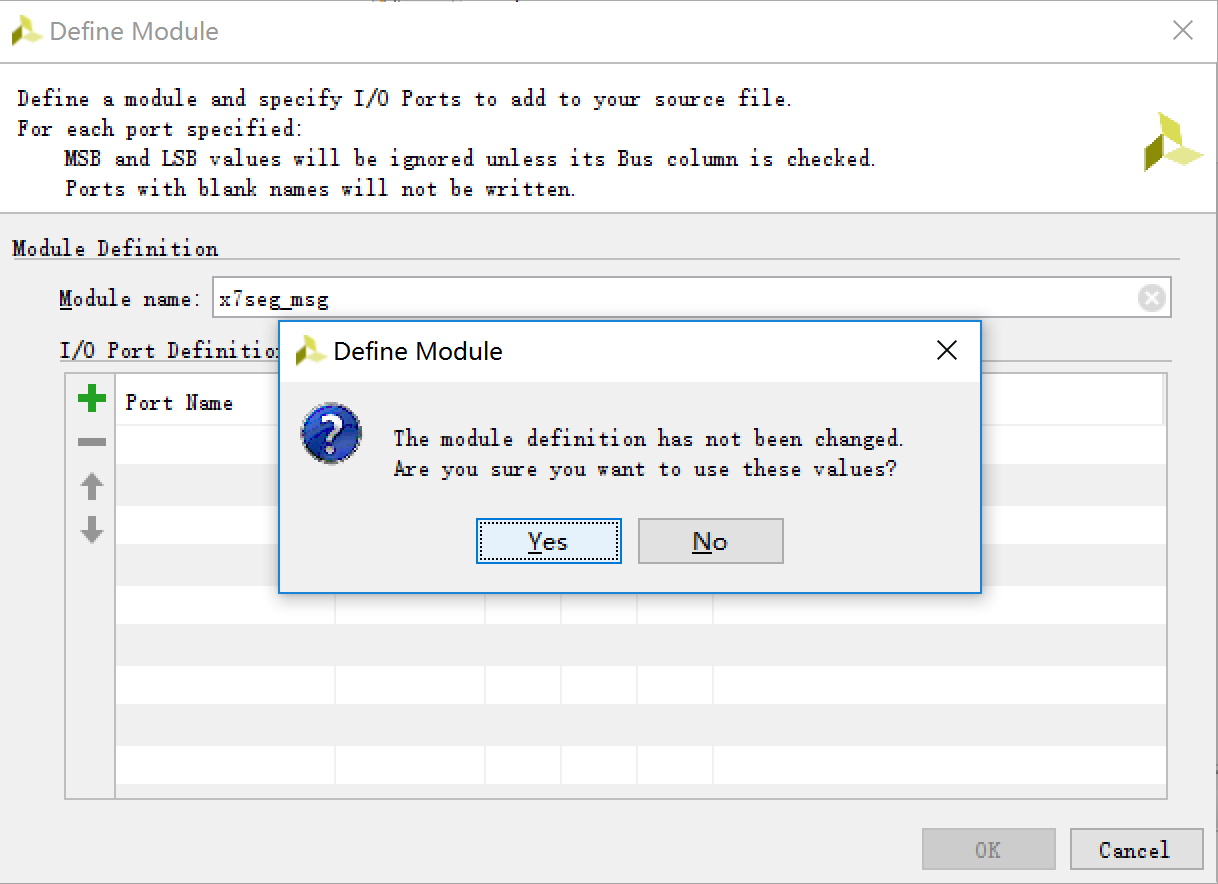
1. **在左侧“Flow Navigator”栏中的“Project Manager”下点击“Add Sources”, 在弹出的窗口中选择 “Add or create design sources”, 点击 “Next”。**



1. **选择 “Create File”。输入四个文件名。**

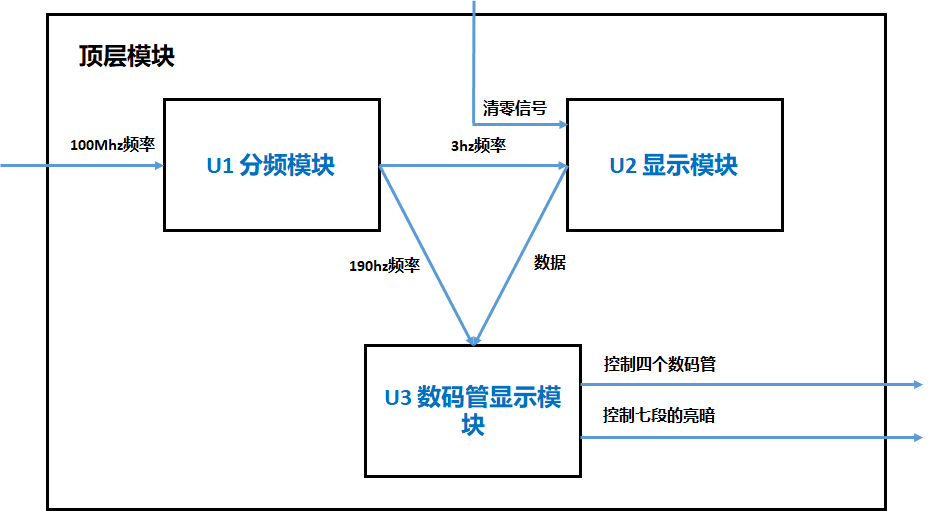


1. **分别点击“OK”，然后点击“Finish”。**
2. **再点“OK”。选择“Yes”。**



1. **在 Sources 窗口可查看添加的源文件。**

* **模块图：**



**15.在Design Sources 里完成以下各个模块**

**各模块代码如下：**

* 顶层模块（top）:

|  |
| --- |
| top1 |

* 分频模块（clkDiv）

|  |
| --- |
| 2 |

* 显示处理模块（GPU）

|  |
| --- |
| C:\Users\asus\AppData\Local\Microsoft\Windows\INetCache\Content.Word\1.jpg  negedge |

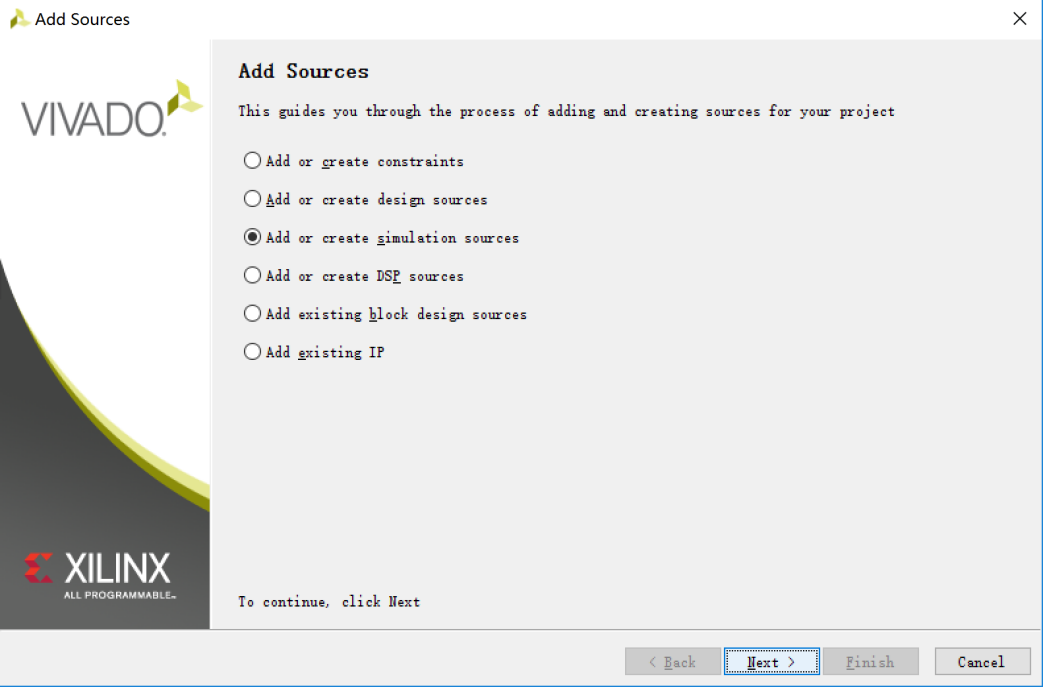
* 数码管显示模块（segMsg）

|  |
| --- |
| C:\Users\asus\AppData\Local\Microsoft\Windows\INetCache\Content.Word\seg.jpg |

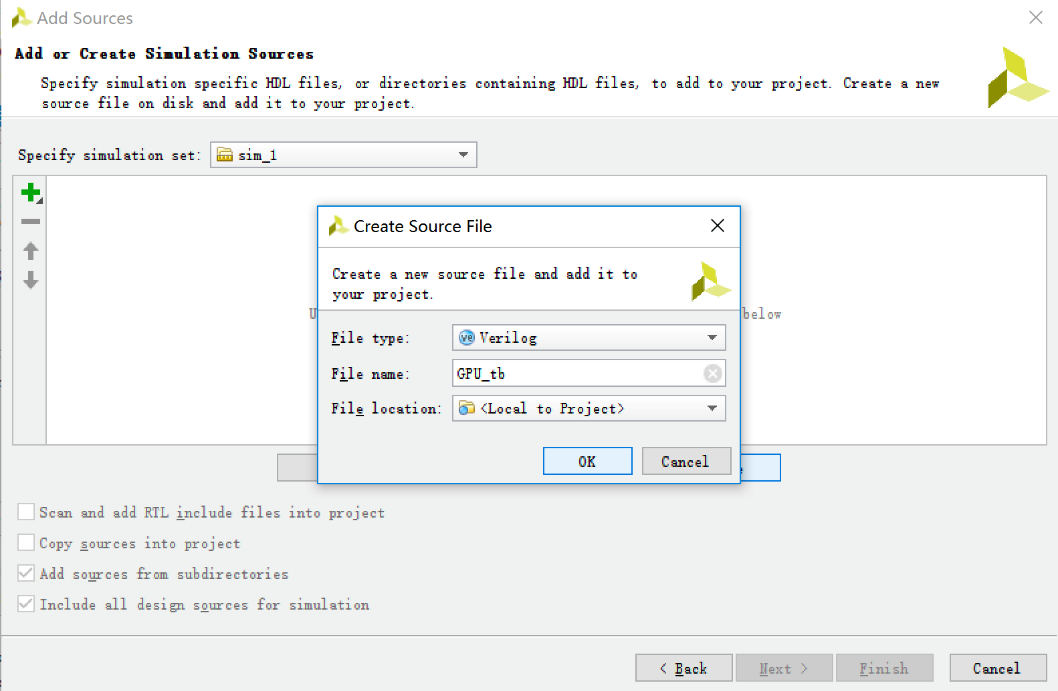
#### 4.3创建仿真激励文件

**注：模块仿真可以根据需要自由选择，可以针对顶层模块仿真，关心整个工程的输入输出对应关系。也可以针对某一个你需要测试的模块进行仿真。**

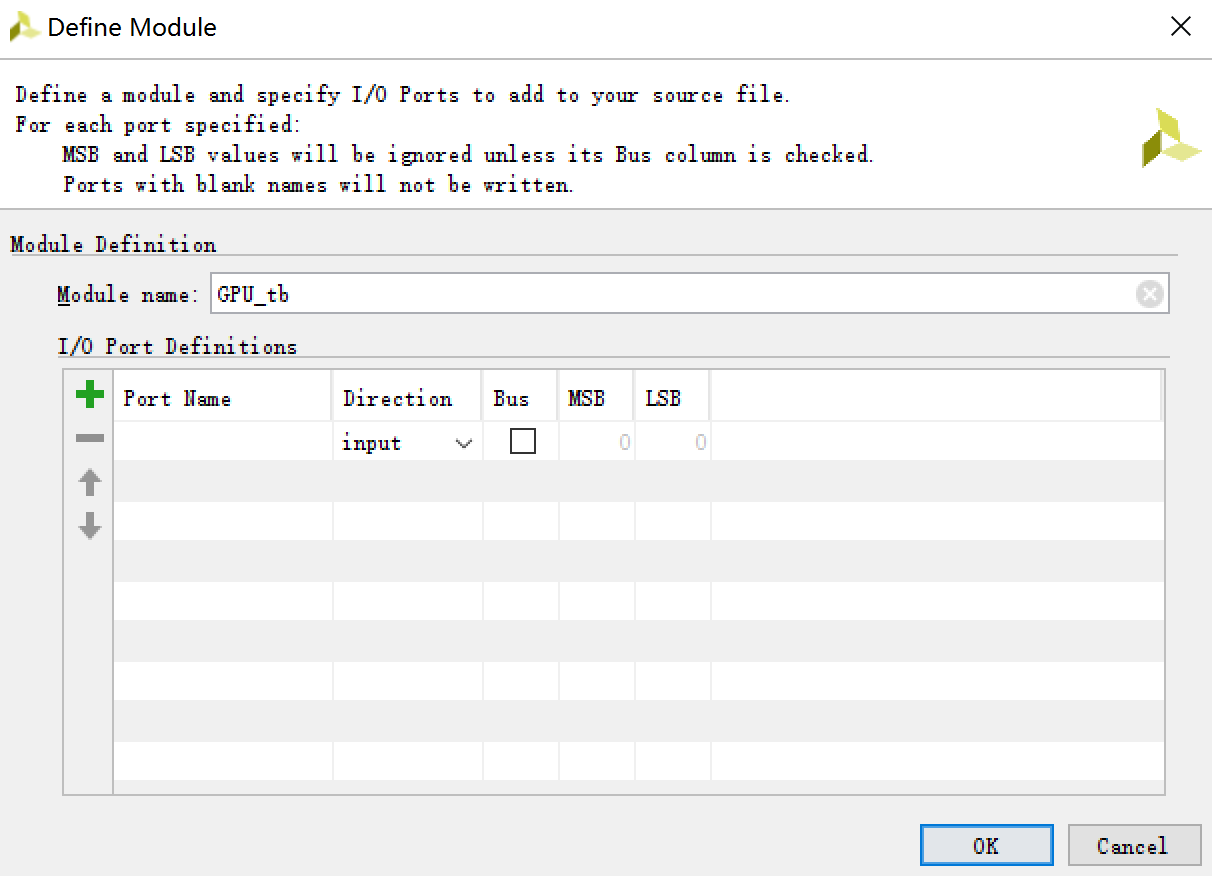
**16.在左侧“Flow Navigator”栏中的“Project Manager”下点击“Add Sources”, 在弹出的窗口中选择 “Add or create simulation sources”, 点击 “Next”。**



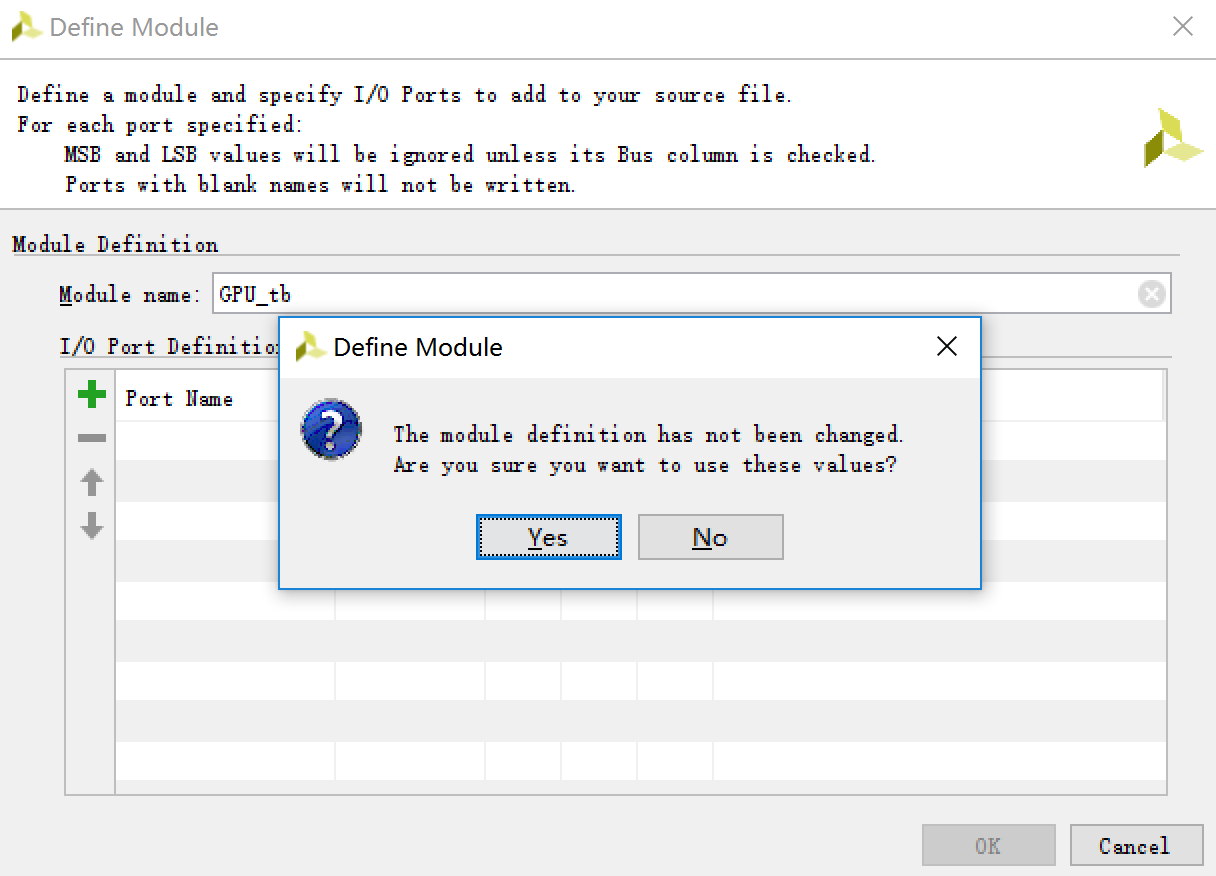
**17.点击Create File**



**18.点击“Finish”再点击“OK”**

****

**19.点击“YES”**

****

**20.在 Sources 窗口展开Simulation Sources下的sim\_1可以看到添加的激励文件。**

**21.在Simulation Sources 里完成以下激励模块编辑**

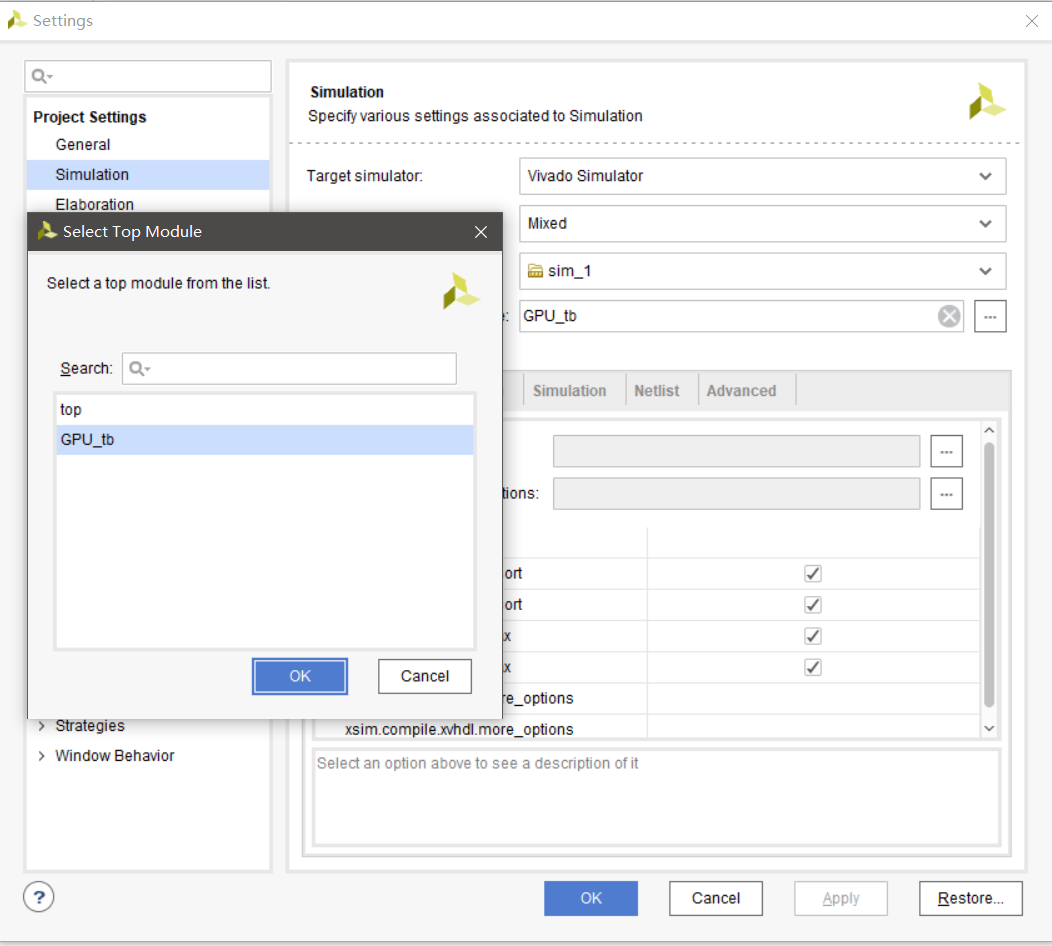
**模块代码如下：**

* 针对GPU模块的仿真模块（GPU\_tb）

|  |
| --- |
| **tb1** |

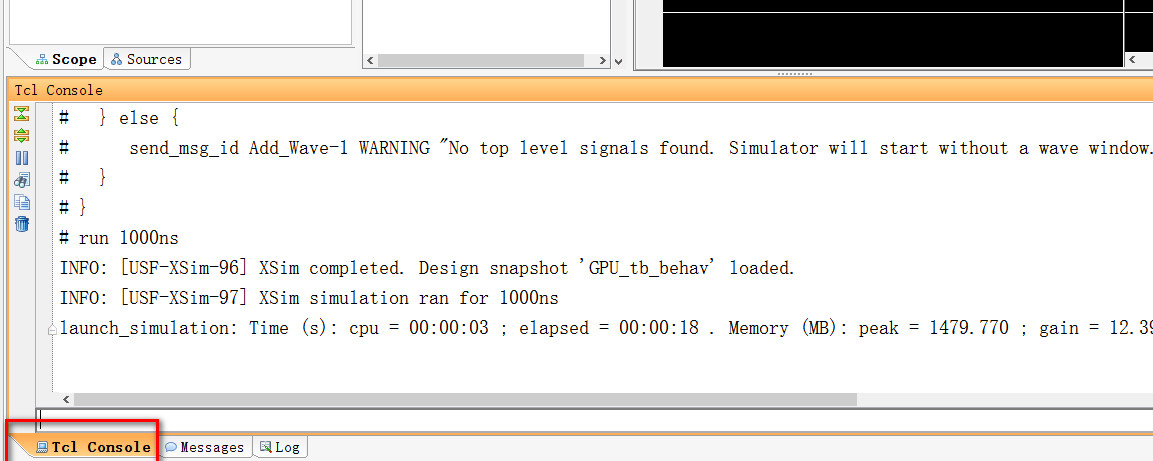
**22.在Flow Navigator 中找到Simulation Settings 点击打开Project Settings**

**23.将Simulation top module name 改为GPU\_tb，点击“OK”**

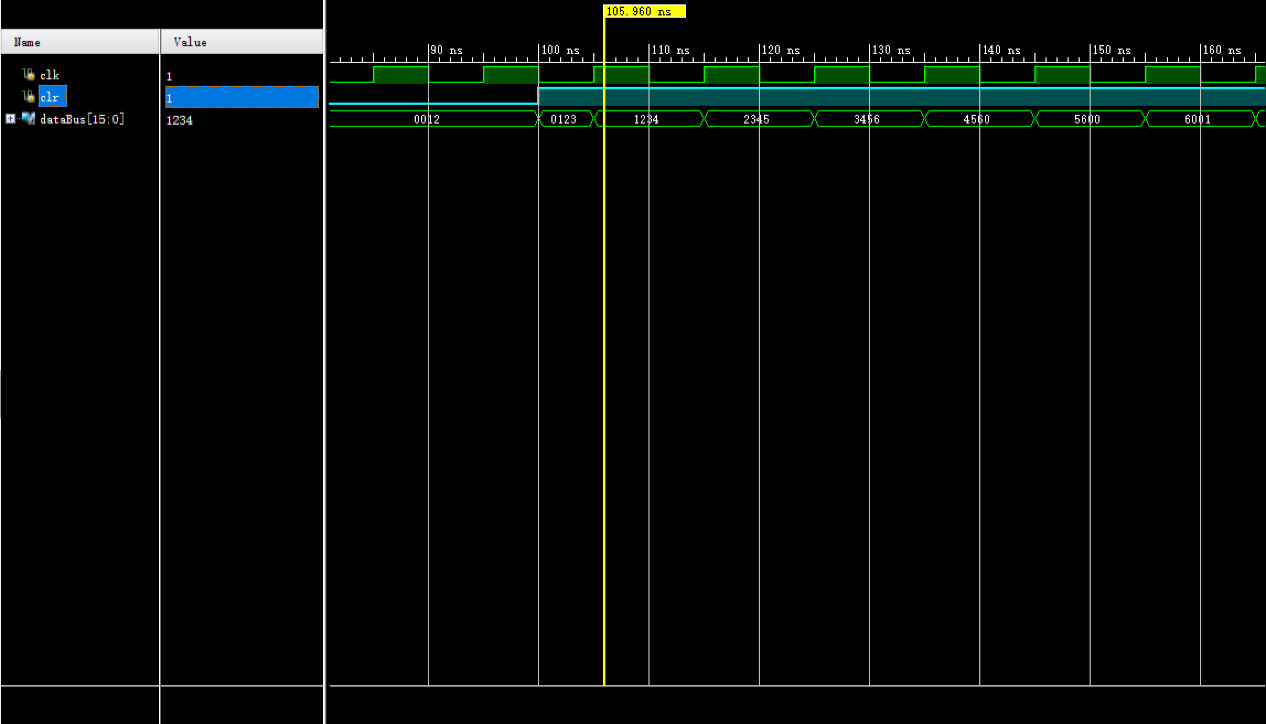


#### 4.4行为仿真

**24. 在左侧“Flow Navigator”栏中的的“Simulation”下点击“Run Simulation”->“Run Behavior Simulation”。如果出现错误可以在软件下方的“Tcl Console”窗口中查看错误详情。**

****

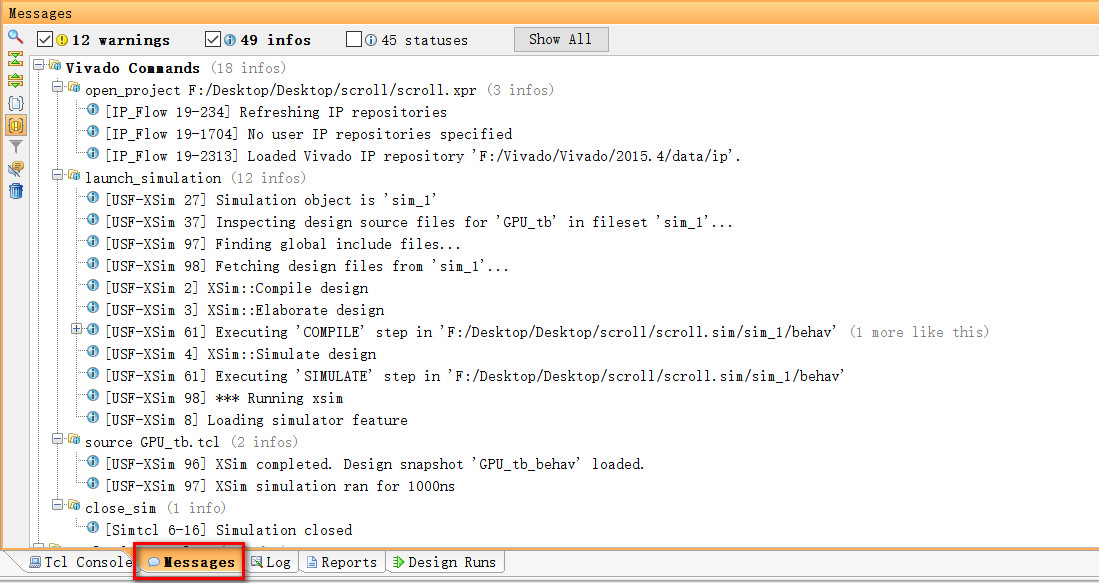
**25.若没有错误，点击波形窗口,（Ctrl+鼠标滑轮）调整波形到合适的尺度，观察波形。**



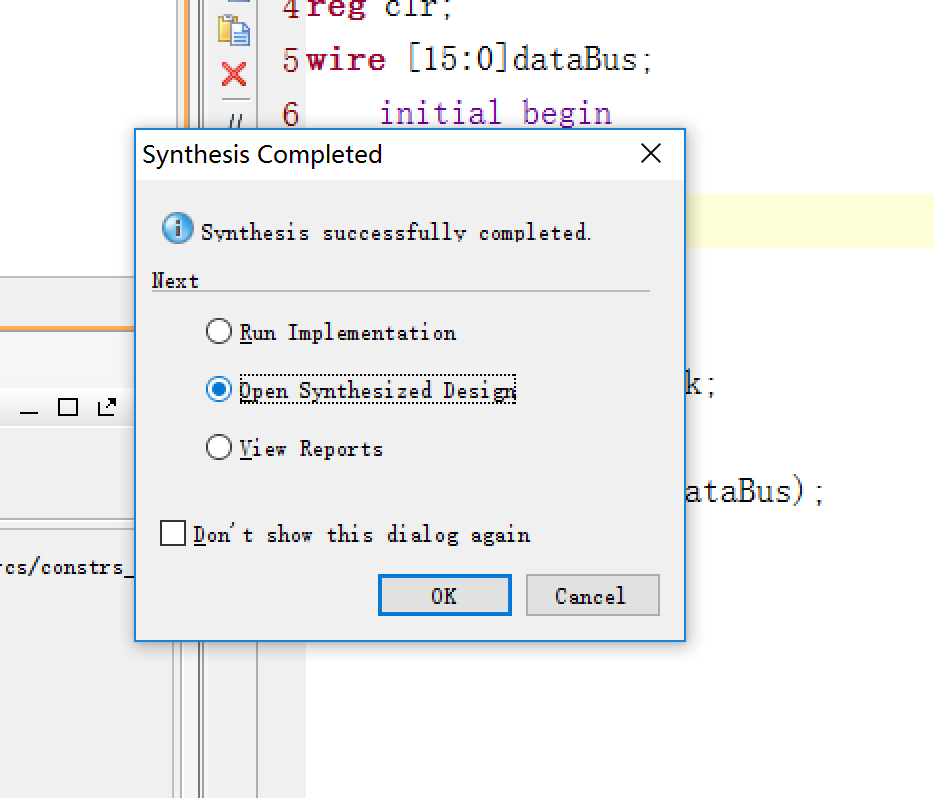
可以看到数据总线在每个时钟周期上升沿进行数据移位

#### 4.5综合、实现、下载

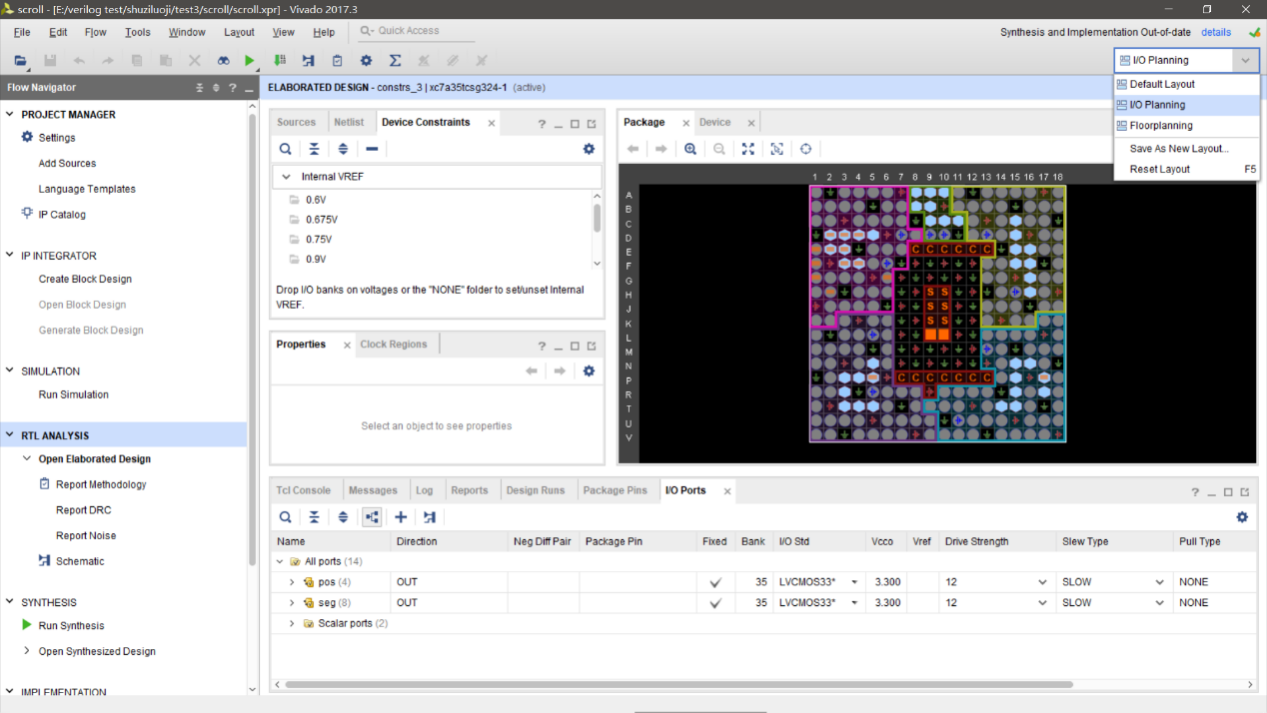
**25. 在左侧“Flow Navigator”栏中的的“Synthesis”下点击“Run Synthesis”。如果出现错误可以在软件下方的“Messages”窗口中查看错误。**

****

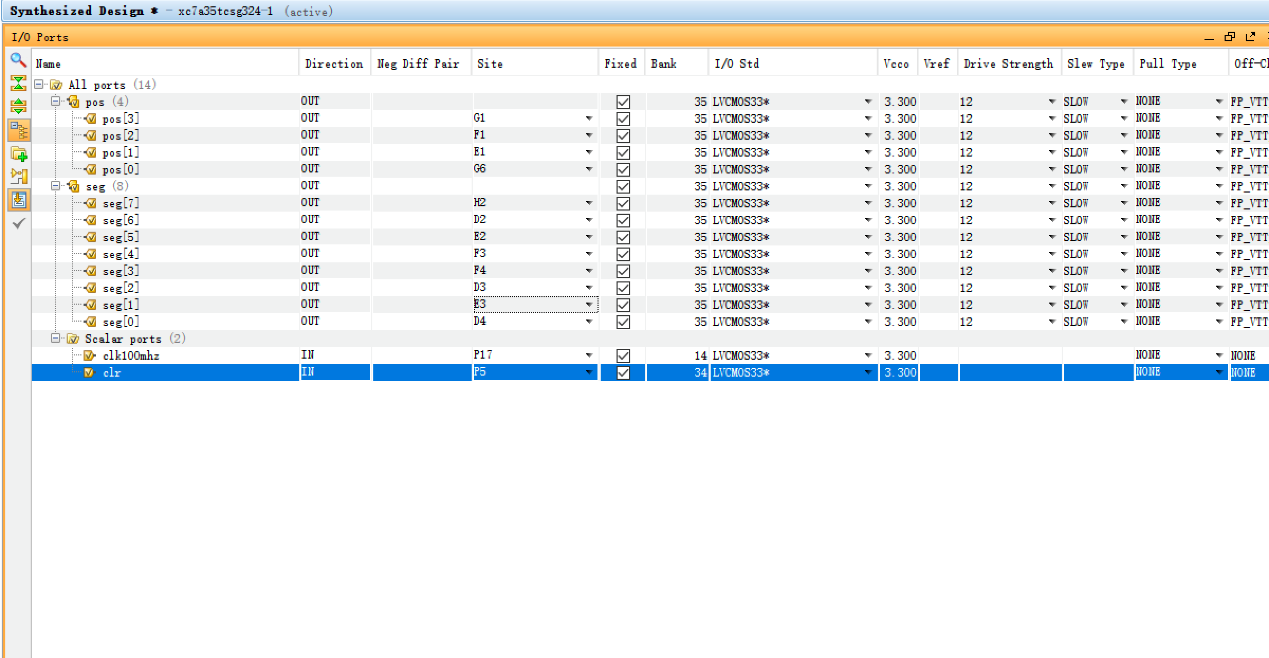
**26.若没有错误，则可以直接打开综合设计，点击“Open Synthesized Design”**

****

**26. 将窗口切换到I/O Planning**

****

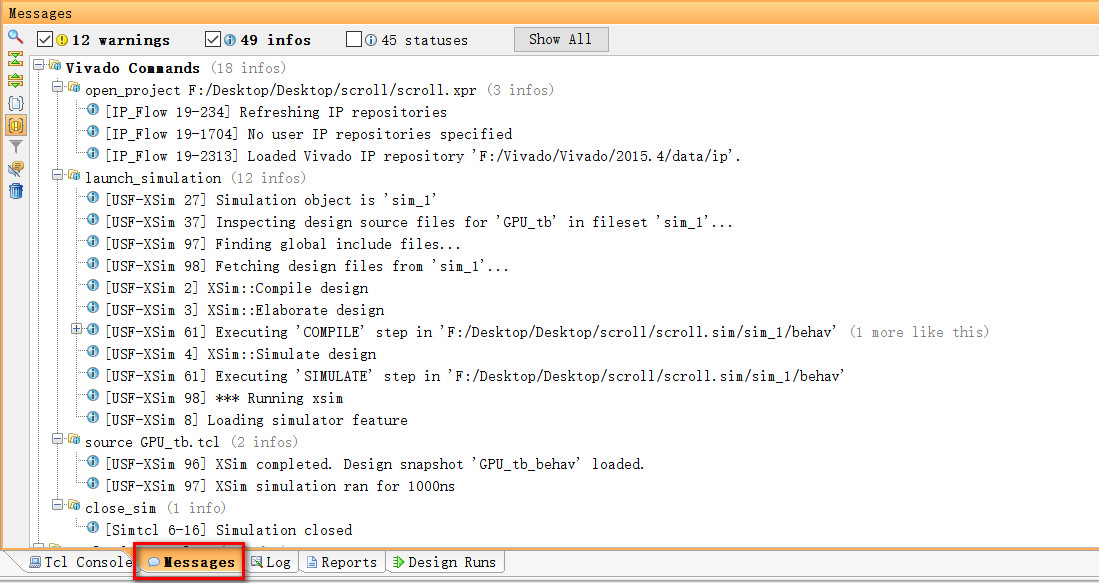
**27.进行管脚分配和电压设置**

****

**28.Ctrl+S保存，给管脚文件命名**

****

**29. 在左侧“Flow Navigator”栏中的“Implementation”下点击“Run Implementation”。如果出现错误可以在软件下方的“Messages”窗口中查看错误。**

****

**30.如果没有错误就直接选择“Generate Bitstream”,或者在左侧“Flow Navigator”栏中的“Program and Debug”下点击“Generate Bitstream”**

**31.最后在左侧“Flow Navigator”栏中的“Program and Debug” 点击“Open Hardware Manager”,将程序下载到FPGA上，将开关D1拨上去，观察效果。**

## 实验参考

#### 5.1 实验3.2的相关说明

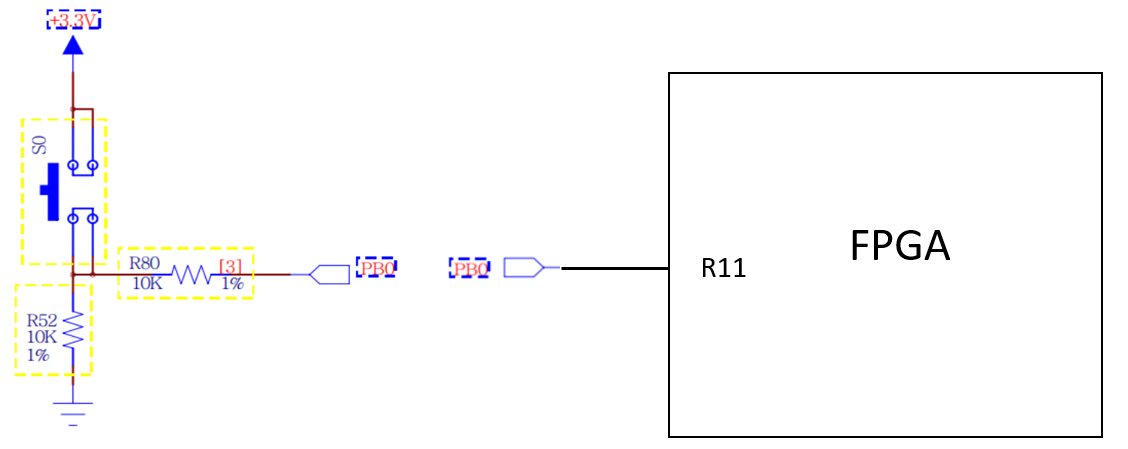
1）EG01按键说明

EGO1具有2个专用按键和5个通用按键。本实验中主要用到5个通用按键，在PCB板上的名称是S0，S1，S2，S3，S4。



下图以S0为例显示了通用按键的电路原理图，图中按键输出信号PB0连接到了PFGA的R11管脚（提示：电路原理图中，通常以同一个名称来命名多个信号线，表示这几个信号线都是同一个根信号线，这样使得复杂的原理图变简洁）。从该原理图可知：

* 当S0按键没有按下时，PB0低电平；
* 当S0被按下时，PB0为高电平；

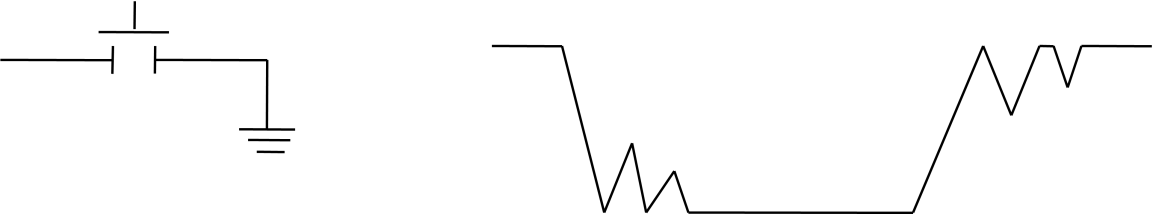


S0，S1，S2，S3，S4的管脚约束如下：

|  |  |
| --- | --- |
| 按键名称 | FPGA管脚 |
| S0 | R11 |
| S1 | R17 |
| S2 | R15 |
| S3 | V1 |
| S4 | U4 |

2）按键防抖说明

为了保证键每闭合一次 FPGA 仅作一次处理，必须去除键按下时和释放时的抖动。开发板使用的按键是触点式的，由于按键是机械触点，当机械触点断开、闭合时，会有抖动。



按键抖动示意图

按键抖动示意图中的这种抖动对于人来说是感觉不到的，但对于FPGA或者处理处理器来说，其运行速度的是在微秒级甚至纳秒级，而机械抖动的时间至少是毫秒级，因此这种抖动是一个“漫长”的时间。

对于按键存在的抖动，如果在抖动过程中高低电平的状态没发生变化，则这个抖动我们是不需要考虑的。但是，如果在抖动过程中高低电平状态发生了变化甚至是频繁变化，则这个抖动我们就消除，下面所提到的抖动就是这种类型的抖动。 为使 FPGA 能正确的读出按键的状态，对每一次按键只作一次响应，就必须考虑如何去除抖动，常用的去抖动的方法有两种：硬件方法和软件方法。

FPGA 设计中，常用软件法去抖，因此对于硬件方法我们在此不作介绍。 软件法去抖其实很简单，按键初始状态为低电平，当 FPGA 获得键值为 1 的信息后，不是立即认定按键已被按下，而是延时 5ms 或更长一些时间后再次检测按键，如果仍为低，说明按键的确按下了，这实际上是避开了按键按下时的抖动时间。而在检测到按键释放后再延时 5ms，消除后沿的抖动，然后再对键值处理。当然，实际应用中，按键的质量也是千差万别，要根据按键的不同，来设定这个延时时间，通常这个延时时间不会太短，一般设为 5~20ms。具体做法是，将按键信息延时 3 次取样 3 次，每次延迟取样间隔约为 5ms，当这 3 个取样值都一样时，说明抖动已消失，如果 3 个取样值不一样，说明抖动存在，直至这 3 个取样值一样时，才认为按键稳定。将这 3 个取样值与运算后得到的信号作为按键的状态，这个按键状态可作为稳定的按键输入，参与到后续对按键的处理操作。

3）拨码开关说明

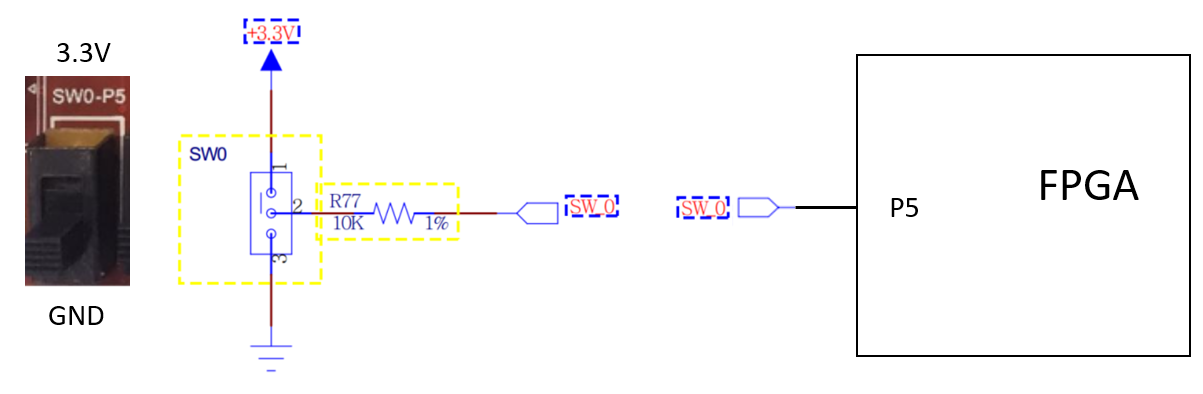
EGO1上有8个拨码开关（SW0~SW7）



SW0~ SW7

以SW0为例，拨码开关的原理图如下图所示，开关输出管脚SW\_0与FPGA的P5管脚相连。开关具有三个状态：

* 开关拨到3.3V侧，SW\_0为高电平；
* 开关拨到GND侧，SW\_0为低电平；
* 开关拨到中间，SW\_0位悬空，建议不要用悬空。

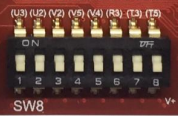


拨码开关管脚约束



4）DIP开关说明

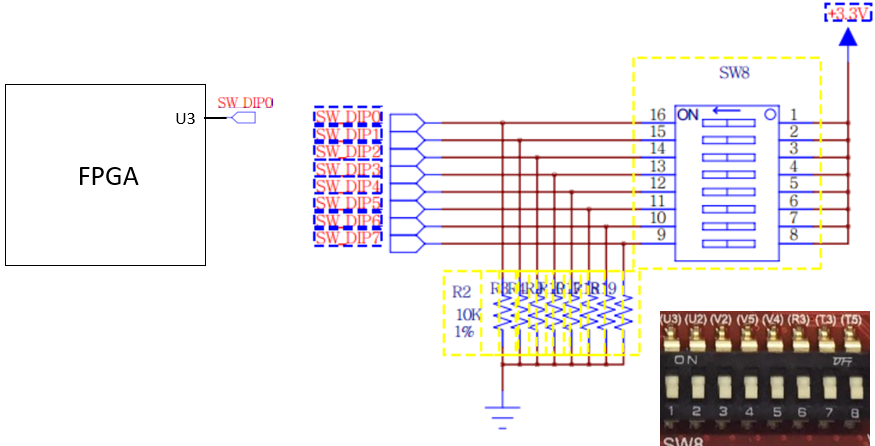
1个8位的DIP开关(SW8,包含8个小开关)，如下图所示。



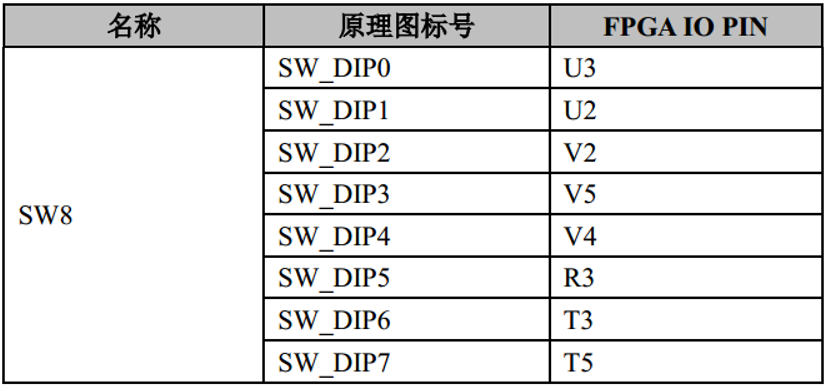
DIP开关 SW8

DIP拨码开关的原理图如下图所示，开关具有两种状态。以SW\_DIP0为例

* 开关拨到“ON” 侧，SW\_DIP0为高电平；
* 开关拨到“1” 侧，SW\_DIP0为低电平；



DIP开关管脚约束



5）七段数码管说明

EGO1上共有8个七段数码管，DK1~DK8。

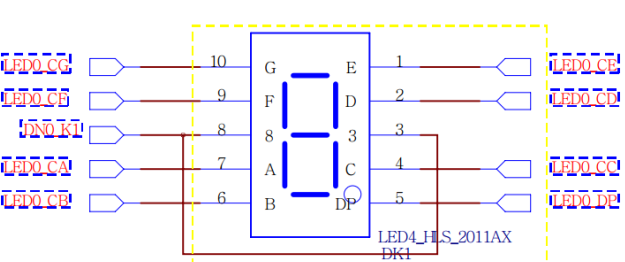


如下图以DK1为例说明**单个数码管**的控制，每个数码管有10个管脚（A~G,DP,8，3）。

段选信号（高电平有效）：A~G控制七段显示和DP控制小数点显示；

控制信号：8和3位驱动管脚（信号线名称为DN0\_K1）。

其中，DK1 的 A~G和 DP分别通过信号线LED0\_CA~LED0\_CG, LED0\_DP，连接到FPGA的B4，A4，A3，B1，A1，B3，B2和D5管脚。



**数码管的使能**：数码管为共阴极数码管，即控制信号（DN0\_K1）输入低电平使能数码管。如下图所示，控制信号由三极管驱动，通过信号线LED\_BIT1连接到FPGA的G2管脚。因此，**G2高电平使能DK1**。



数码管使能控制

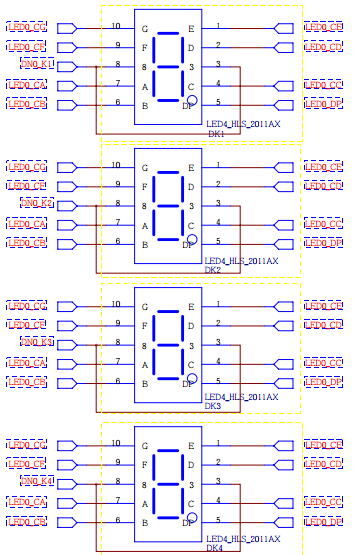
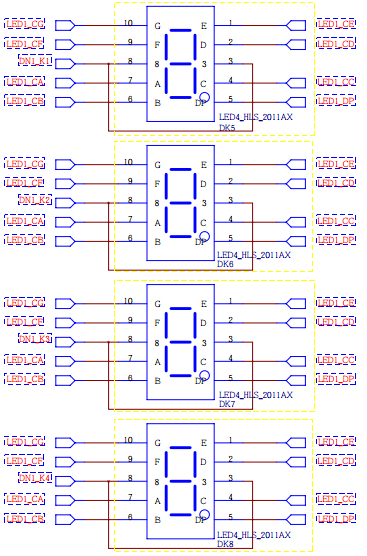
**数码管的分组管脚复用**

如下面原理图所示，8个数码管分为两组：

第一组：DK1~DK4，如图可知所有DK1~DK4的段选信号（A~G,DP）管脚复用一组信号线（LED0\_CA~LED0\_CG,LED0\_DP）。例如：DK1~DK4的A管脚都连到了LED0\_CA。 而 DK1~DK4的使能信号是单独控制的。如上图所示，DK1~DK4的使能管脚分别连到了DN0\_K1~DN0\_K4，链接到FPGA的使能控制信号分别是LED\_BIT1~LED\_BIT4。

第二组：DK5~DK8，也是段选信号复用（LED1\_CA~LED1\_CG,LED1\_DP），使能信号单独控制LED\_BIT5~LED\_BIT8。

更详细的原理图参见EGO1电路原理图。

.

数码管脚约束如下：

