RISC-V 处理器阿里平头哥玄铁 E902 与 wujian100 的 FPGA 实现

目录

实验准备

1. 开发板

2. 参考资料和源代码下载

   邸老师主页：http://www.dizhixiong.cn/class5/

3. 工具：Vivado，CDK（百度查找下载方法）

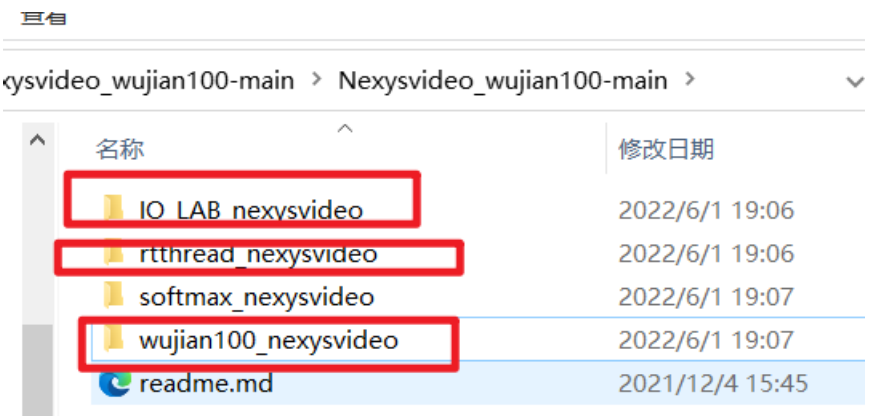
   Vivado：可以下载最新版，也可以下载 2018 版，占内存小；

   CDK：

4. 下载器：

- FPGA USB Cable（连接 Vivado 和板子，将 wujian100 下载到板子上）
- Clink （调试器，用来连接 CDK 和开发板，下载 LED、SD 卡等测试程序到开发板)
- USB 转串口（用来读串口数据）

5. 邸老师主页的源代码相关文件有：



实验一：Wujian100 的仿真

实验目标

Wujian100_open 是阿里平头哥在 Github 上开源了 RISC-V 项目。本实验主要完成在 Linux 环境下实现 Wujian100 的仿真。

实验步骤

1.下载官方代码

无剑的仿真需要 Linux 环境，这里使用的是 MobaXterm 的 SSH 终端连接 Ubuntu 的工作站。

```
eda@abc-MW51-HP0-00:~$ cd Desktop
eda@abc-MW51-HP0-00:~/Desktop$
```

建一个文件夹，作该为项目的文件夹：mkdir wujian

切换工作目录到该文件夹下：cd wujian

将 Wujian100 的代码从 github 上下载到该文件夹下: git clone http://github.com/T-head-Semi/wujian100_open.git

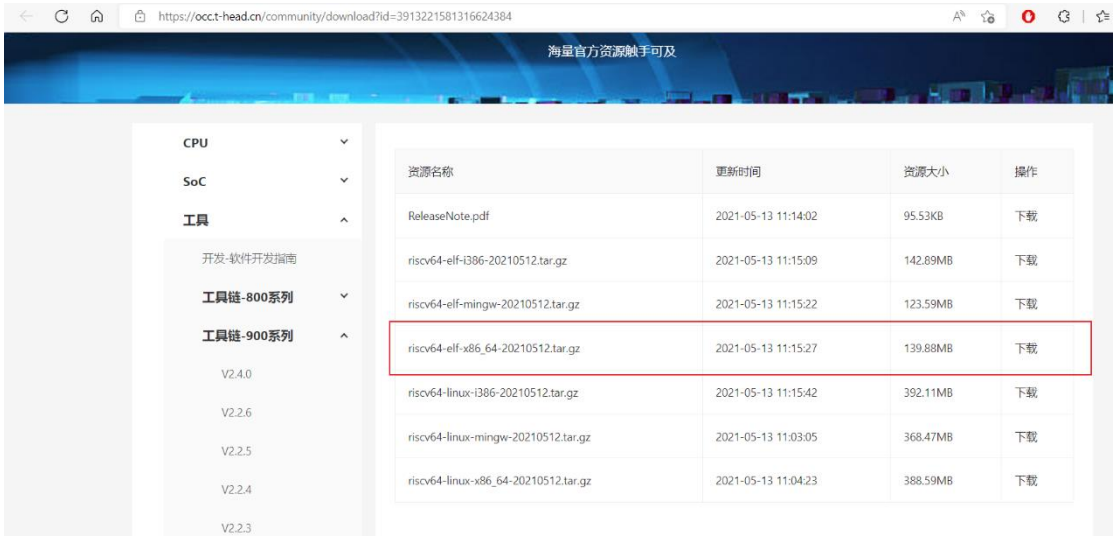
```
eda@abc-MW51-HP0-00:~/Desktop$ mkdir wujian
eda@abc-MW51-HP0-00:~/Desktop$ cd wujian
eda@abc-MW51-HP0-00:~/Desktop/wujian$ git clone http://github.com/T-head-Semi/wujian100_open.git
```

```
eda@abc-MW51-HP0-00:~/Desktop/wujian$ git clone http://github.com/T-head-Semi/wujian100_open.git
正克隆到 'wujian100_open'...
warning: 重定向到 https://github.com/T-head-Semi/wujian100_open.git/
remote: Enumerating objects: 928, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 928 (delta 6), reused 2 (delta 0), pack-reused 911
接收对象中: 100% (928/928), 1.43 MiB | 931.00 KiB/s, 完成.
处理 delta 中: 100% (320/320), 完成.
eda@abc-MW51-HP0-00:~/Desktop/wujian$
```

2.下载工具链：

在平头哥的官网上下载工具链。版本可能不同，下载最新的就好。

地址：https://occ.t-head.cn/community/download?id=3913221581316624384

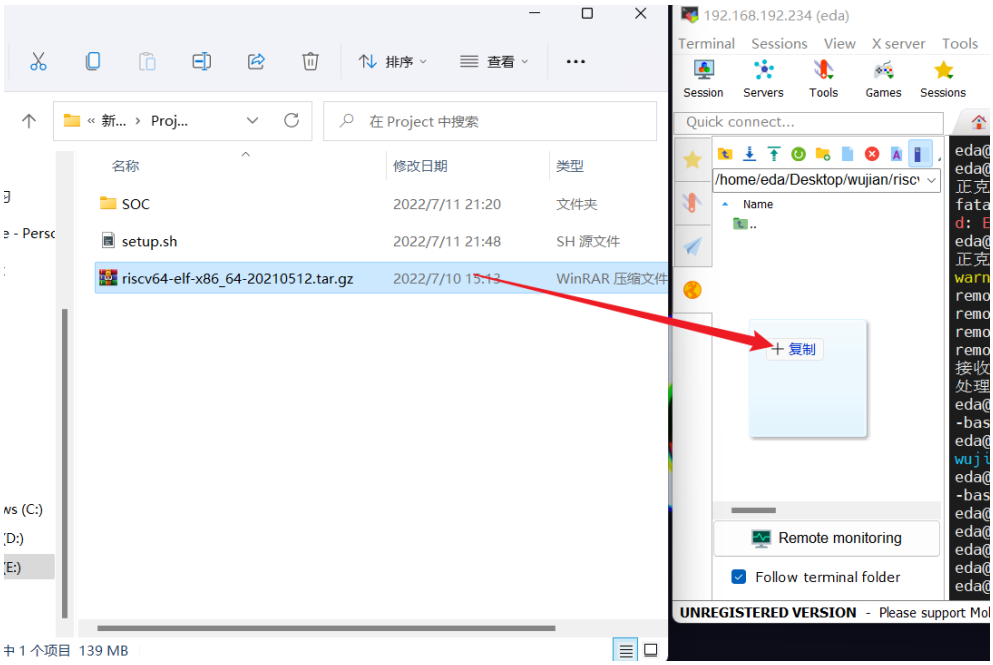


创建工具链文件夹：mkdir riscv_toolchain

切换工作目录到该文件夹下：cd riscv_toolchain

```
eda@abc-MW51-HP0-00:~/Desktop/wujian$ mkdir riscv_toolchain
eda@abc-MW51-HP0-00:~/Desktop/wujian$ cd riscv_toolchain
eda@abc-MW51-HP0-00:~/Desktop/wujian/riscv_toolchain$
```

将下载好的工具链压缩包放到该文件夹下，这里直接拖拽复制，将文件由 Windows 本机上传到了该 Ubuntu 系统的工作站上。也可用 cp 复制命令复制到文件夹下。



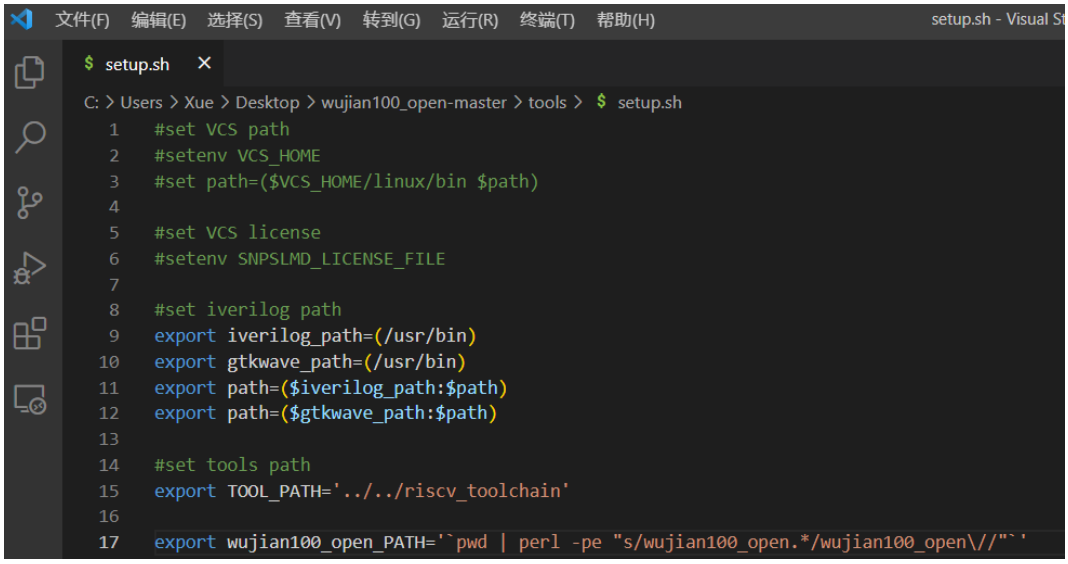
解压文件：tar -zxvf riscv64-elf-x86_64-20210512.tar.gz

```
riscv64-elf-x86_64-20210512.tar.gz
eda@abc-MW51-HP0-00:~/Desktop/wujian/riscv_toolchain$ tar -zxvf riscv64-elf-x86_64-20210512.tar.gz
```

等待解压完成。

3.修改脚本：

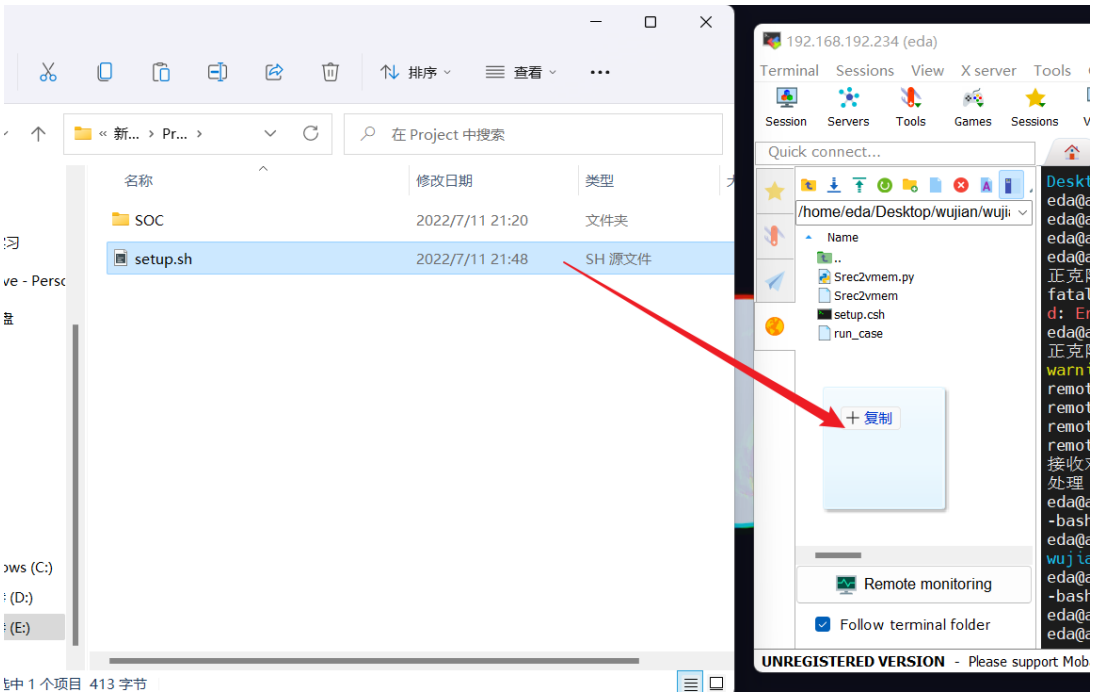
由于原本的脚本是 setup.csh 文件，在 bash 环境下有一些不兼容的地方。这里重新建立一个 setup.sh 脚本，如下图所示，放入 wujian100_open/tools 目录下(原 setup.csh 目录)



Cd 到 wujian100_open/tools 目录下：

```
eda@abc-MW51-HP0-00:~/Desktop/wujian/riscv_toolchain$ cd ../wujian100_open/tools
eda@abc-MW51-HP0-00:~/Desktop/wujian/wujian100_open/tools$
```

这里也是将 Windows 本机的 setup.sh 文件上传到工作站上。



输入： source setup.sh

```
eda@abc-MW51-HP0-00:~/Desktop/wujian/wujian100_open/tools$ ls
run_case  setup.csh  setup.sh  Srec2vmem  Srec2vmem.py
eda@abc-MW51-HP0-00:~/Desktop/wujian/wujian100_open/tools$ source setup.sh
```

4.运行仿真

进入到 wujian100_open/workdir 目录下运行仿真：cd ../workdir

开始运行仿真： ../tools/run_case -sim_tool iverilog ../case/timer/timer_test.c

```
eda@abc-MW51-HP0-00:~/Desktop/wujian/wujian100_open/tools$ cd ../workdir
eda@abc-MW51-HP0-00:~/Desktop/wujian/wujian100_open/workdir$ ../tools/run_case -sim_tool iverilog ../
case/timer/timer_test.c
```

```
        ******START TO LOAD PROGRAM******


Hello Friend!

timer test successfully
**************************************

*              Test Pass              *

**************************************

../tb/busmnt.v:109: $finish called at 1039250 (100ps)

Step4 (Run simulation) is finished
eda@abc-MW51-HP0-00:~/Desktop/wujian/wujian100_open/workdir$
```

仿真成功。

5.仿真波形的查看

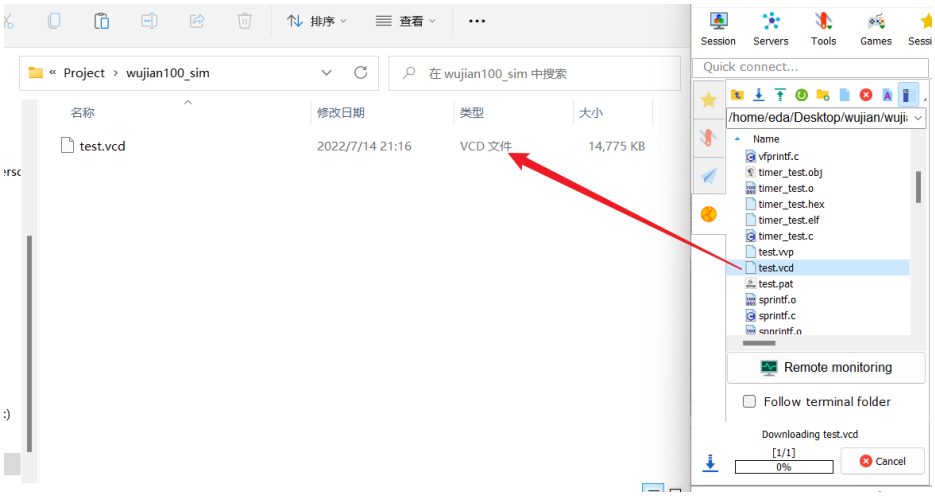
在运行完仿真后，在 wujian100_open/workdir 目录下会生成 test.vcd，如下图所示。VCD 文件包含了信号的变化信息，就相当于记录了整个仿真的信息，我们可以用这个文件来再现仿真，也就能够显示波形。

```
eda@abc-MW51-HP0-00:~/Desktop/wujian/wujian100_open/workdir$ ls
config.h    getc.c       __lltostr.o      putchar.o       test.pat       __v_printf.c
crt0.o      getchar.c    __ltostr.c       putc.o          test.vcd       vprintf.c
crt0.s      getchar.o    __ltostr.o       puts.c          test.vvp       __v_printf.o
datatype.h  getc.o       Makefile         puts.o          timer_test.c   vprintf.o
__dtostr.c  __isinf.c    minilibc_stdio.h rtl_gpr.log     timer_test.elf vsnprintf.c
__dtostr.o  __isinf.o    printf.c         run_case.report timer_test.hex vsnprintf.o
fprintf.c   __isnan.c    printf.h         snprintf.c      timer_test.o   vsprintf.c
fprintf.o   __isnan.o    printf.o         snprintf.o      timer_test.obj vsprintf.o
fputc.c     linker.lcf   putc.c          sprintf.c       vfprintf.c     vtimer.h
fputc.o     __lltostr.c  putchar.c        sprintf.o       vfprintf.o
eda@abc-MW51-HP0-00:~/Desktop/wujian/wujian100_open/workdir$
```

因为 VCD 是 Verilog HDL 语言标准的一部分，因此所有的 verilog 的仿真器都能够查看该文件。

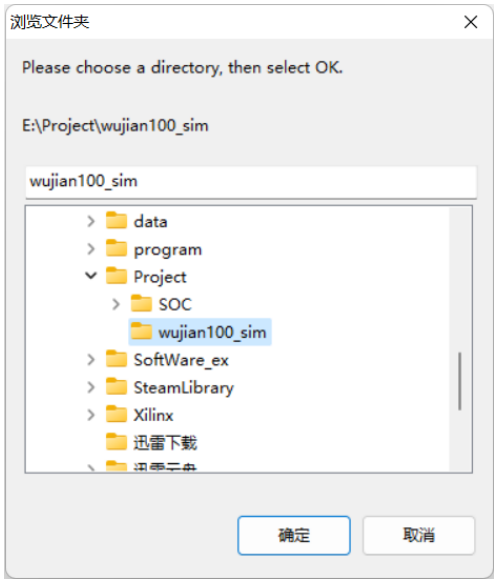
这里介绍用 Modelsim 软件查看波形。将生成的 test.vcd 文件传输到 Windows 主

机上，存放于 E:/Project/wujian100_sim 文件夹下。



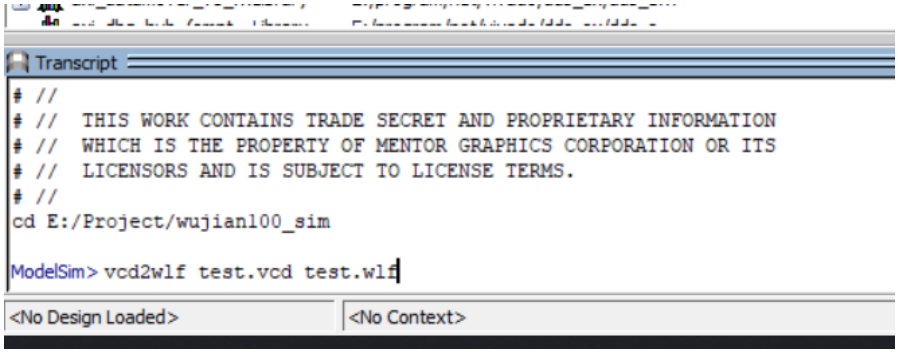
打开 Modelsim 软件。

点击 File->Change Directory，将目录更改至 test.vcd 的目录下；或者直接在在 Modelsim 中的控制台输入：cd E:/Project/wujian100_sim 直接定为到该文件夹下。

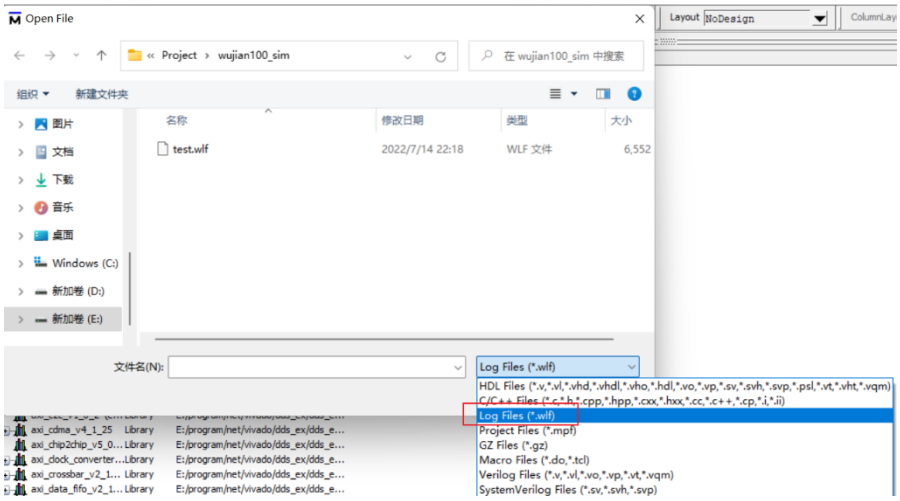


因为 Modelsim 只支持.wlf 波形文件，所以需要做格式转换。

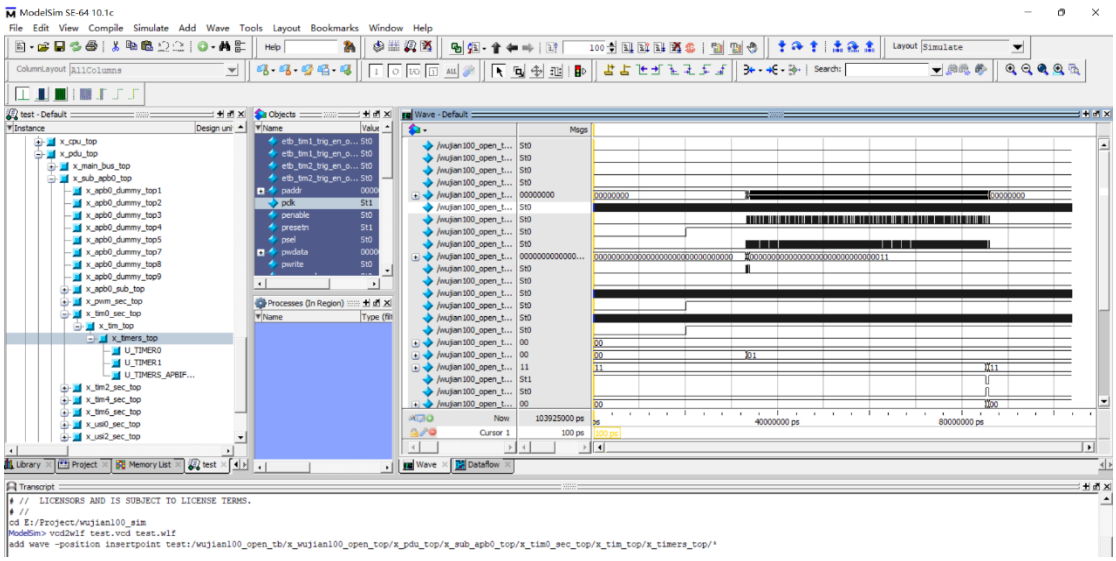
在 Modelsim 中的控制台输入：vcd2wlf test.vcd test.wlf



在 Modelsim 中打开 test.wlf，点击 File->Open->test.wlf



在 object 标签中选取需要观察的信号添加到波形窗即可。



实验二：Wujian100 的 SoC 实现

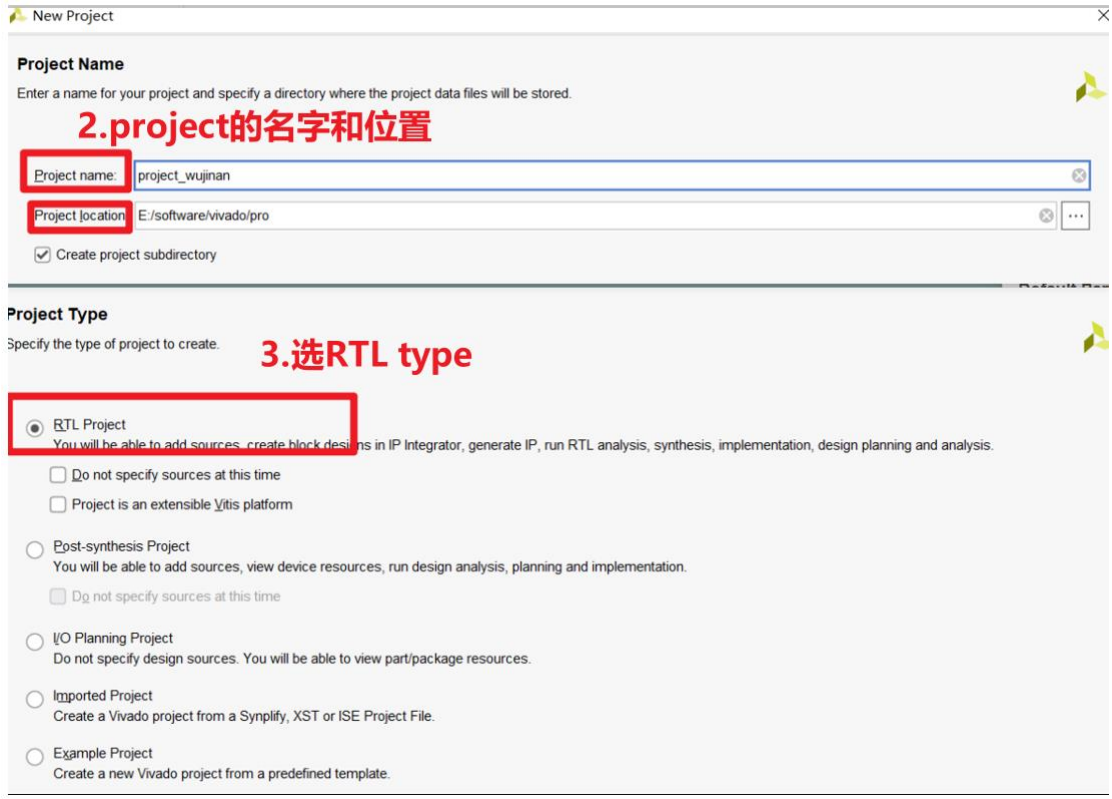
实验目标

主要完成 wujian100 SoC 在 FPGA 上的部署。学会使用 Xilinx vivado 工具。

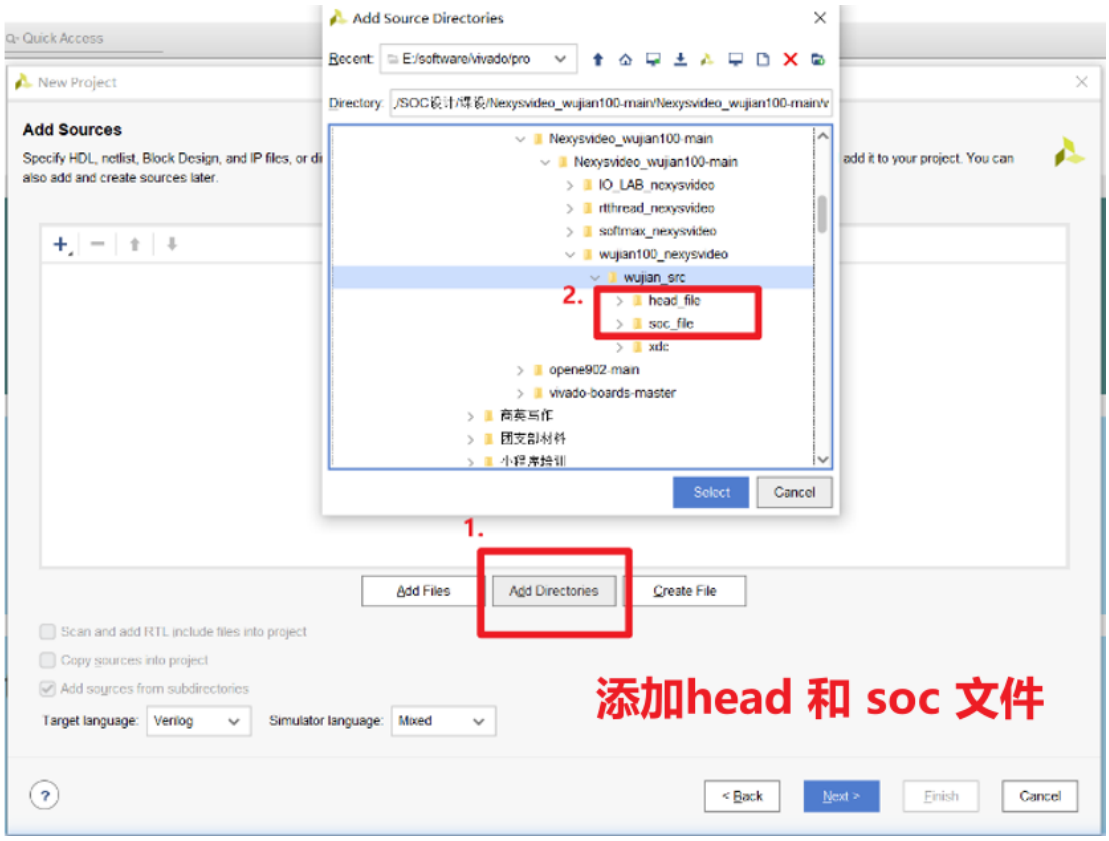
实验步骤

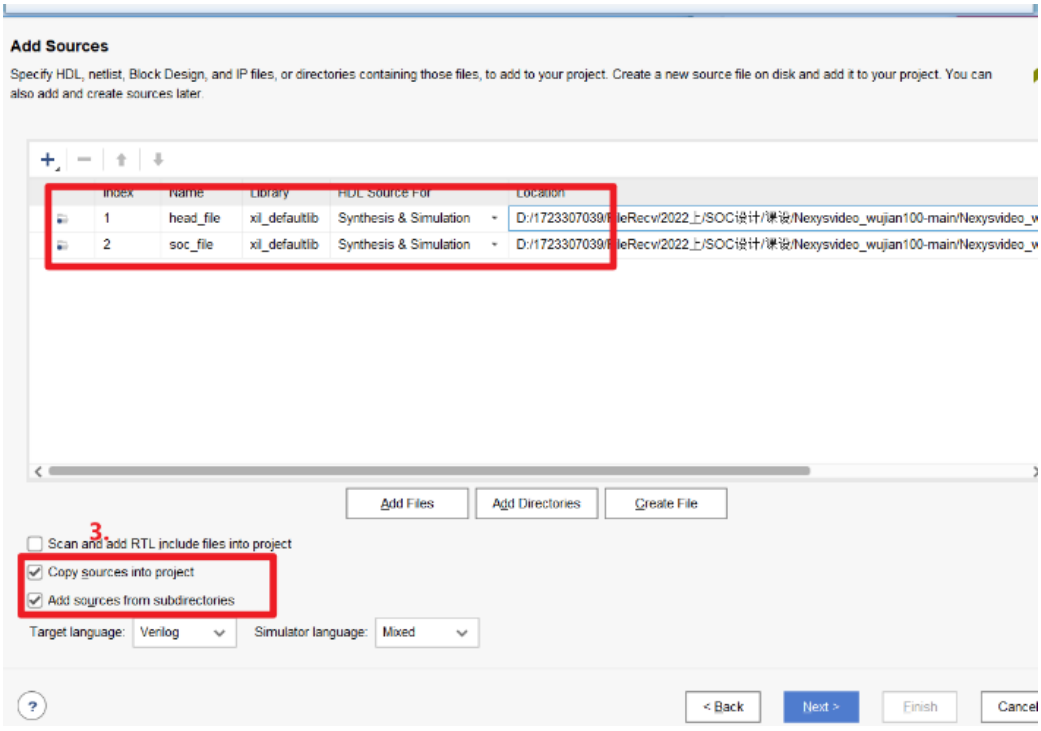
1. 把 wujian100 用 Vivado 打开

   1) 创建工程：

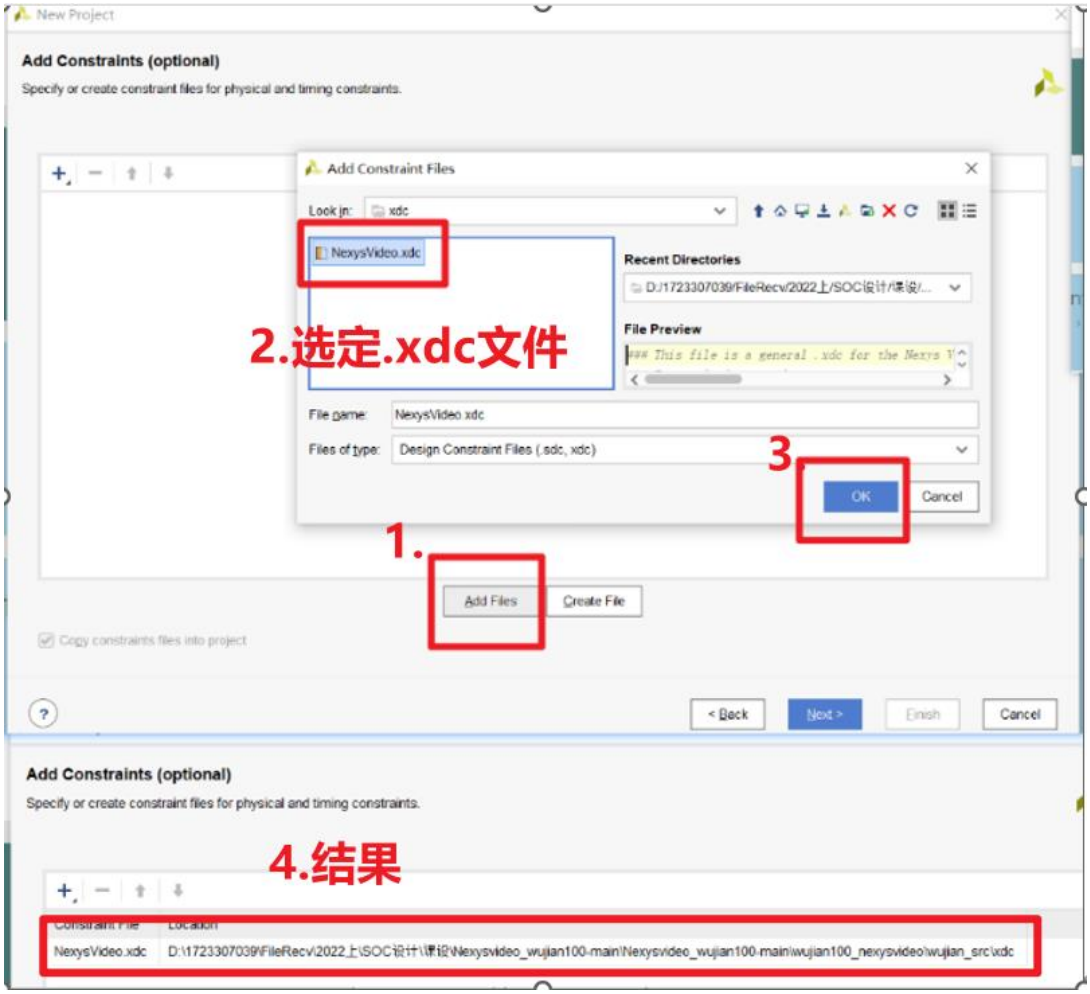


next 后添加 verilog 源码：

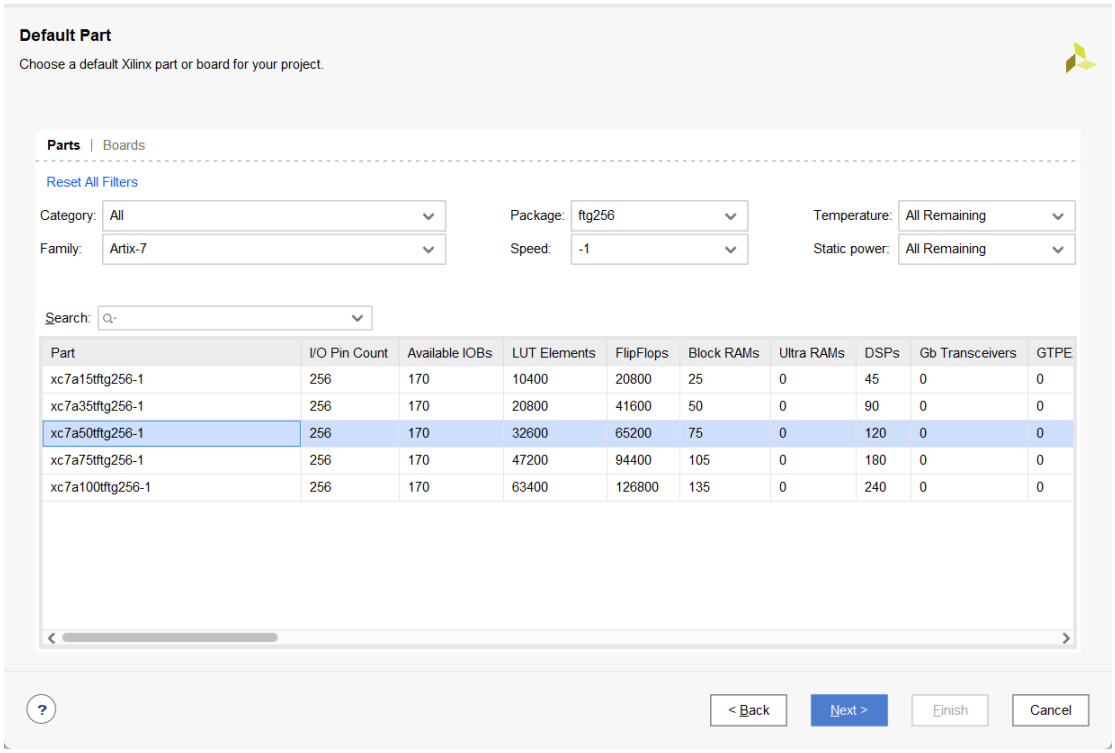




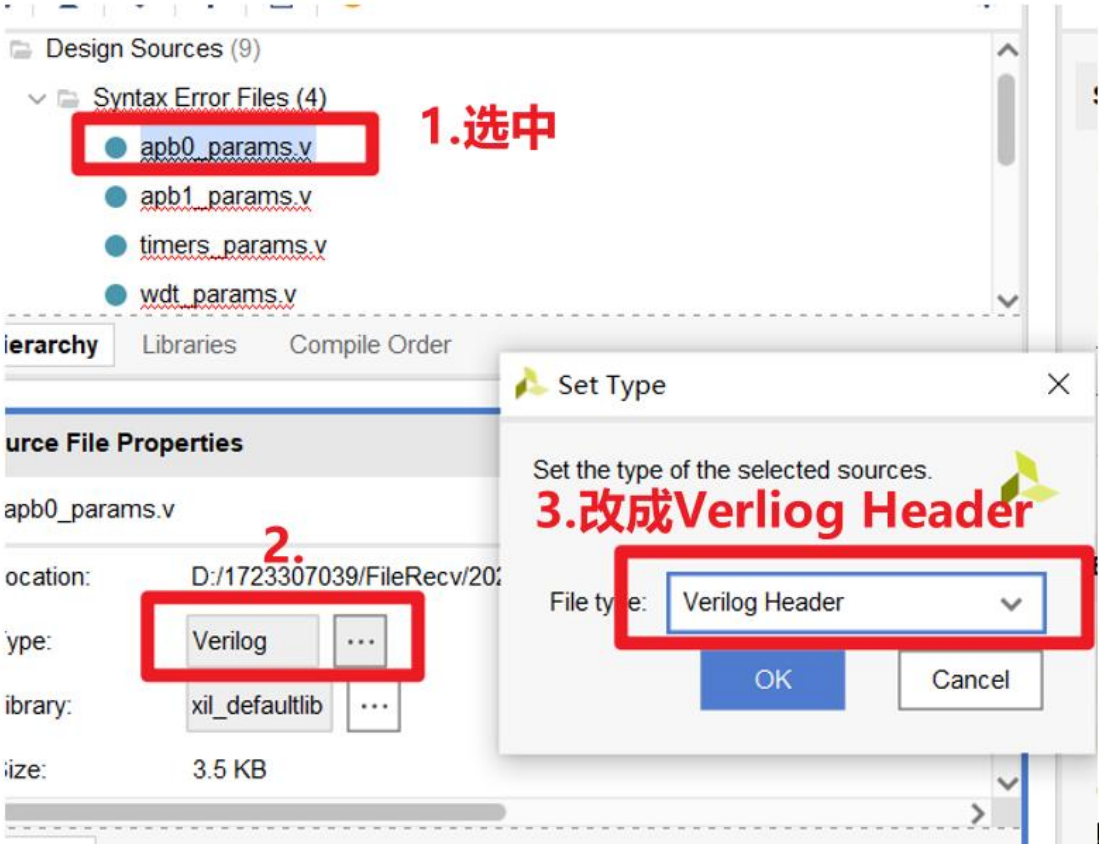
点击 next 后添加 xdc 文件


New Project
Add Constraints (optional)
Specify or create constraint files for physical and timing constraints.
Add Constraint Files
Look in: xdc
NexysVideo.xdc
Recent Directories
File Preview
2.选定.xdc文件
File name: NexysVideo.xdc
Files of type: Design Constraint Files (.sdc, xdc)
3
OK
Cancel
1.
Add Files
Create File
Copy constraints files into project
< Back
Next >
Finish
Cancel
Add Constraints (optional)
Specify or create constraint files for physical and timing constraints.
4.结果
Constraint File
Location
NexysVideo.xdc
D:\1723307039\FileRecv\2022上\SOC设计\开题\Nexysvideo_wujian100-main\Nexysvideo_wujian100-main\wujian100_nexysvideo\wujian_src\xdc

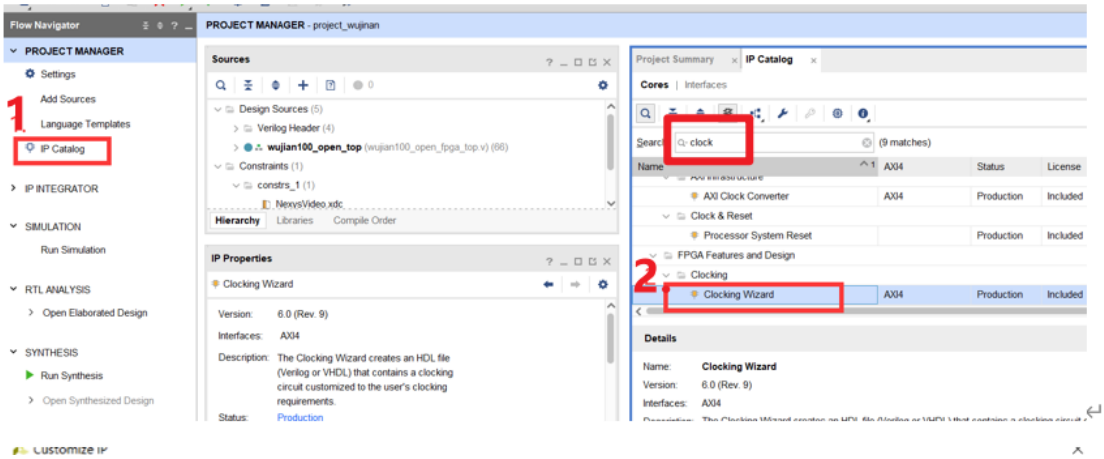
1) 选择开发板

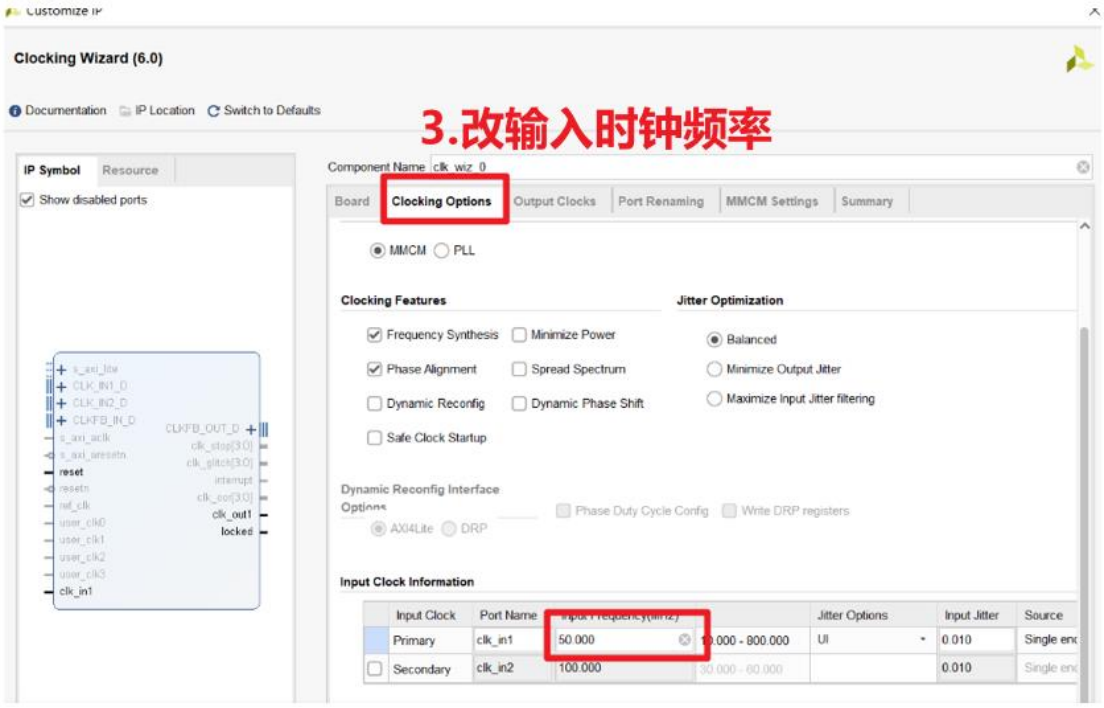

Default Part
Choose a default Xilinx part or board for your project.
Parts | Boards
Reset All Filters
Category: All
Family: Artix-7
Package: ftg256
Speed: -1
Temperature: All Remaining
Static power: All Remaining
Search:
Part | I/O Pin Count | Available IOBs | LUT Elements | FlipFlops | Block RAMs | Ultra RAMs | DSPs | Gb Transceivers | GTPE
xc7a15tftg256-1 | 256 | 170 | 10400 | 20800 | 25 | 0 | 45 | 0 | 0
xc7a35tftg256-1 | 256 | 170 | 20800 | 41600 | 50 | 0 | 90 | 0 | 0
xc7a50tftg256-1 | 256 | 170 | 32600 | 65200 | 75 | 0 | 120 | 0 | 0
xc7a75tftg256-1 | 256 | 170 | 47200 | 94400 | 105 | 0 | 180 | 0 | 0
xc7a100tftg256-1 | 256 | 170 | 63400 | 126800 | 135 | 0 | 240 | 0 | 0
< Back
Next >
Finish
Cancel

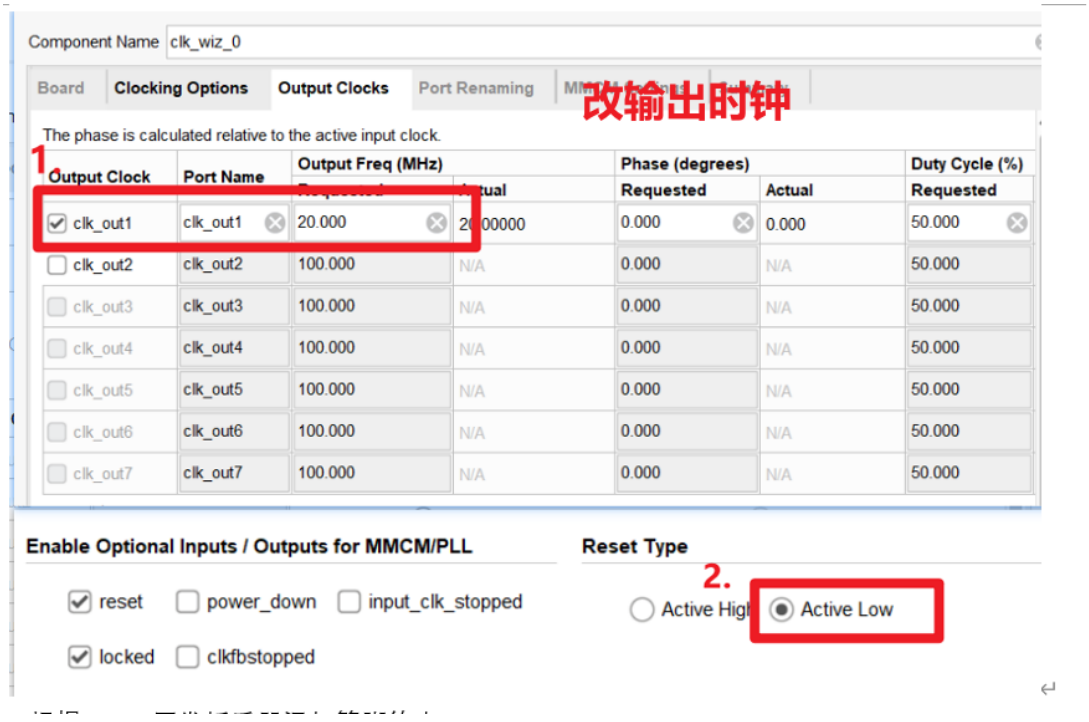
2) 文件添加进来后，VIVADO 会自动识别、编译、分析，VIVADO 分析文件中的错误，用红色波浪线标识（错误原因是没有识别出头文件，将这四个文件类型改为头文件类型即可）



3) 调用时钟 IP



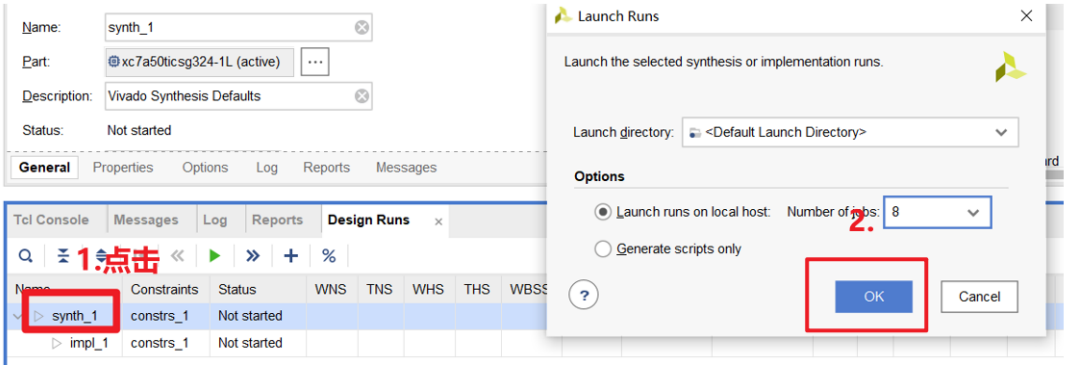




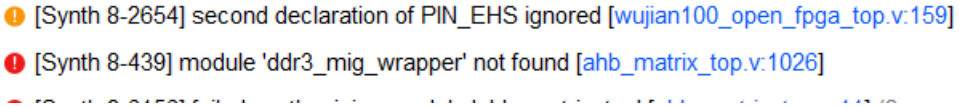
2. 根据 perfv 开发板手册添加管脚约束

详细见文件“xdc 的更改”.

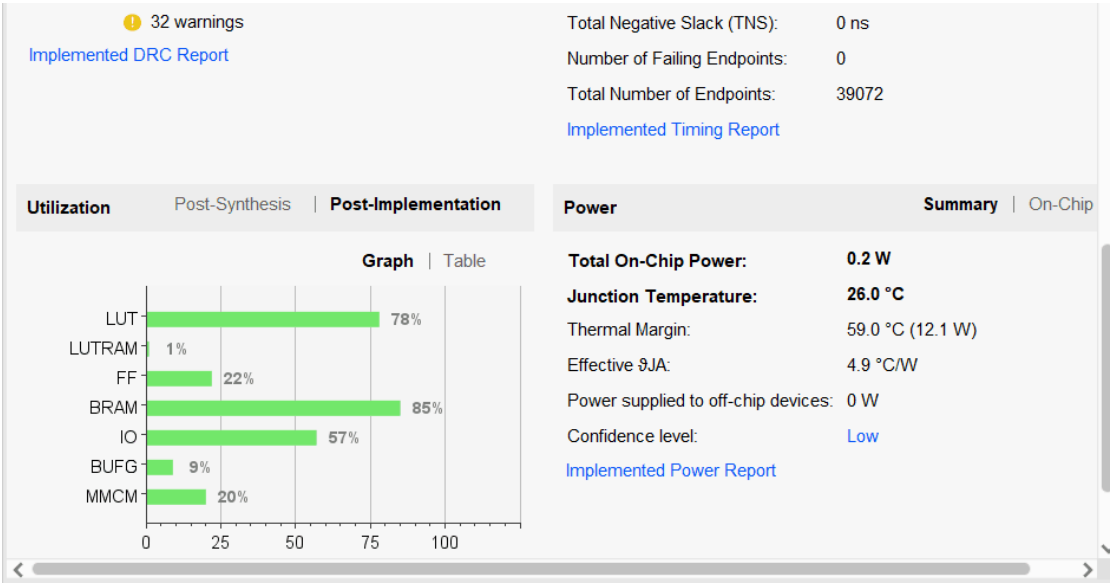
3. 综合工程，综合无误后进行 bit 流下载

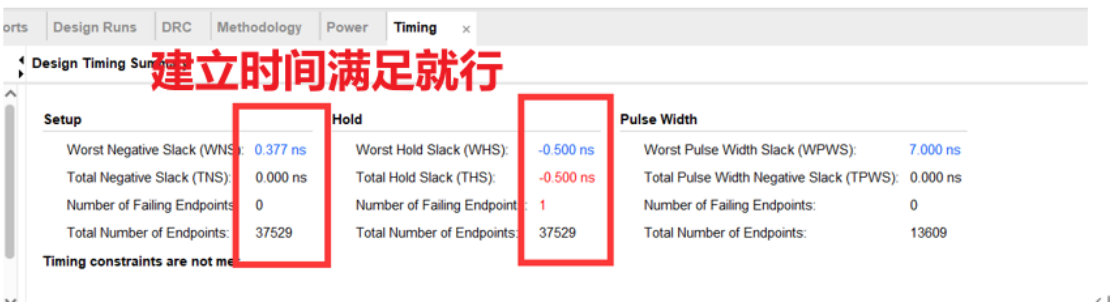


可能出现 error-某某 module 未找到，注释掉这个 module 就行

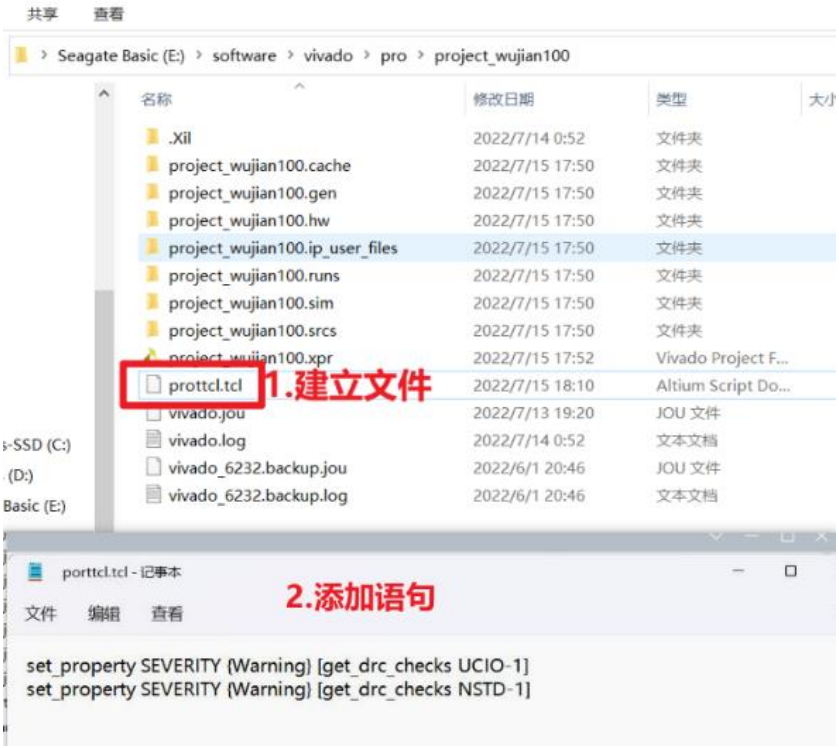


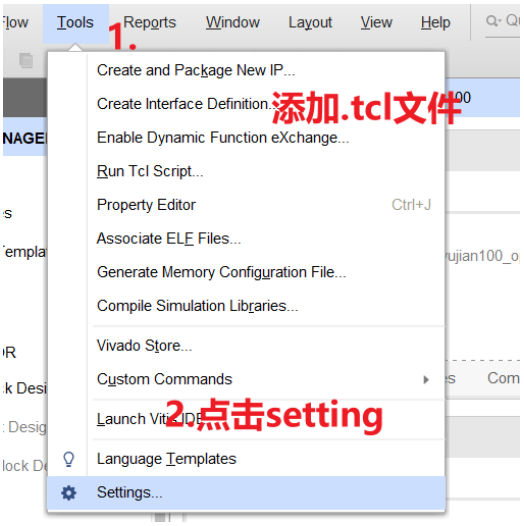
综合结果如下：

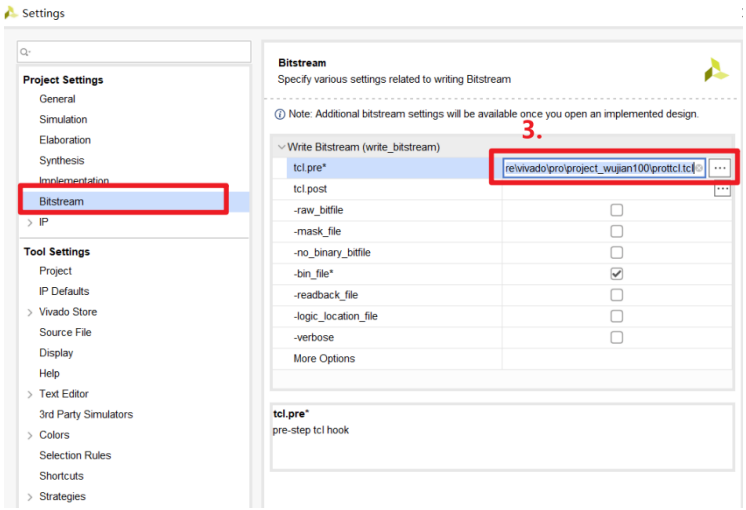




如果生成 bit 流时出现“没有约束的逻辑端口的错误”：可以添加 tcl 文件消除错误

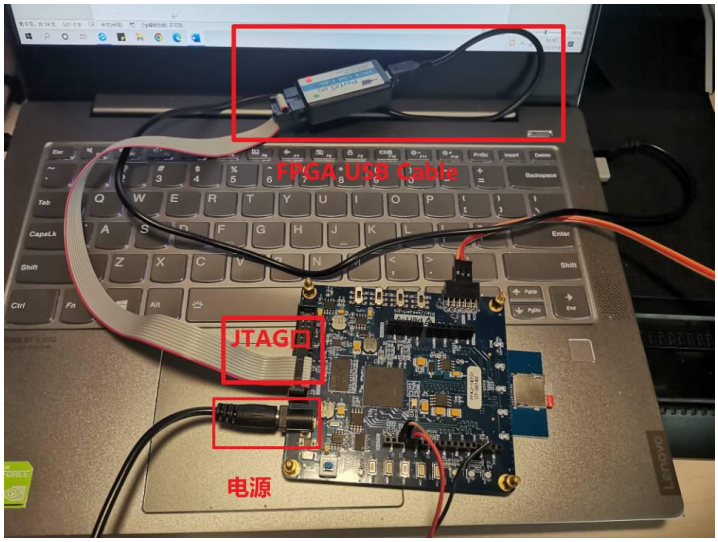

共享
查看
Seagate Basic (E:) › software › vivado › pro › project_wujian100
名称
修改日期
类型
.Xil
2022/7/14 0:52
文件夹
project_wujian100.cache
2022/7/15 17:50
文件夹
project_wujian100.gen
2022/7/15 17:50
文件夹
project_wujian100.hw
2022/7/15 17:50
文件夹
project_wujian100.ip_user_files
2022/7/15 17:50
文件夹
project_wujian100.runs
2022/7/15 17:50
文件夹
project_wujian100.sim
2022/7/15 17:50
文件夹
project_wujian100.srcs
2022/7/15 17:50
文件夹
project_wujian100.xpr
2022/7/15 17:52
Vivado Project F...
prottcl.tcl
1.建立文件
2022/7/15 18:10
Altium Script Do...
vivado.jou
2022/7/13 19:20
JOU 文件
vivado.log
2022/7/14 0:52
文本文档
vivado_6232.backup.jou
2022/6/1 20:46
JOU 文件
vivado_6232.backup.log
2022/6/1 20:46
文本文档
porttcl.tcl - 记事本
文件
编辑
查看
2.添加语句
set_property SEVERITY {Warning} [get_drc_checks UCIO-1]
set_property SEVERITY {Warning} [get_drc_checks NSTD-1]


Tools
Reports
Window
Layout
View
Help
1.
Create and Package New IP...
Create Interface Definition...
添加.tcl文件
Enable Dynamic Function eXchange...
Run Tcl Script...
Property Editor
Ctrl+J
Associate ELF Files...
Generate Memory Configuration File...
Compile Simulation Libraries...
Vivado Store...
Custom Commands
Launch Vitis IDE
2.点击setting
Language Templates
Settings...

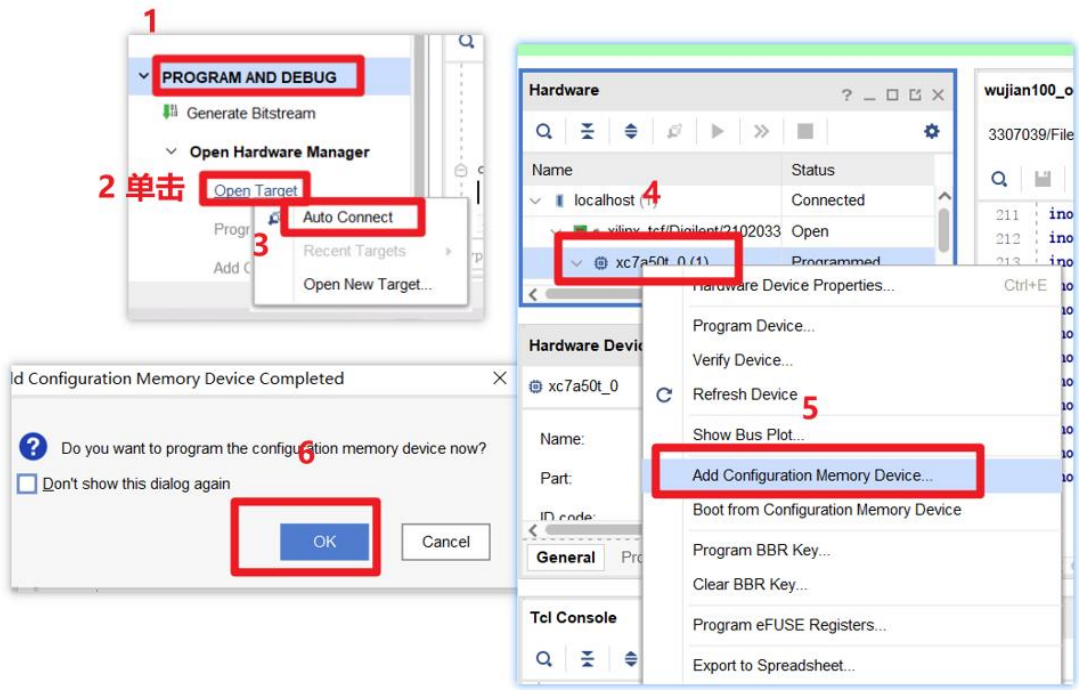


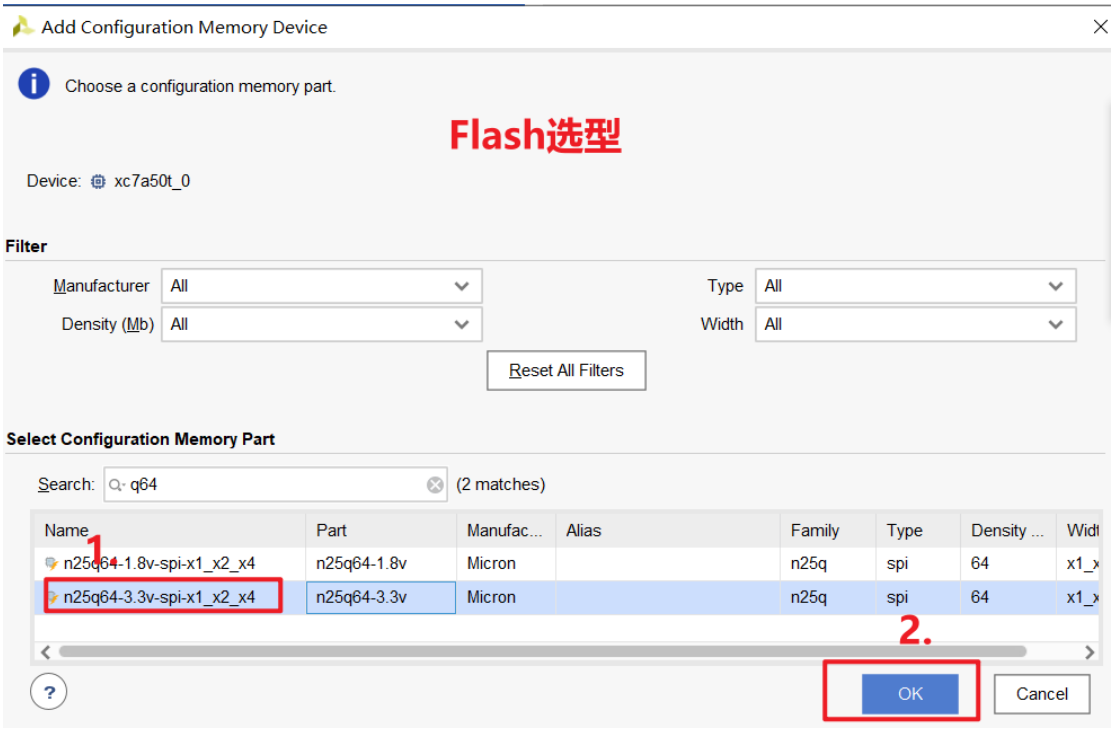
4. 下载到开发板上

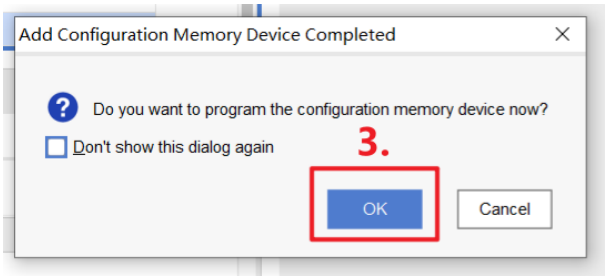
1) 接口连接电脑和开发板

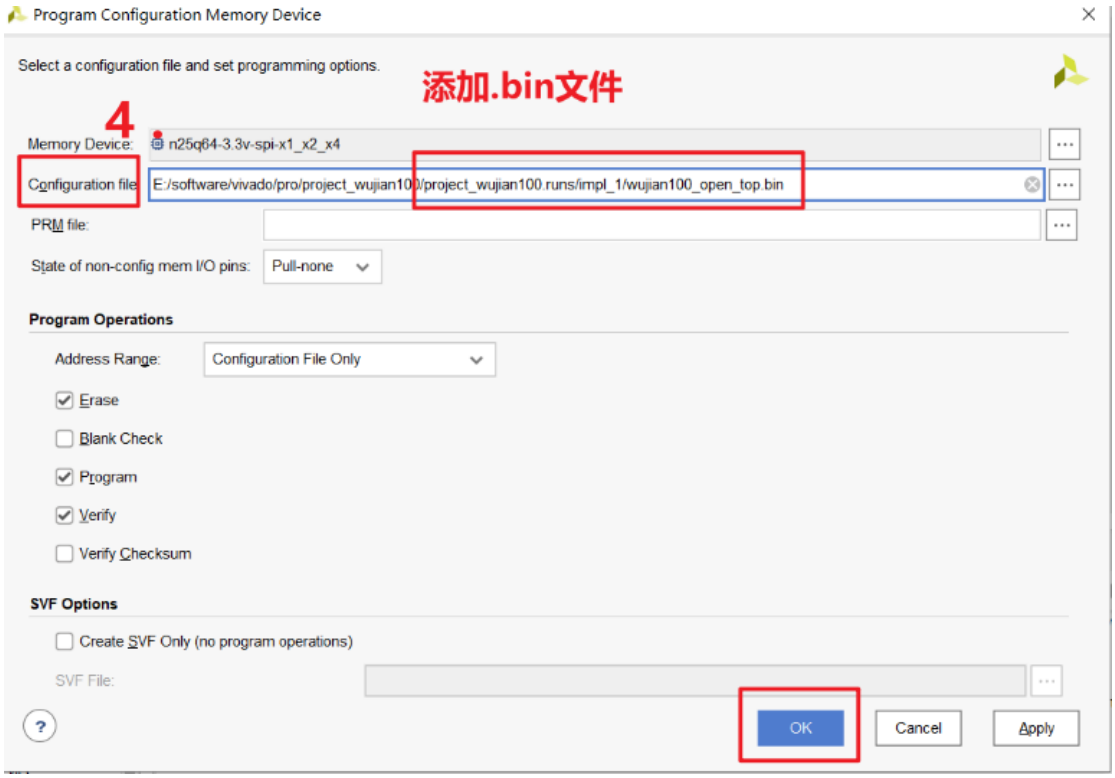


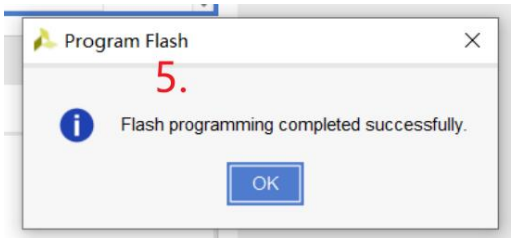
2) 固化 FLASH：


1
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
2 单击
Open Target
Auto Connect
3
Recent Targets
Open New Target...
Hardware
Name
Status
localhost
Connected
Open
4
Hardware Device Properties...
Ctrl+E
Program Device...
Verify Device...
Refresh Device
5
Show Bus Plot...
Add Configuration Memory Device...
Boot from Configuration Memory Device
Program BBR Key...
Clear BBR Key...
Program eFUSE Registers...
Export to Spreadsheet...
xc7a50t_0
Name:
Part:
General
Tcl Console
Configuration Memory Device Completed
Do you want to program the configuration memory device now?
6
Don't show this dialog again
OK
Cancel


Add Configuration Memory Device
Choose a configuration memory part.
Flash选型
Device: xc7a50t_0
Filter
Manufacturer
All
Density (Mb)
All
Type
All
Width
All
Reset All Filters
Select Configuration Memory Part
Search:
q64
(2 matches)
Name
Part
Manufac...
Alias
Family
Type
Density ...
Widt
1
n25q64-1.8v-spi-x1_x2_x4
n25q64-1.8v
Micron
n25q
spi
64
n25q64-3.3v-spi-x1_x2_x4
n25q64-3.3v
Micron
n25q
spi
64
2.
OK
Cancel


Add Configuration Memory Device Completed
Do you want to program the configuration memory device now?
Don't show this dialog again
3.
OK
Cancel





显示 5 表明 wujian100 已经下载到了开发板里，并且上电复位就可以运行。

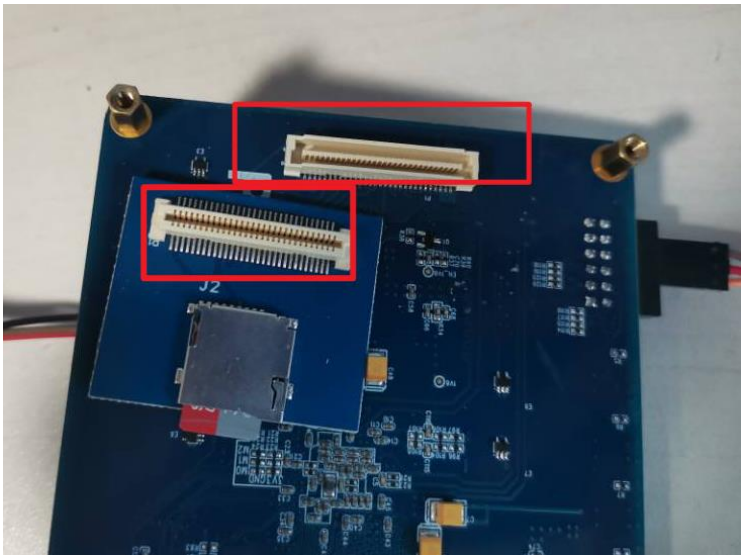
实验三：I/O 口与 SD 卡读写实验

实验目标

（1）在 wujian100 SoC 在 FPGA 上完成部署之后，通过设置该 SoC 的 I/O 引脚为高电平或者低电平，观察 LED 的亮灭，学会使用 E902 的 I/O。

（2）学会使用 E902 处理器读写 SD 卡，需要 SD 卡和 SD 卡的拓展板。

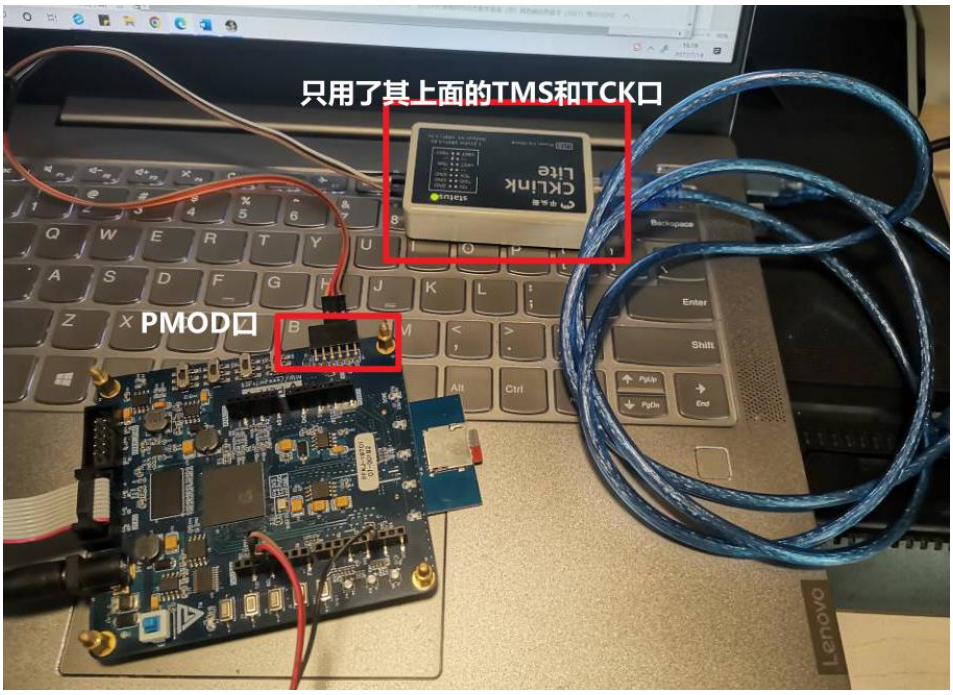
注意：SD 卡相关功能需要 Pref 的 SD 卡拓展板，借插口如下。SD 卡的程序和 LED 的程序是一起的，所以一起就可以测试。（不要热插拔）



实验步骤

1. 连接电脑和 wujian100

Clink 与 PMOD 连接：



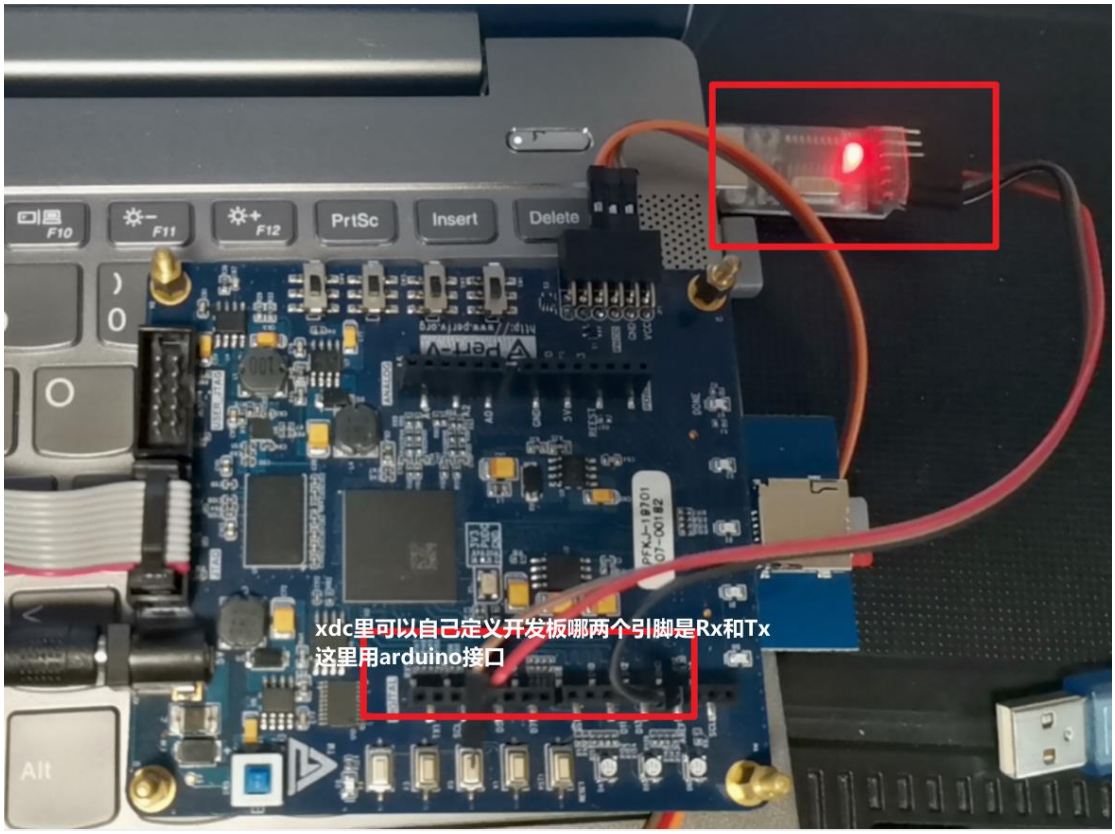
如下对应关系：

CKLink JTAG 引脚与 PMOD 对应关系

CKLink 接口					
JTAG 引脚	信号名称	对应引脚	JTAG 引脚	信号名称	对应引脚
1	Verf	12(-)	2	NC	6(-)
3	GND	11(-)	4	NC	5(-)
5	TMS	10(F13)	6	TDO	4(D11)
7	TCK	9(F12)	8	TRST	3(E11)
9	TDI	8(D15)	10	NRST	2(E13)

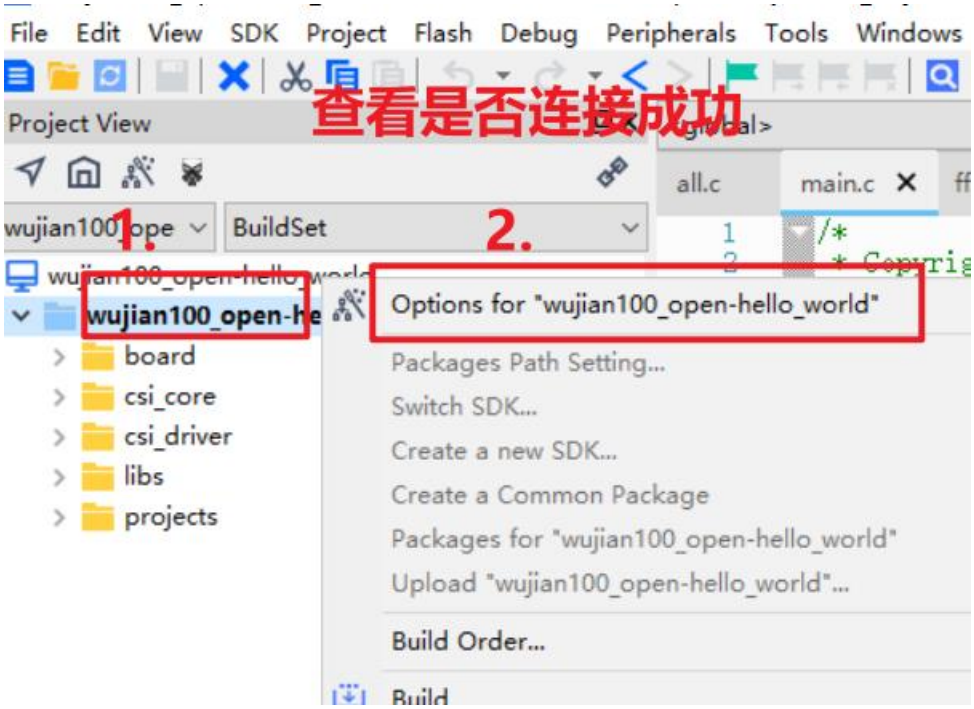
JP1（PMOD）接口					
JP1 引脚	信号名称	FPGA 引脚	JP1 引脚	信号名称	FPGA 引脚
1	B15_L13P	E12	2	B15_L13N	E13
3	B15_L14P	E11	4	B15_L14N	D11
5	GND	-	6	3V3	-
7	B15_L15P	D14	8	B15_L15N	D15
9	B15_L16P	F12	10	B15_L16N	F13
11	GND	-	12	3V3	-

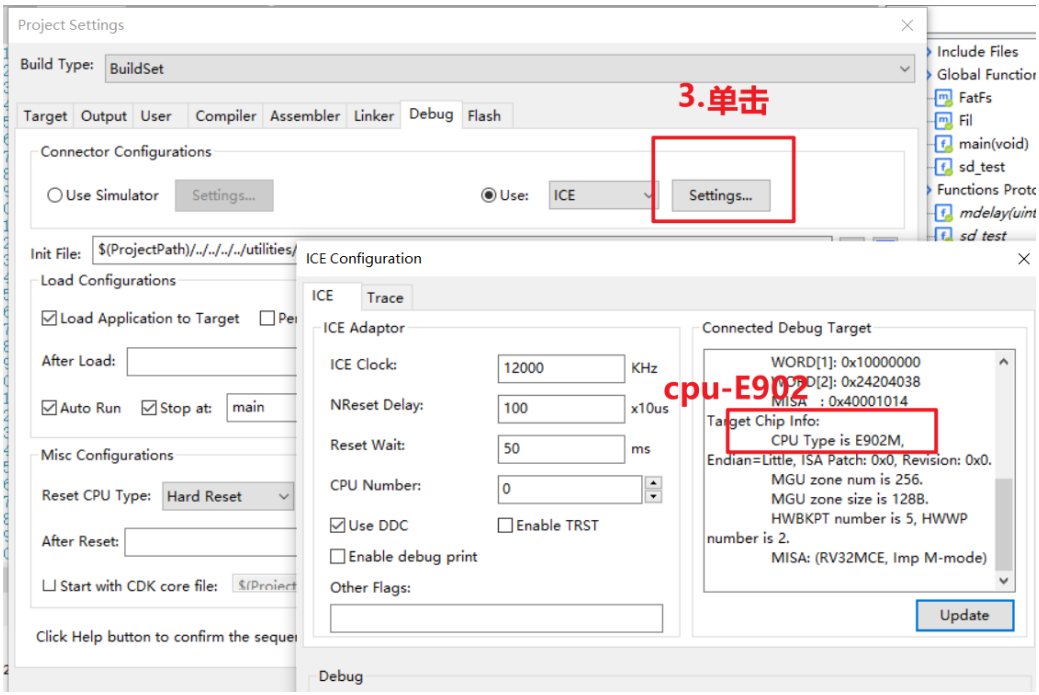
2. 连接 USB 转串口下载器和开发




打开(O)
固定到快速访问
管理(G)
固定到"开始"屏幕(P)
映射网络驱动器(N)...
断开网络驱动器的连接(C)...
创建快捷方式(S)
删除(D)
重命名(M)
属性(R)
1.
相关设置
BitLocker 设置
设备管理器
2.
远程桌面
系统保护
高级系统设置
重命名这台电脑
获取帮助
提供反馈
设备管理器
文件(F) 操作(A) 查看(V) 帮助(H)
LAPTOP-G6NQ2KBA
安全设备
处理器
磁盘驱动器
存储控制器
打印队列
电池
USB-SERIAL CH340 (COM13)
3.
蓝牙链接上的标准串行 (COM3)
蓝牙链接上的标准串行 (COM4)
蓝牙链接上的标准串行 (COM5)
蓝牙链接上的标准串行 (COM6)
蓝牙链接上的标准串行 (COM7)
蓝牙链接上的标准串行 (COM8)
固件
计算机
监视器
键盘
蓝牙
人机接口设备
软件设备
可以通过插拔来确定
转换器对应的COM序号

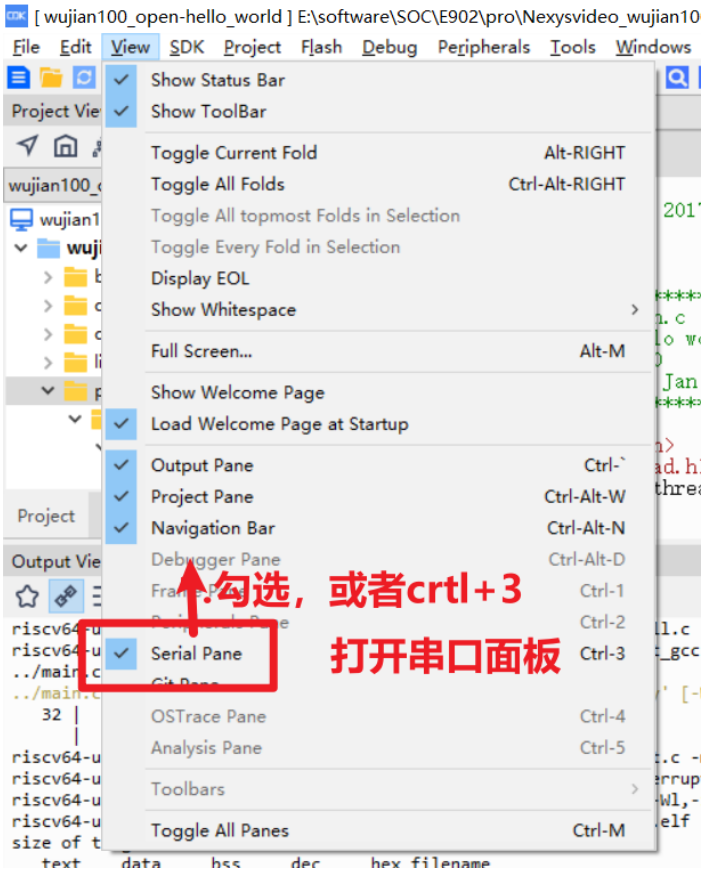
3. 编译调试

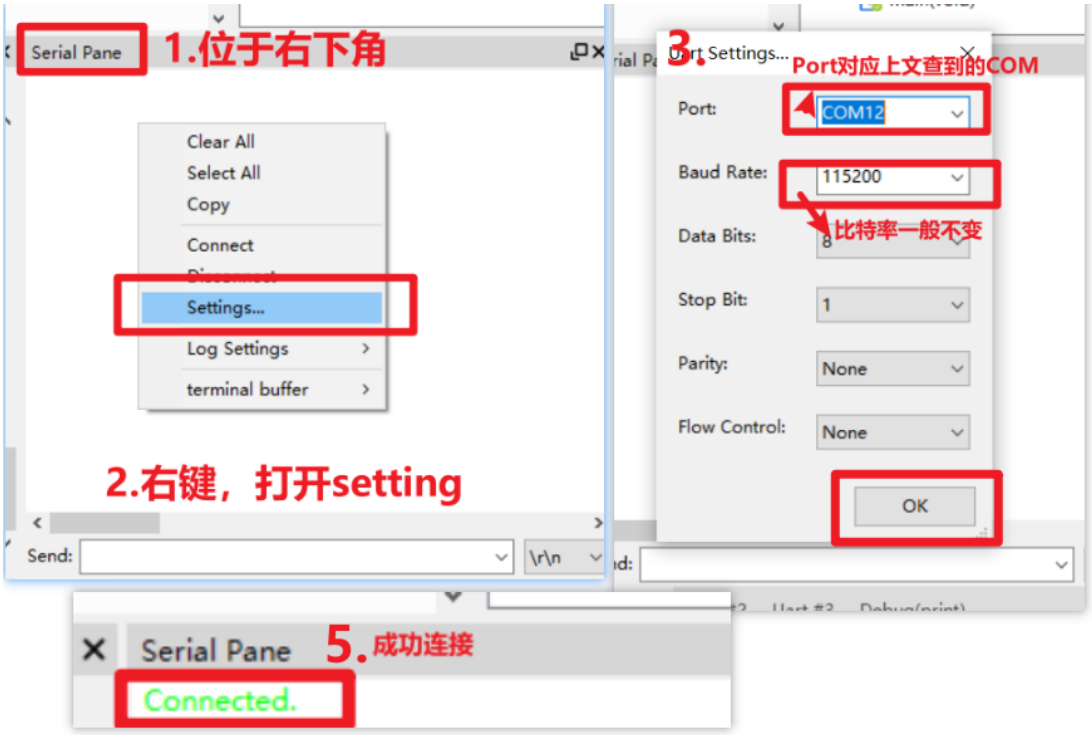

File Edit View SDK Project Flash Debug Peripherals Tools Windows
查看是否连接成功
Project View
all.c
main.c
BuildSet
1.
2.
wujian100_open-hello_world
board
csi_core
csi_driver
libs
projects
Options for "wujian100_open-hello_world"
Packages Path Setting...
Switch SDK...
Create a new SDK...
Create a Common Package
Packages for "wujian100_open-hello_world"
Upload "wujian100_open-hello_world"...
Build Order...
Build



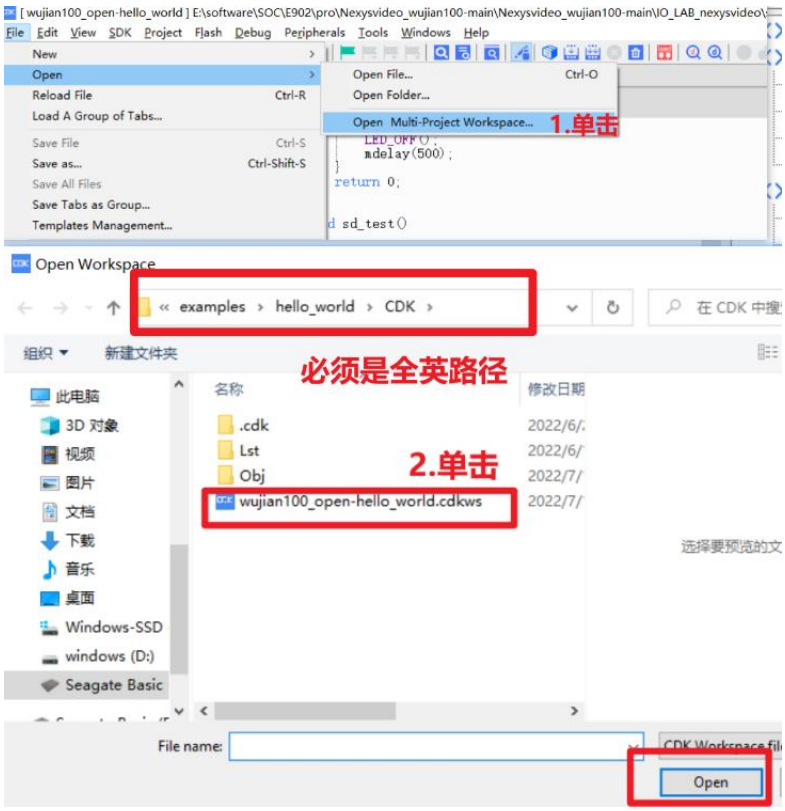
显示出来了 E902 才可以进行下一步，没有显示出来可以去查一下 PMOD 和 CLINK 的连接是不是错的，或者没连上。

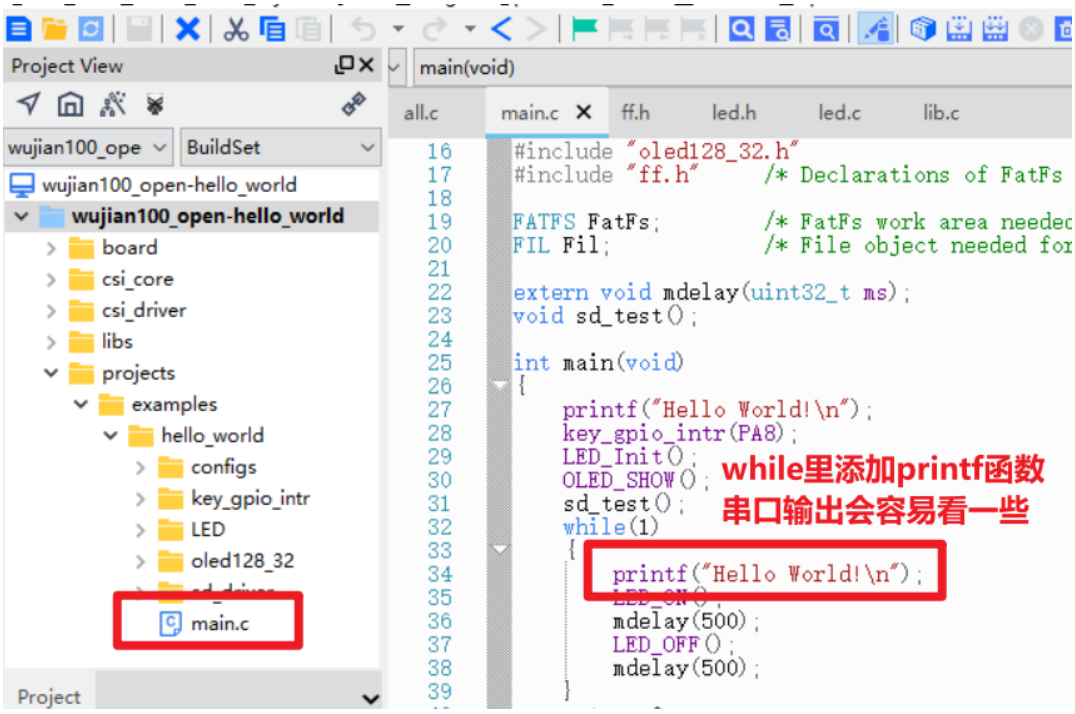
4. CDK 的串口窗口

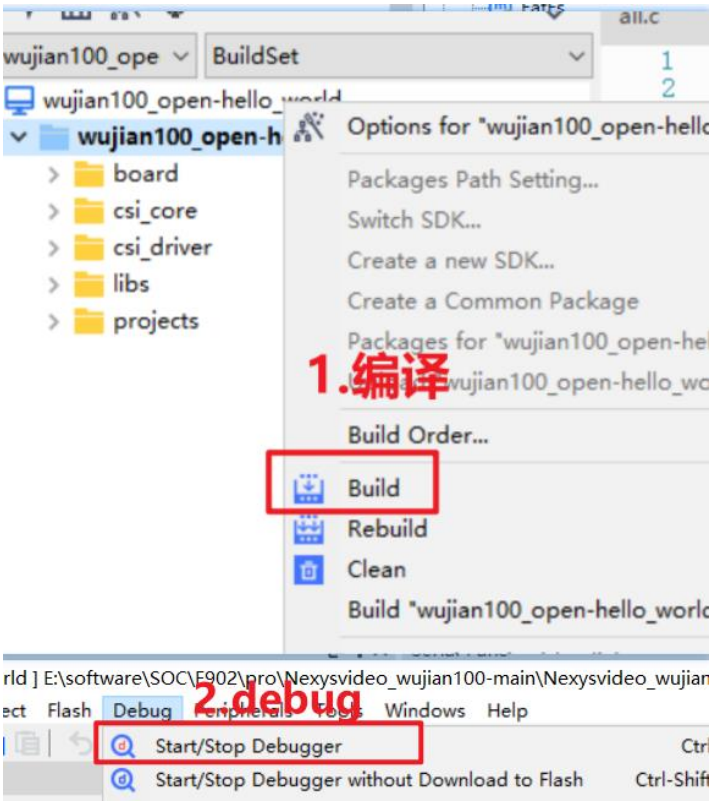



Serial Pane
1.位于右下角
Clear All
Select All
Copy
Connect
Settings...
Log Settings
terminal buffer
2.右键，打开setting
Send:
\r\n
3.
Port Settings...
Port对应上文查到的COM
Port:
COM12
Baud Rate:
115200
比特率一般不变
Data Bits:
Stop Bit:
1
Parity:
None
Flow Control:
None
OK
Serial Pane
5.成功连接
Connected.

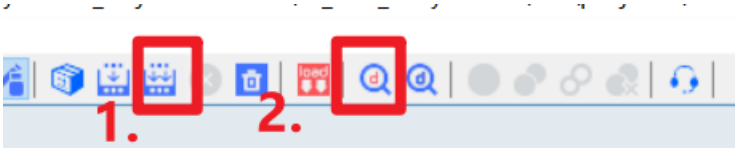
5. 下载程序


File Edit View SDK Project Flash Debug Peripherals Tools Windows Help
New
Open
Reload File Ctrl-R
Load A Group of Tabs...
Save File Ctrl-S
Save as... Ctrl-Shift-S
Save All Files
Save Tabs as Group...
Templates Management...
Open File... Ctrl-O
Open Folder...
Open Multi-Project Workspace...
1.单击
Open Workspace
examples › hello_world › CDK ›
必须是全英路径
组织
新建文件夹
此电脑
3D 对象
视频
图片
文档
下载
音乐
桌面
Windows-SSD
windows (D:)
Seagate Basic
名称
修改日期
.cdk
Lst
Obj
2.单击
wujian100_open-hello_world.cdkws
File name:
Open

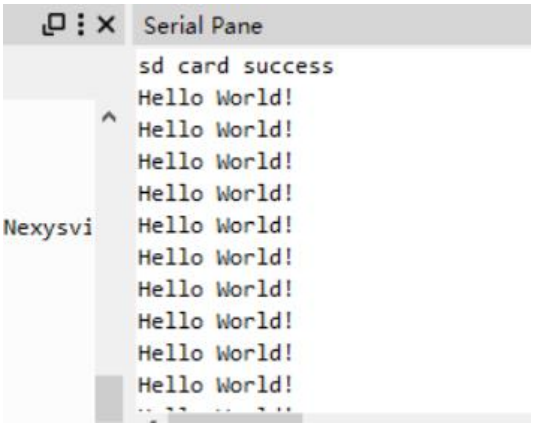

Project View
main(void)
wujian100_ope
BuildSet
wujian100_open-hello_world
board
csi_core
csi_driver
libs
projects
examples
hello_world
configs
key_gpio_intr
LED
oled128_32
main.c
Project
all.c
main.c
ff.h
led.h
led.c
lib.c
#include "oled128_32.h"
#include "ff.h" /* Declarations of FatFs
FATFS FatFs; /* FatFs work area needed
FIL Fil; /* File object needed for
extern void mdelay(uint32_t ms);
void sd_test();
int main(void)
{
printf("Hello World!\n");
key_gpio_intr(PA8);
LED_Init();
OLED_SHOW();
sd_test();
while(1)
{
printf("Hello World!\n");
mdelay(500);
LED_OFF();
mdelay(500);
}
while里添加printf函数
串口输出会容易看一些

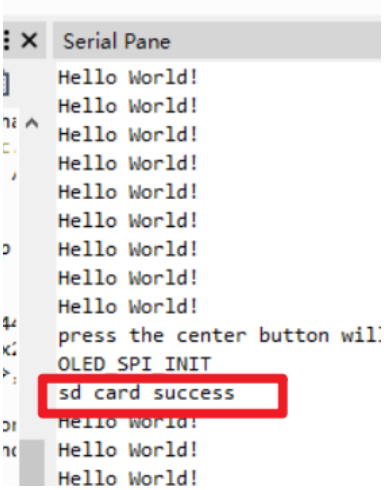

wujian100_ope
BuildSet
wujian100_open-h
board
csi_core
csi_driver
libs
projects
Options for "wujian100_open-hello
Packages Path Setting...
Switch SDK...
Create a new SDK...
Create a Common Package
Packages for "wujian100_open-hel
1.编译
Build Order...
Build
Rebuild
Clean
Build "wujian100_open-hello_world
rld] E:\software\SOC\F902\pro\Nexysvideo_wujian100-main\Nexysvideo_wujian
2.debug
ect Flash Debug Peripherals Tools Windows Help
Start/Stop Debugger
Start/Stop Debugger without Download to Flash
Ctrl-Shift

或者用这两个：


1.
2.

6. 复位后就会正确现象是灯闪烁，以及



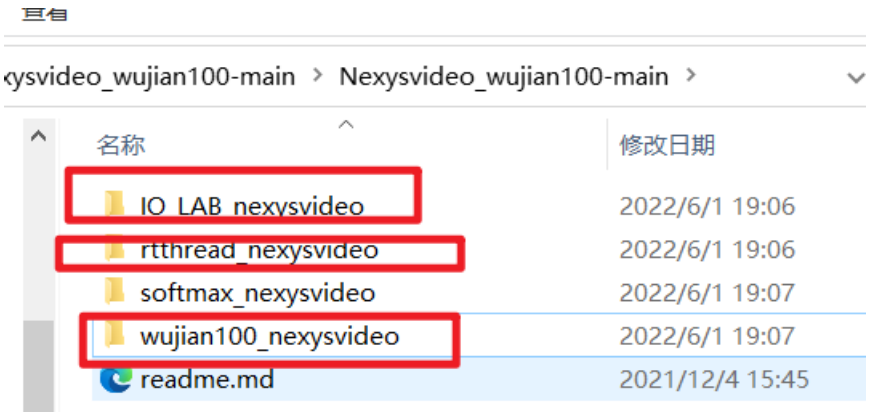


表明 SD 卡读取正确，可以在 Debug 调试界面查看寄存器中具体的值。

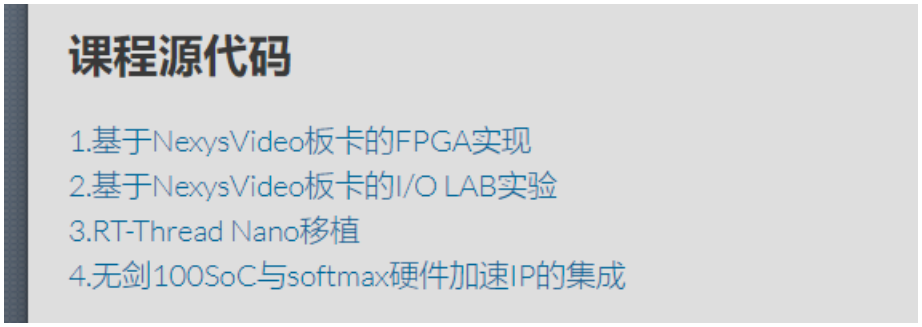
实验四：RT-thread Nano OS 移植

实验目标

RT-thread Nano 是一种实时的嵌入式操作系统。RTT 的代码也包括在从邸老师主页下载的 Nexysvideo_wujian100-main 文件里，这个文件里有着 IO 实验和 RTT 移植要用的所有代码，只需要在 CDK 里打开工程就行了。

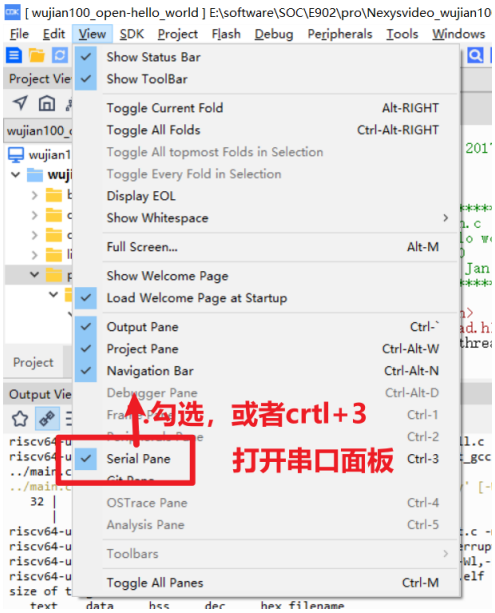


（邸老师主页：http://www.dizhixiong.cn/class5/）

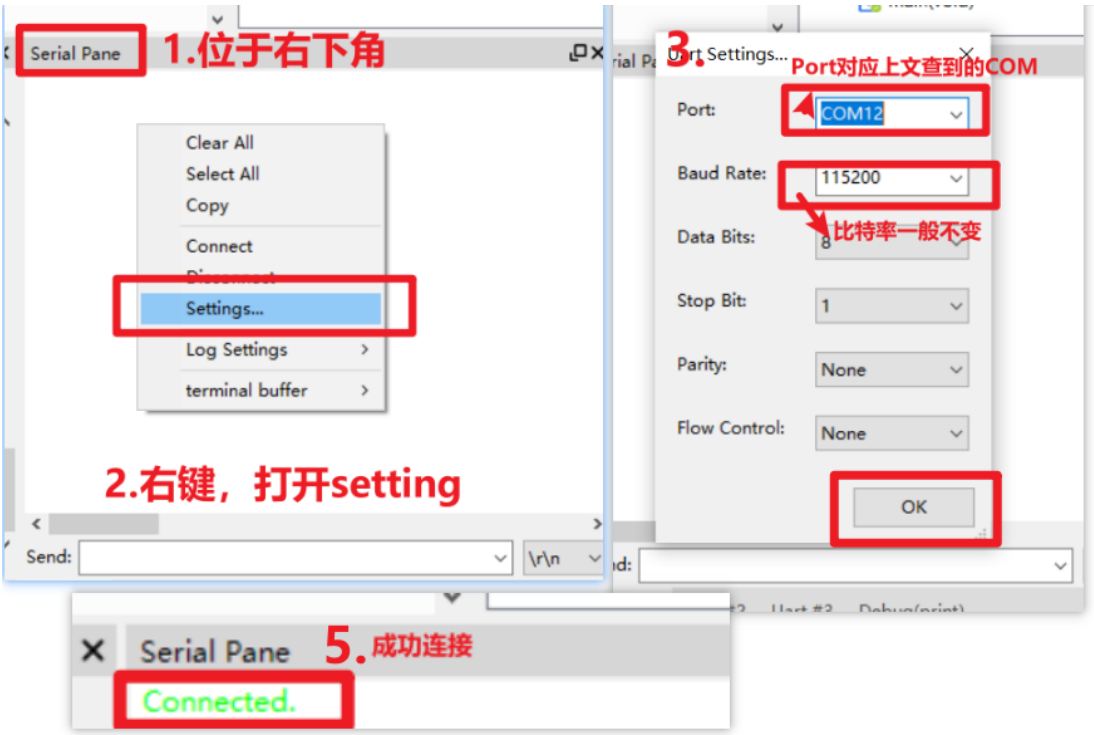


实验步骤

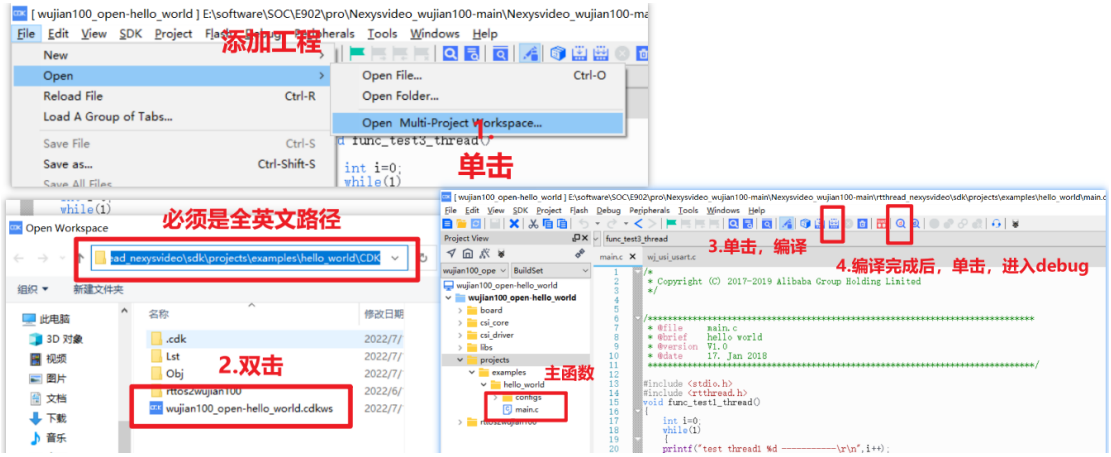
1、在 CDK 中打开串口面板

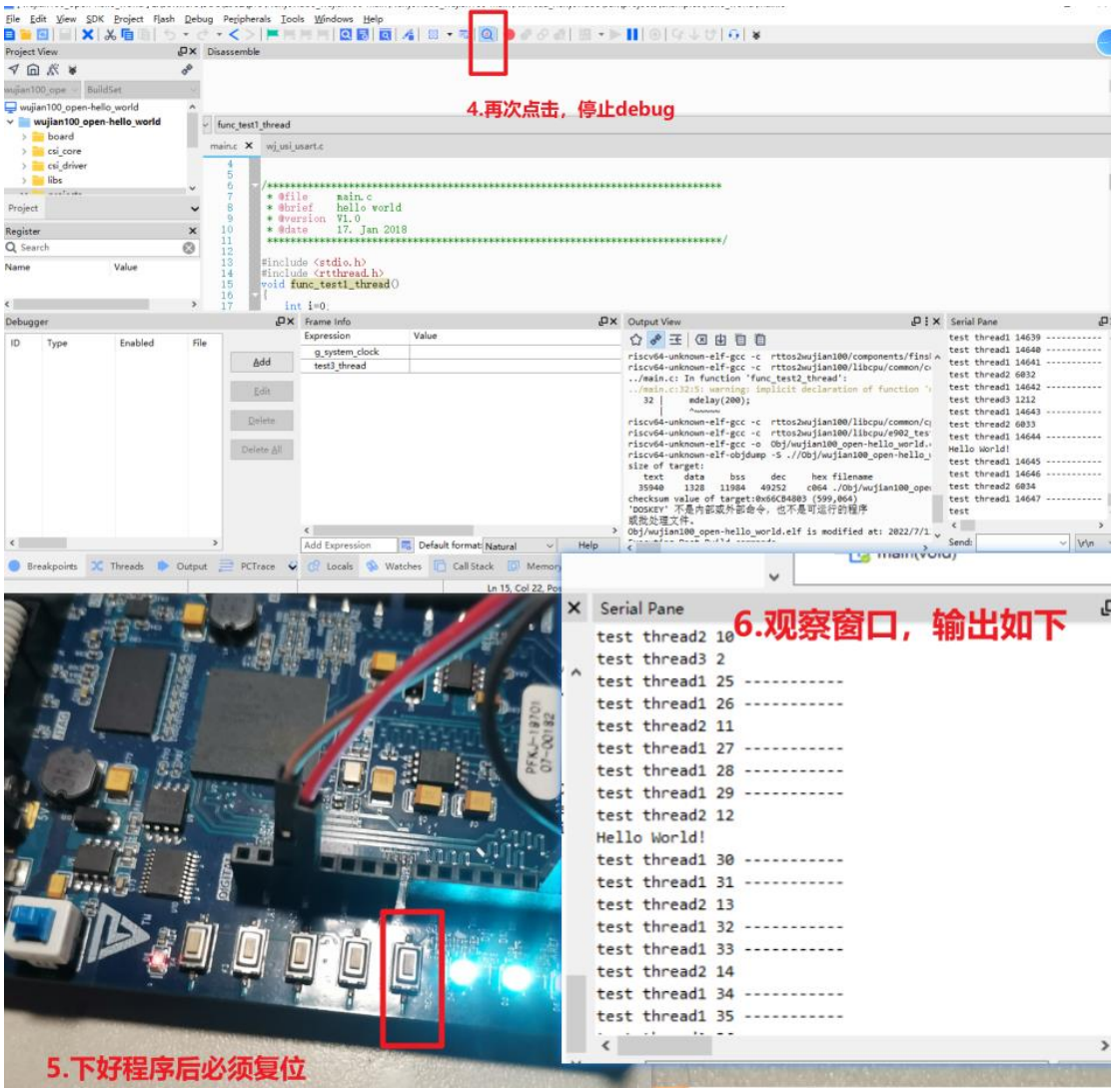


2、设置串口



3、添加针对 E902 处理器移植的 RTT nano 代码修改版

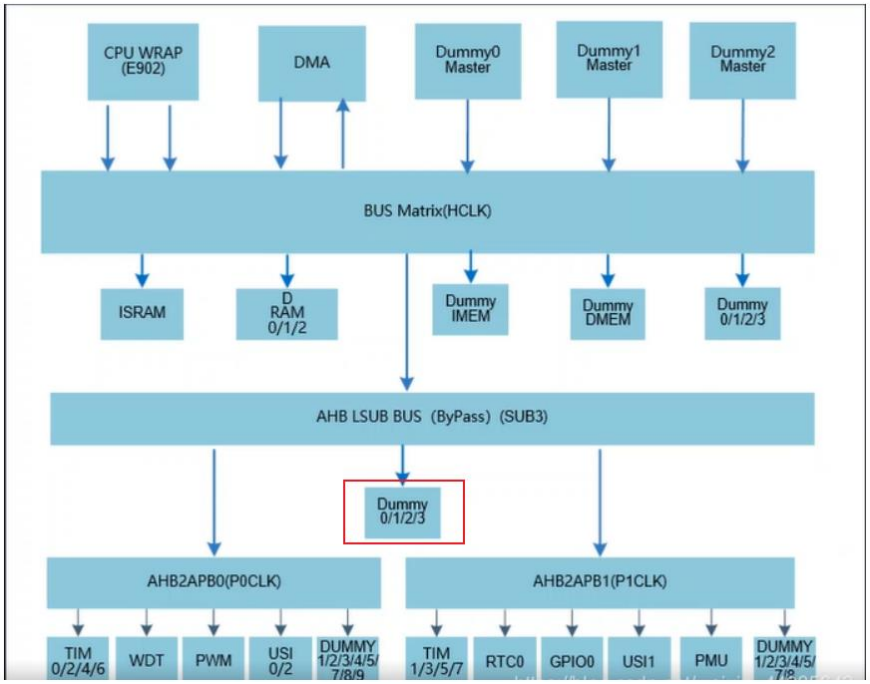

添加工程
单击
必须是全英文路径
2.双击
3.单击，编译
4.编译完成后，单击，进入debug
主函数


4.再次点击，停止debug
5.下好程序后必须复位
6.观察窗口，输出如下
Serial Pane
test thread2 10
test thread3 2
test thread1 25 ----------
test thread1 26 ----------
test thread2 11
test thread1 27 ----------
test thread1 28 ----------
test thread1 29 ----------
test thread2 12
Hello World!
test thread1 30 ----------
test thread1 31 ----------
test thread2 13
test thread1 32 ----------
test thread1 33 ----------
test thread2 14
test thread1 34 ----------
test thread1 35 ----------

实验五：在总线上挂载 IP

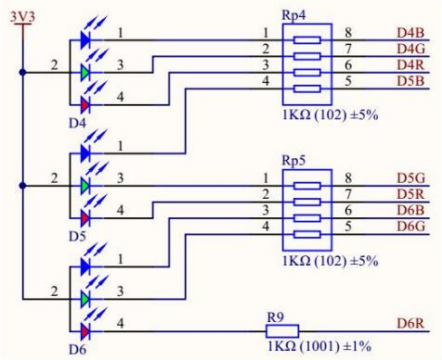
实验目标

Wujian100 中有留有许多 dummy 模块，可供用户自定义设计。本文档使用 AHB 总线上的 Dummy0 模块，通过写寄存器的方式控制 RGB LED 外设。



0x4000_0000~0x4000_3FFF	DMA	16K	S6	DMA Controller
0x4001_0000~0x4001_FFFF	Dummy	64K	S7	main_dummy_top0
0x4002_0000~0x4002_FFFF	Dummy	64K	S8	main_dummy_top1
0x4010_0000~0x401F_FFFF	Dummy	1M	S9	main_dummy_top2
0x4020_0000~0x7FFF_FFFF	LSBUS	1024M-2M	S10	AHB LS BUS
0x8000_0000~0x9FFF_FFFF	Dummy	512M	S11	main_dummy_top3
Other	REV	-	REV	Reserved

本次实验在 vivado 中建立自己的 IP，IP 为 AXI4 接口。基于三色 LED 灯的控制需要 9 位，这里用 32 位寄存器 0 的低 9 位作为控制输出。挂载 IP 到 wujian100、实现与验证功能。



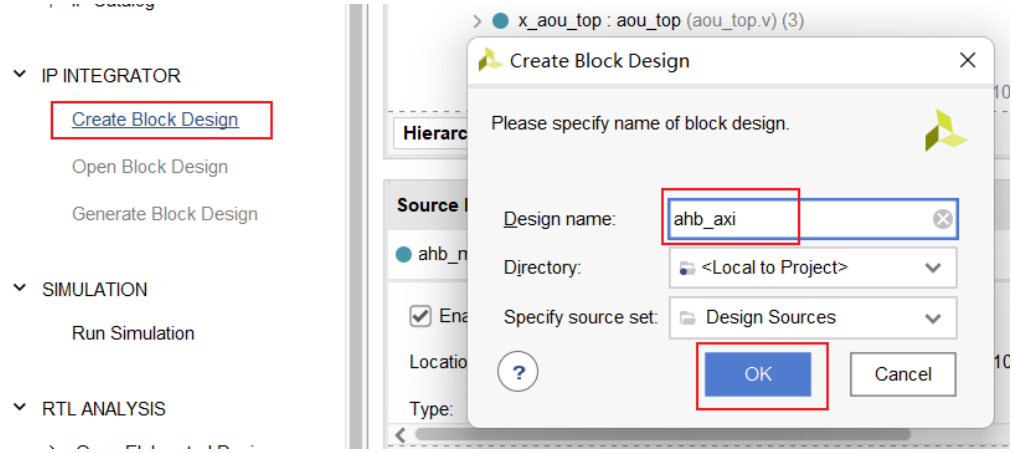
3 路用户 RGB 灯

实验步骤

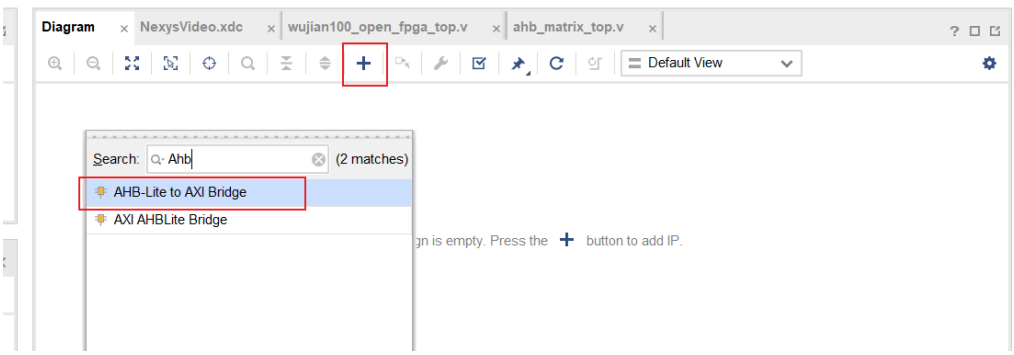
一、 建立 Block Design

Wujian100 软核为 AHB 总线，VIVADO 支持 AXI 总线 IP，需要用到 AHB-Lite to AXI Bridge 转接模块。

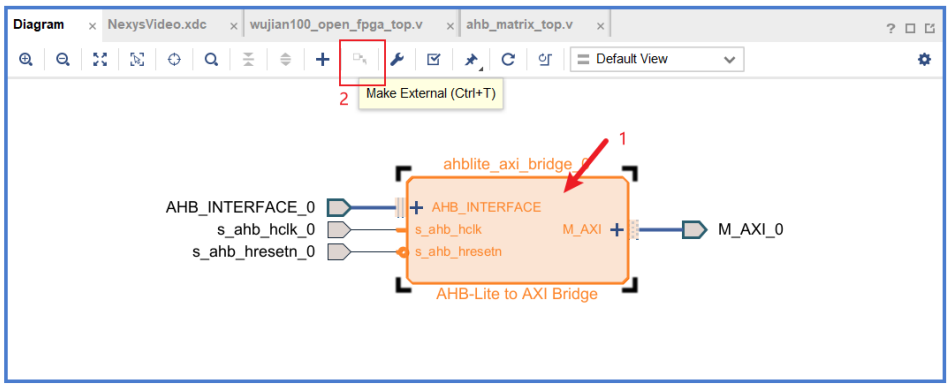
点击 Create Block Design，输入设计名，打开 Block Design 界面。



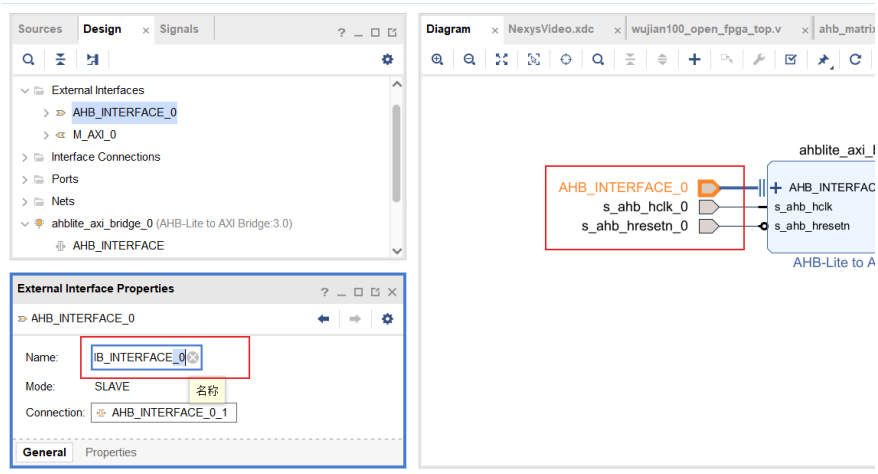
点击加号添加 IP，搜索 AHB-Lite to AXI Bridge，选择双击添加。



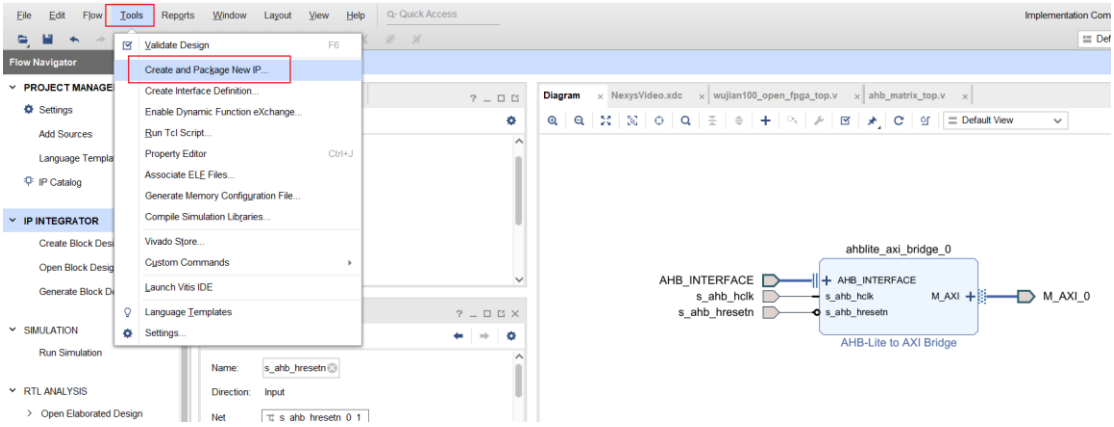
点击模块，按图标或 Ctrl+T 引出端口。该桥接模块作为 AXI 主机，自定义 IP 为 AXI 从机。



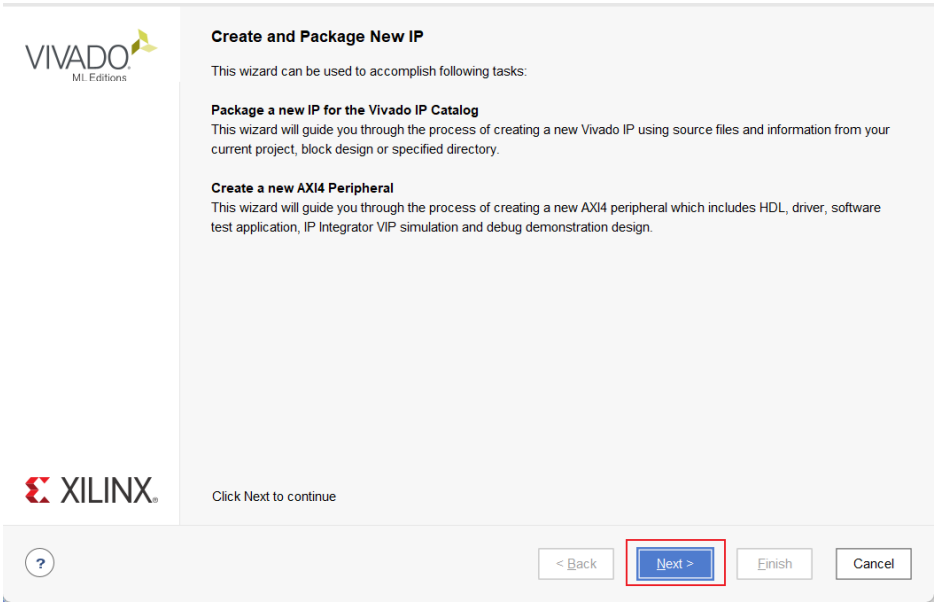
后面需要对源文件中的端口名进行修改，这里将后缀去掉，待会编辑的时候方便一点



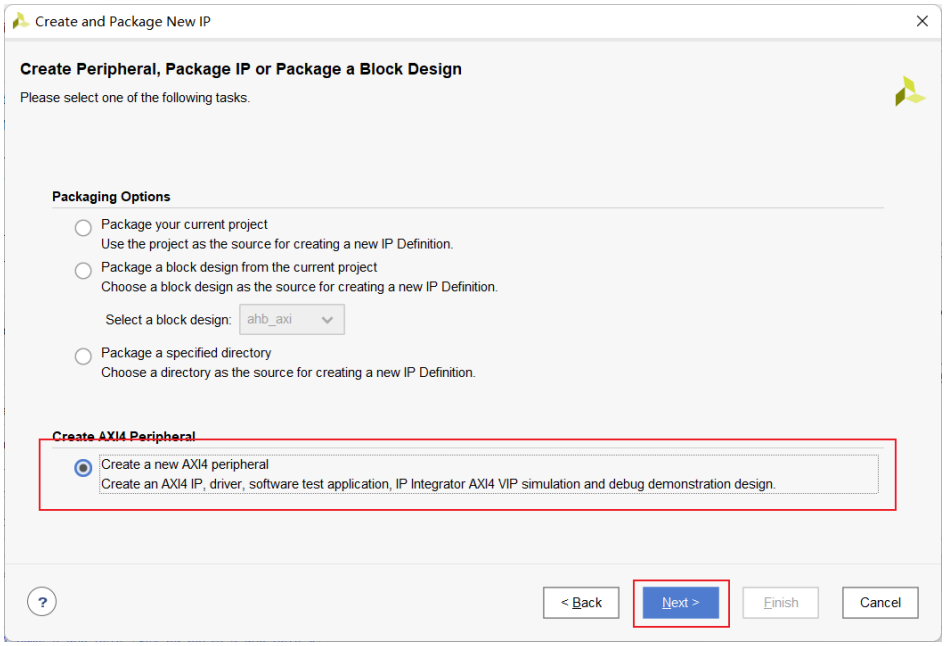
创建自己的 AXI IP 核，自定义 AXI IP。点击 Tools->Create and Package New IP。



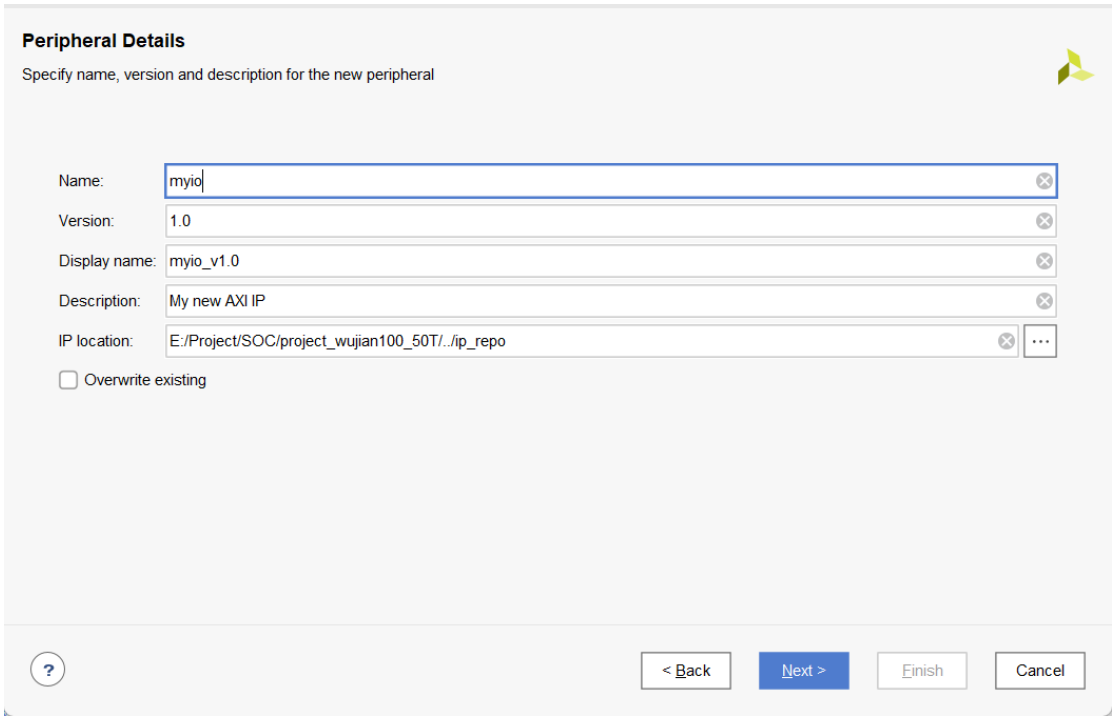
点击 Next。



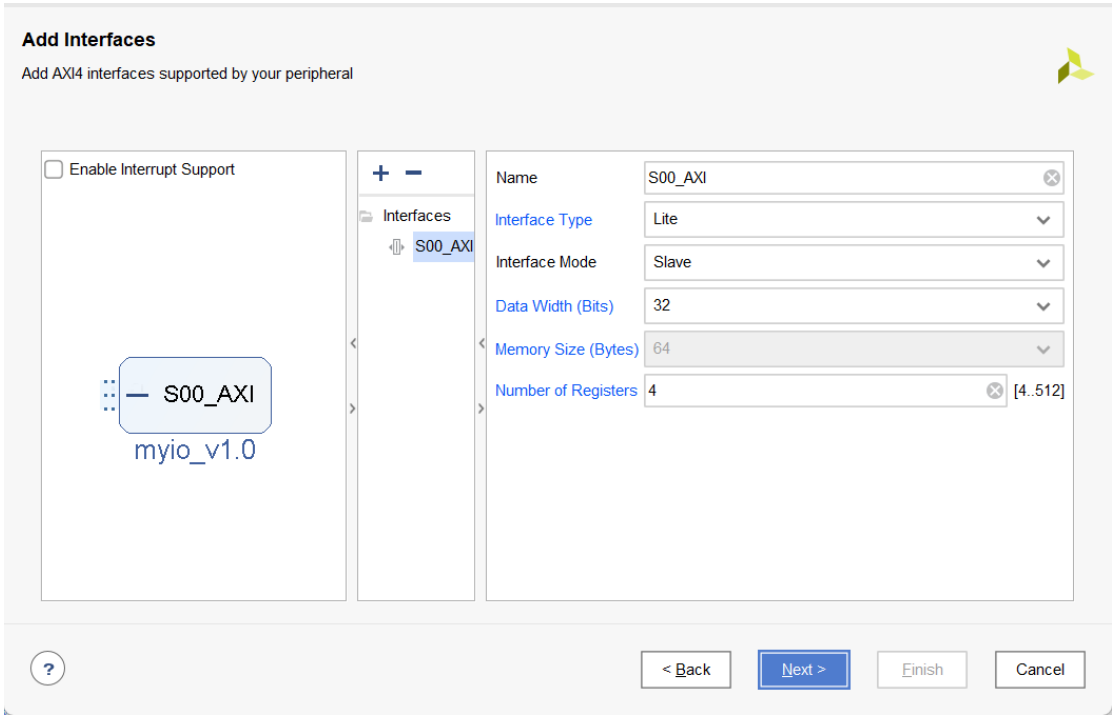
选择创建 AXI 外围的 IP，点击 Next。



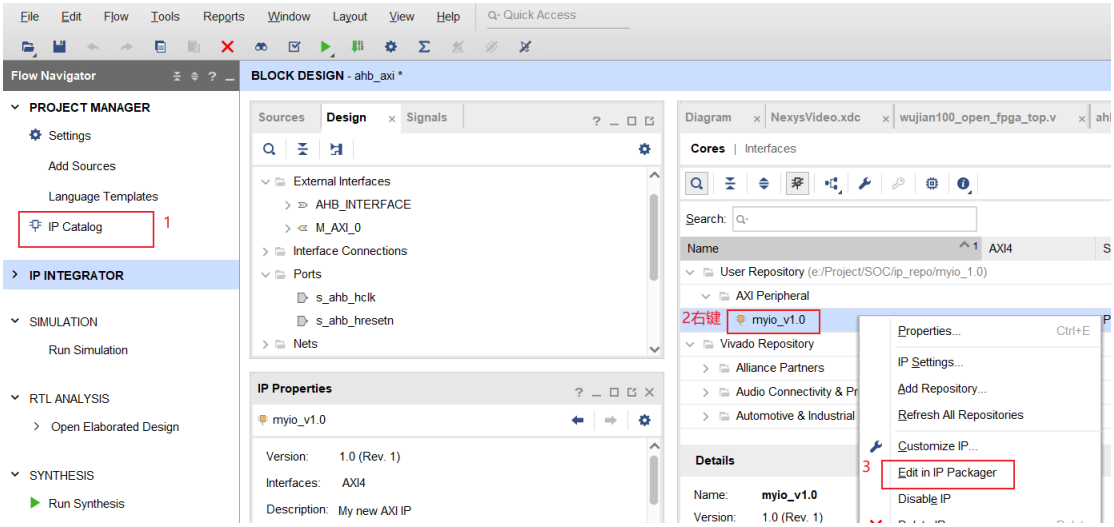
可对 IP 自己命名，写描述。



桥是 master，我们就创建 slave 的模式。如下图，点击 Next。



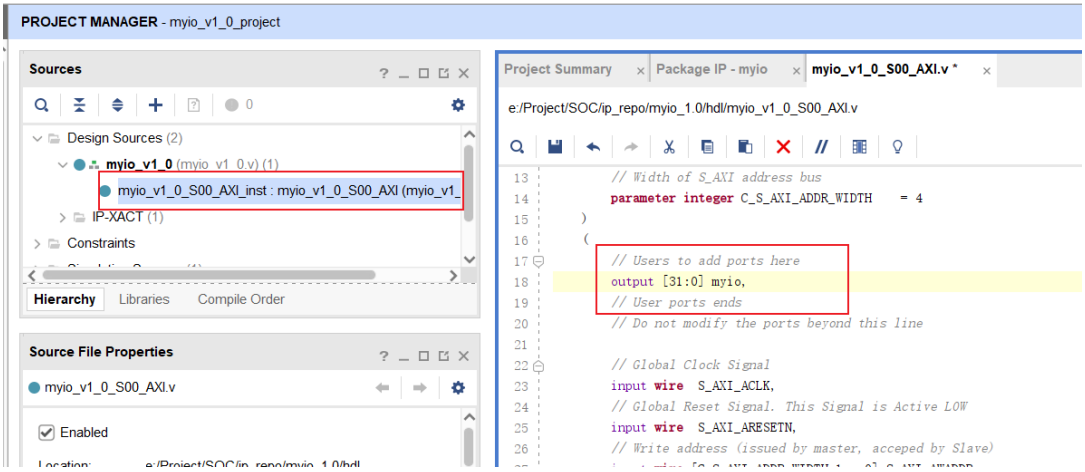
点击左侧 IP Catalog，打开 IP 管理器，看到刚才添加的 myio→ 右击 myio_v1.0 → Edit in IP Packager 选项，单击 OK，此时系统会自动打开另一个 Vivado IDE 来对用户 IP 核进行编辑。



在本模块中，设定有 4 个寄存器，每个寄存器位宽 32 位。这里通过往寄存器中写数据，来达到控制输出端口电平，从而控制三色灯的输出。将输出端口连接寄存器寄存器。

双击进入 myio_v1_0_S00_AXI，设计文件里面有用户代码区域。

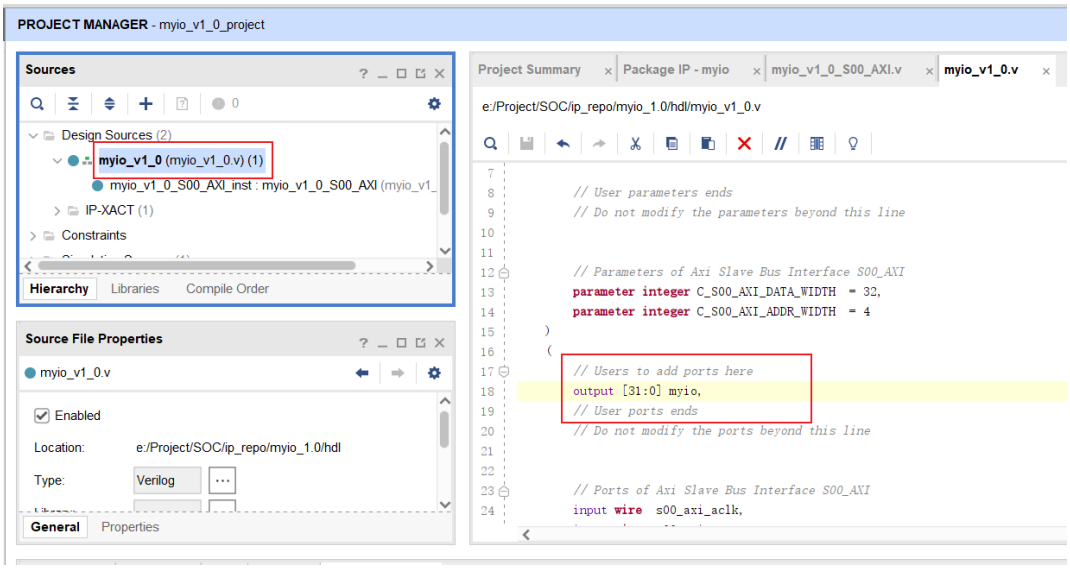
根据需求增加端口声明和添加用户信号。这里增加一个 output [31:0]myio 用于控制三色灯。



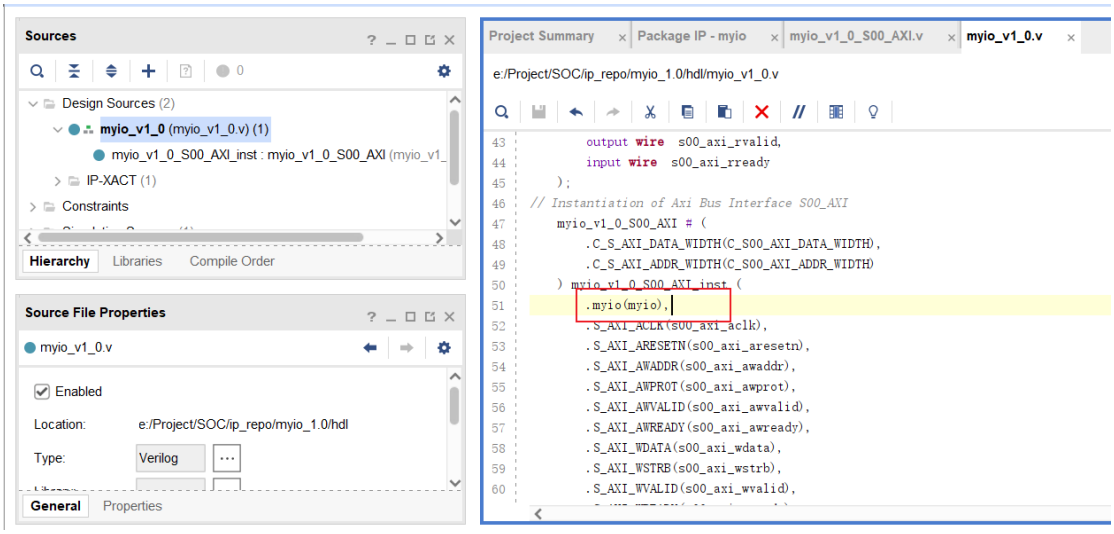
根据需求，添加用户逻辑代码。这里增加 assign myio = slv_reg0; 将输出连上寄存器 0。

```
400     // Add user logic here
401     assign myio = slv_reg0;
402     // User logic ends
```

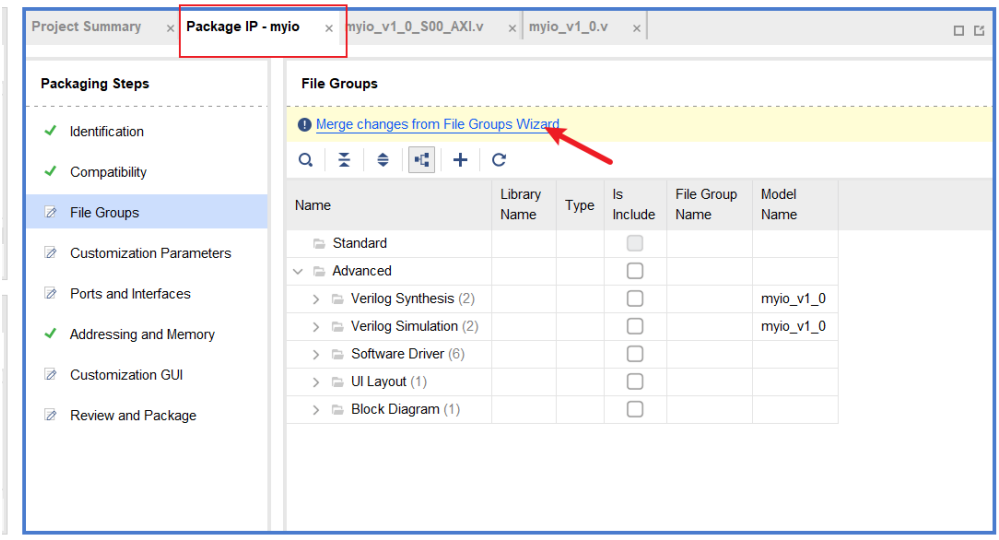
双击击进入 IP 的顶层，增加 output [31:0]myio

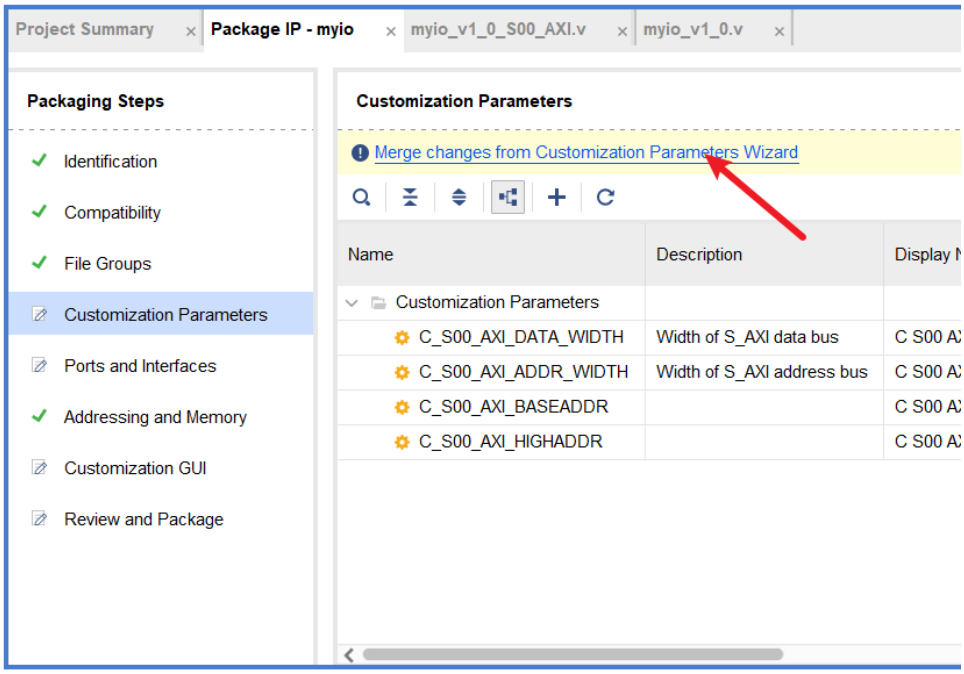


例化中增加.myio(myio),

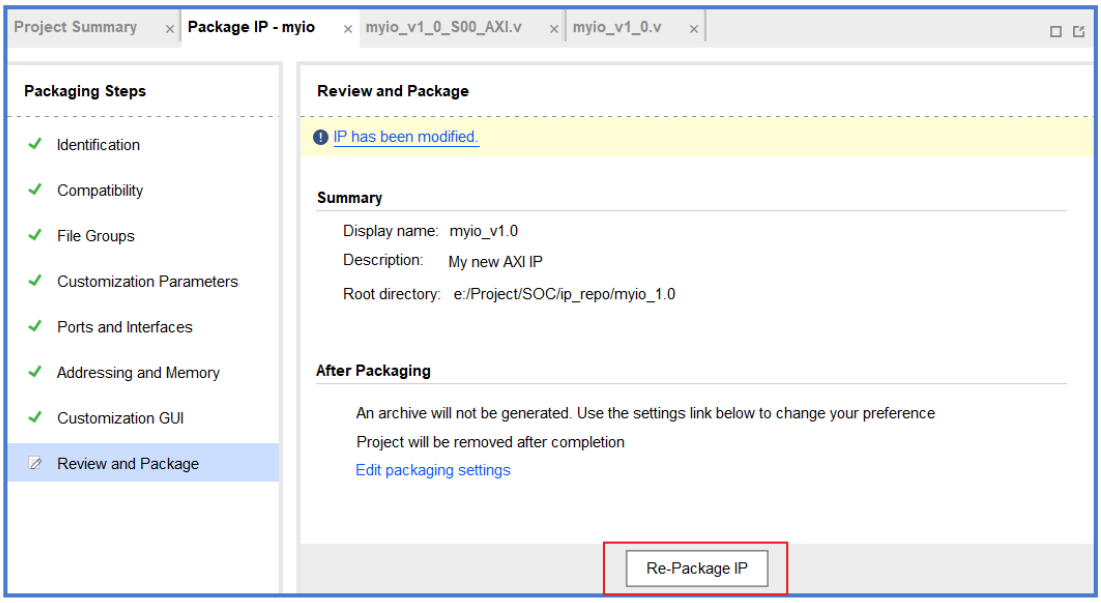


将更改刷新。切换到 Package IP-myio 窗口，单击如图所示链接，对刚才修改过的顶层文件进行更新。

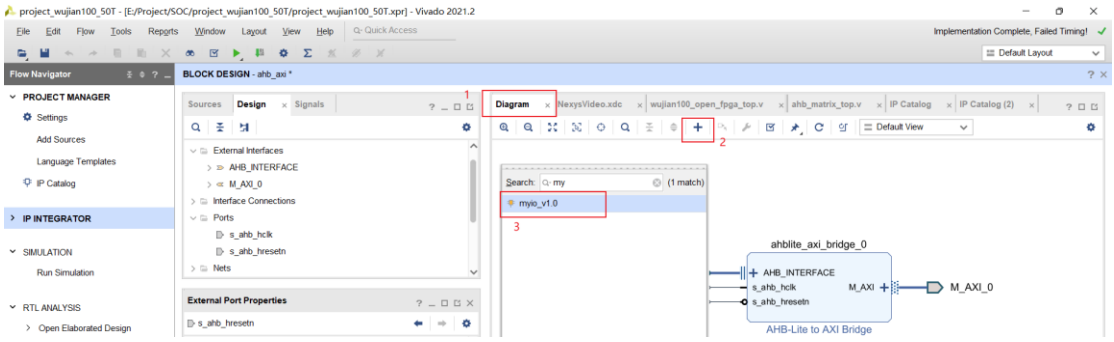

Project Summary
Package IP - myio
myio_v1_0_S00_AXI.v
myio_v1_0.v
Packaging Steps
Identification
Compatibility
File Groups
Customization Parameters
Ports and Interfaces
Addressing and Memory
Customization GUI
Review and Package
File Groups
Merge changes from File Groups Wizard
Name
Library Name
Type
Is Include
File Group Name
Model Name
Standard
Advanced
Verilog Synthesis (2)
Verilog Simulation (2)
Software Driver (6)
UI Layout (1)
Block Diagram (1)
myio_v1_0
myio_v1_0


Project Summary
Package IP - myio
myio_v1_0_S00_AXI.v
myio_v1_0.v
Packaging Steps
Identification
Compatibility
File Groups
Customization Parameters
Ports and Interfaces
Addressing and Memory
Customization GUI
Review and Package
Customization Parameters
Merge changes from Customization Parameters Wizard
Name
Description
Customization Parameters
C_S00_AXI_DATA_WIDTH
Width of S_AXI data bus
C_S00_AXI_ADDR_WIDTH
Width of S_AXI address bus
C_S00_AXI_BASEADDR
C_S00_AXI_HIGHADDR

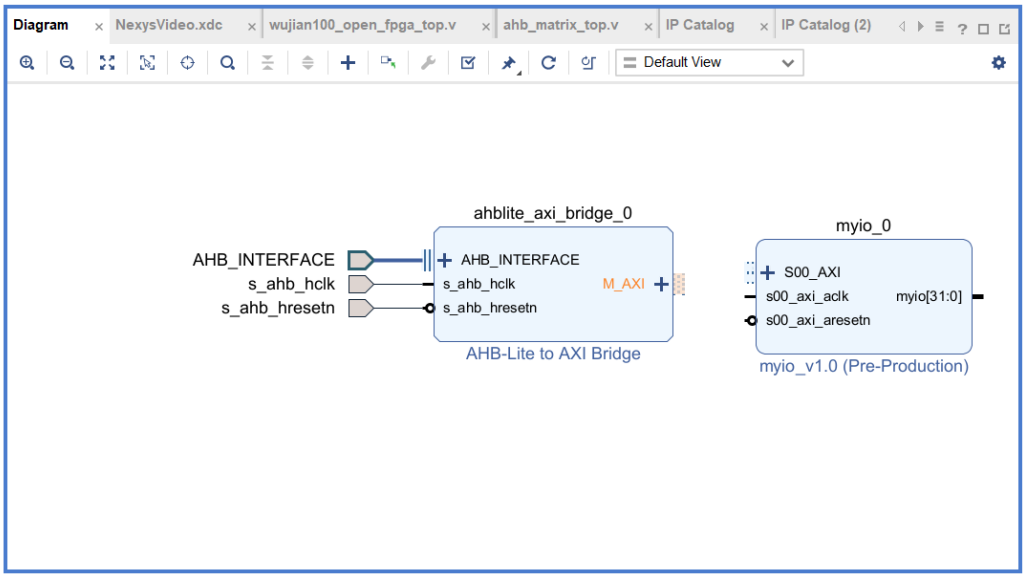
重新打包一下 IP



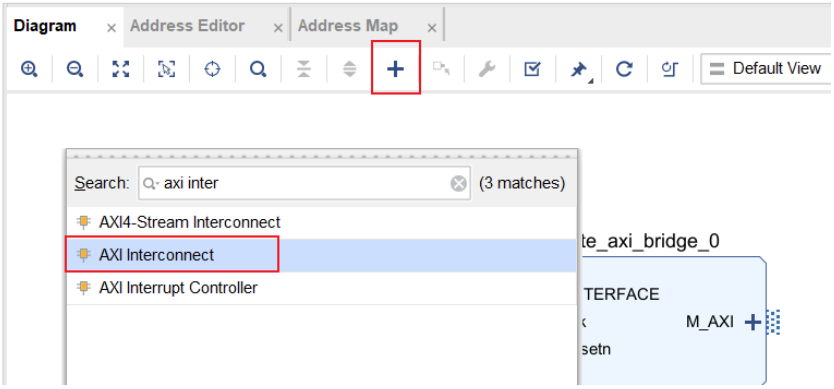
现在可以使用这个 IP 了，将 IP 添加进 Block Design。



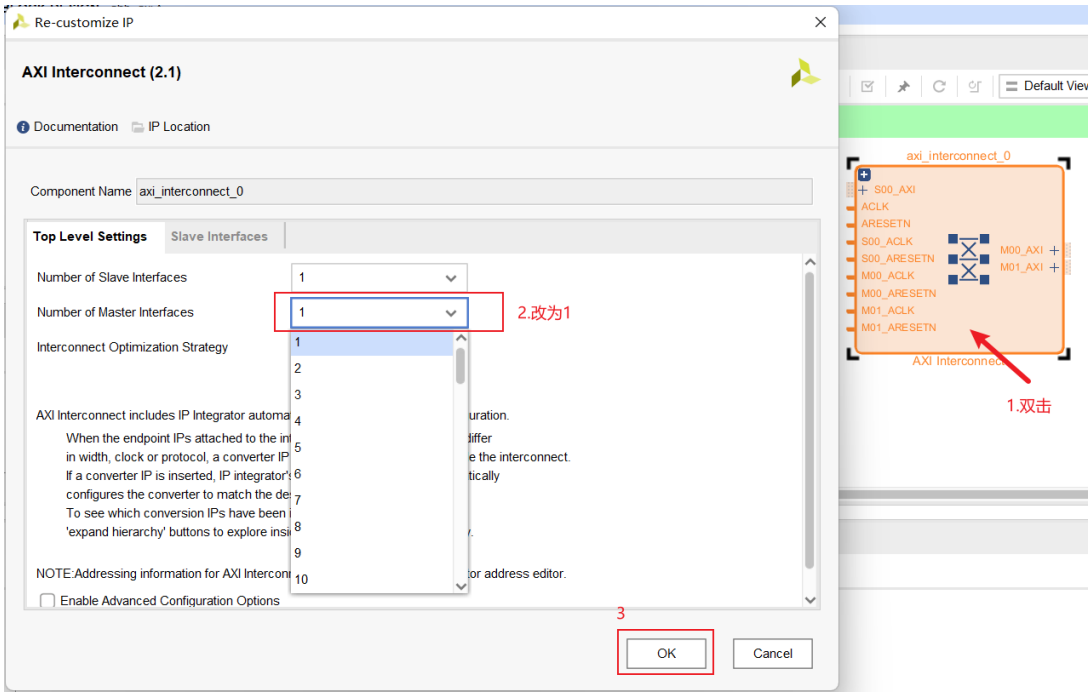
但它们的 AXI port 仍然有点差异，连接不上，需要加一个互联的 IP



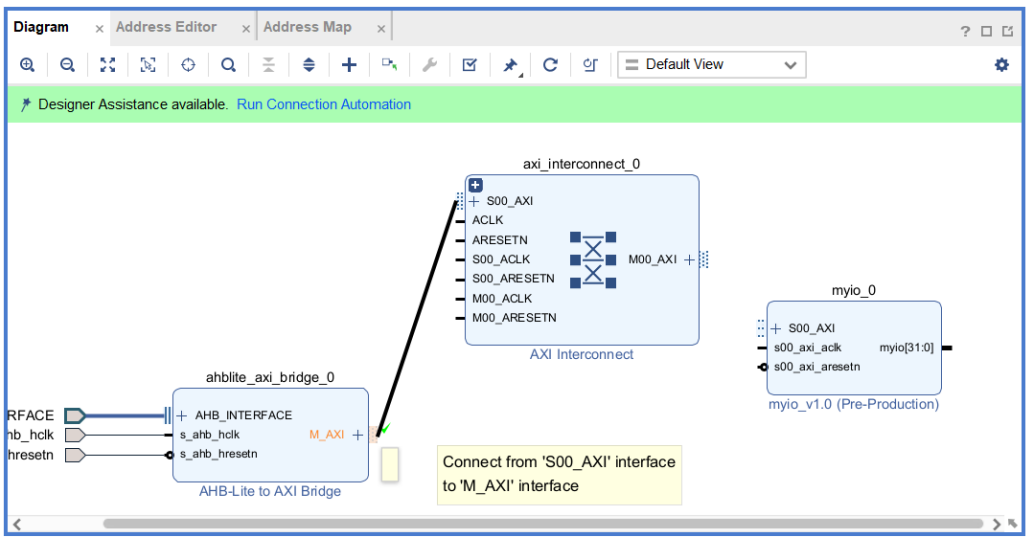
添加 AXI Interconnect 模块



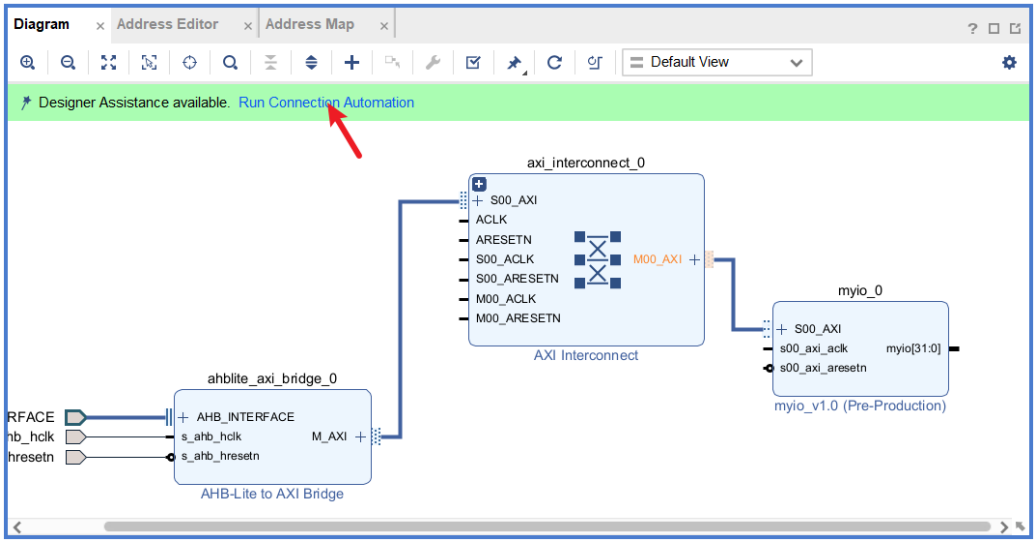
双击打开 AXI Interconnect 模块，Slave 和 Master 的接口都为 1。



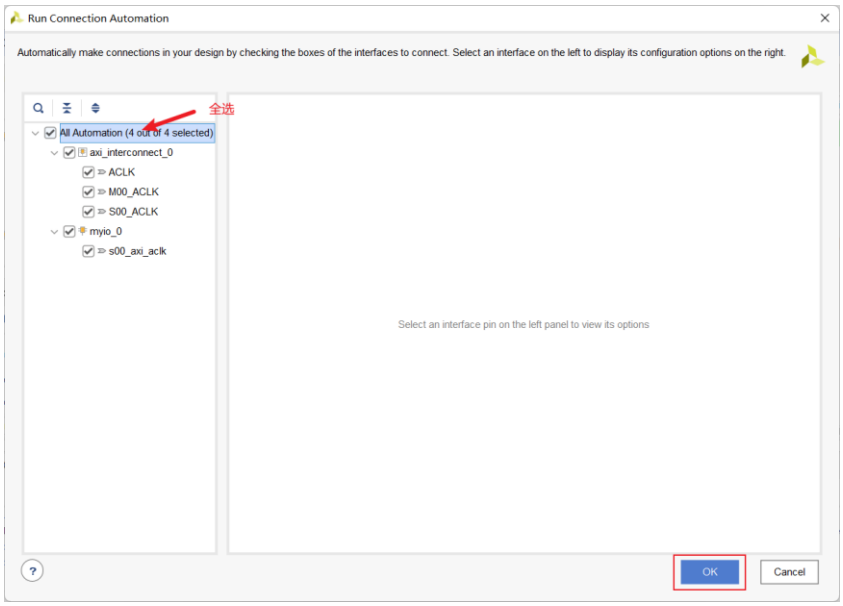
可以鼠标拖拽，连接端口



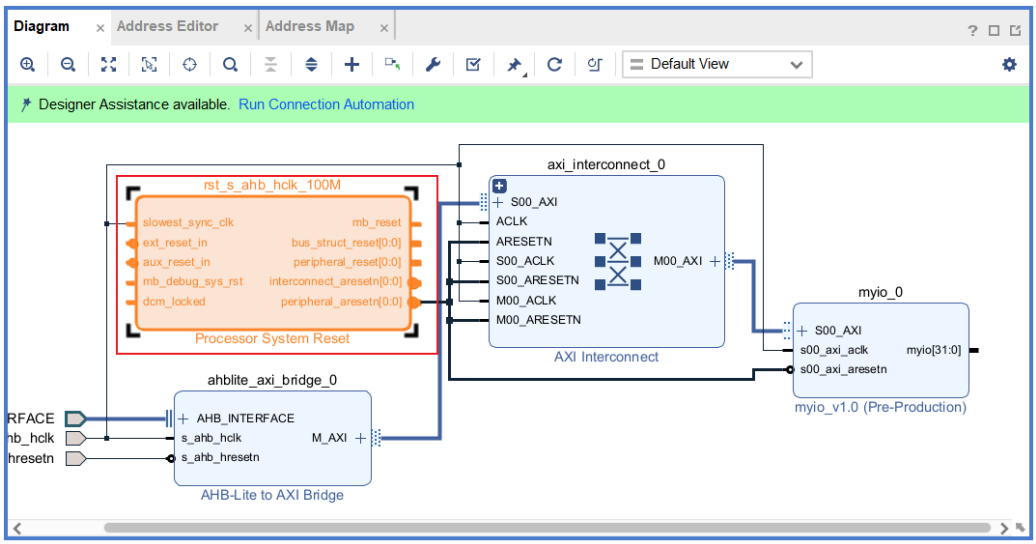
直接点击自动连接



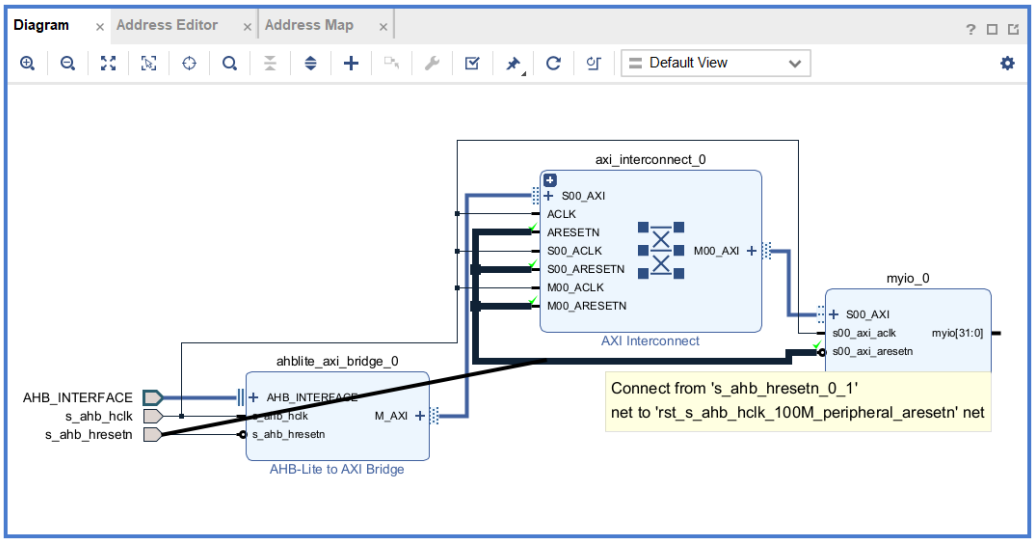
自动连接所以，点击确定。



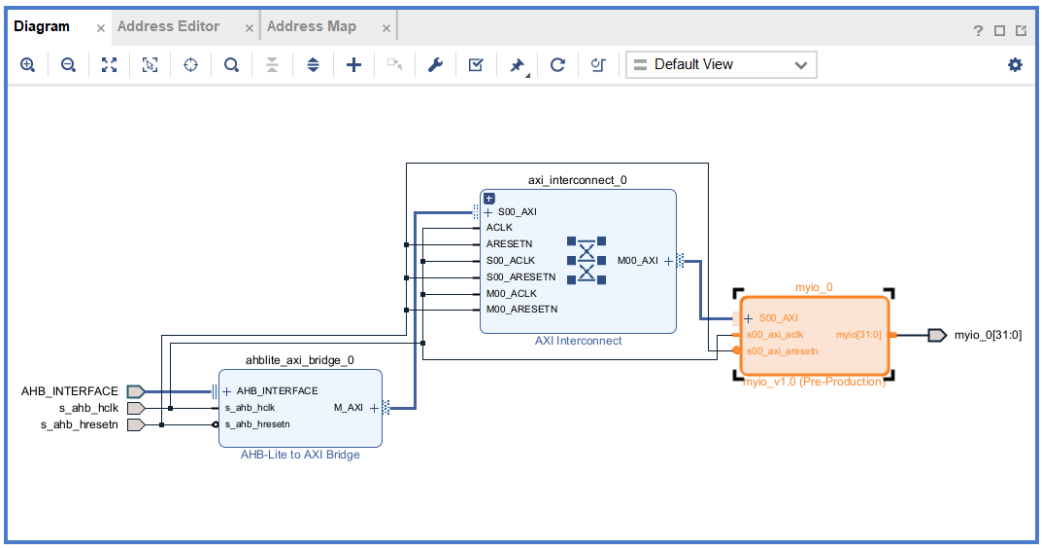
在自动布线后，出现了一个新的模块连接 rst 信号，这里把它删掉，自己手动连接。



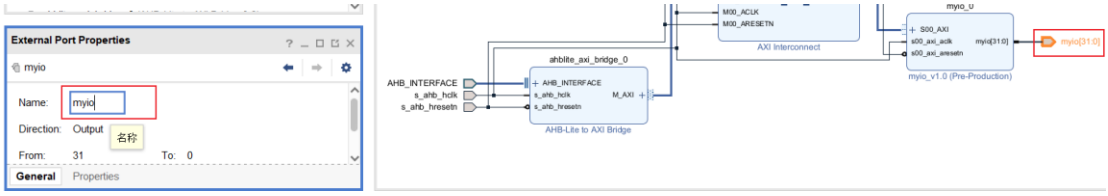
将 reset 信号手动连好。



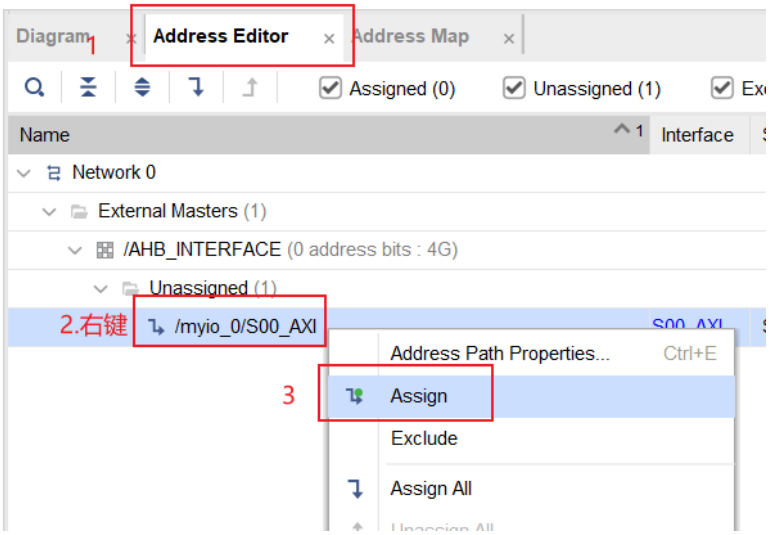
选中 myio 模块，ctrl+t 引出输出



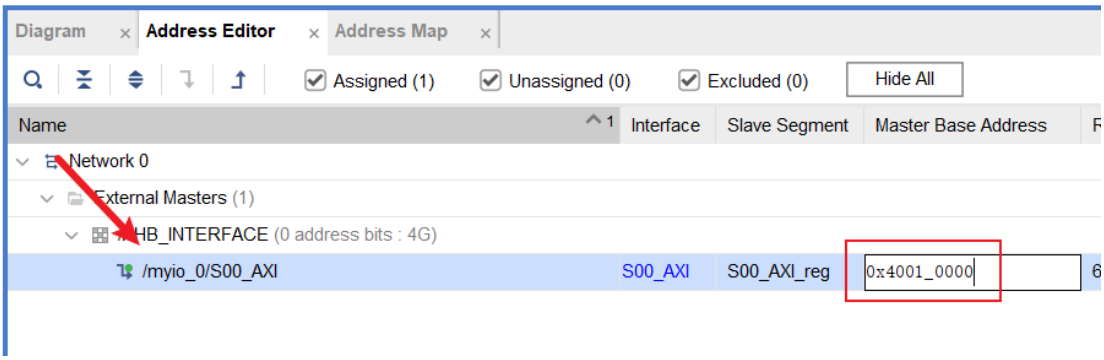
与之前一样修改名称，改为 myio



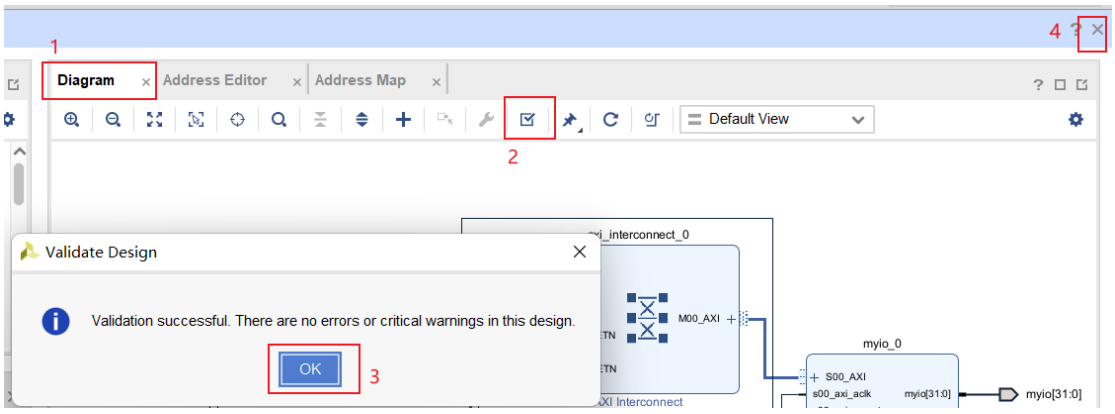
分配地址。去 Adress Editor 界面，找到 myio_0，右键点击 Assign。



将地址修改为 dummy0 的首地址 0x4001_0000。

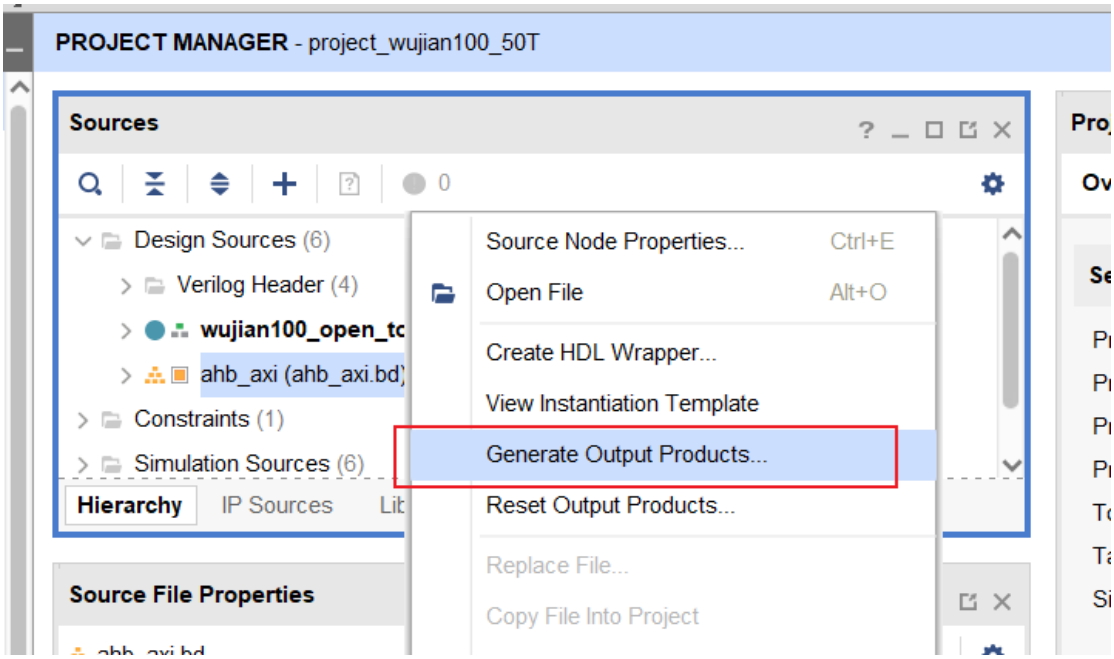


回到 diagram 窗口，点击验证一下，没有错误，退出 block design。

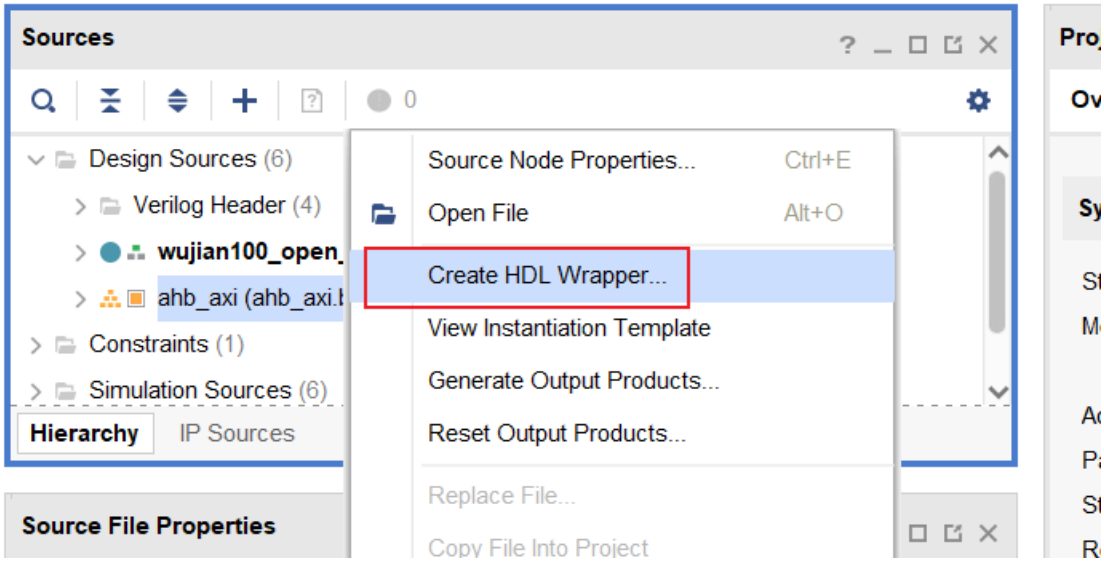


二、 挂 IP 到 wujian100

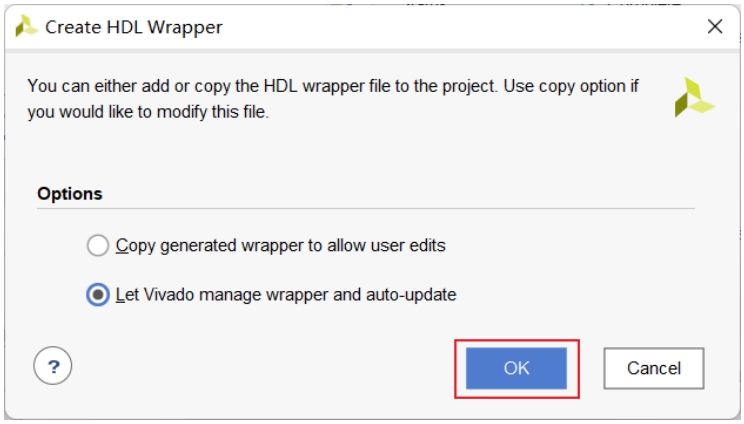
生成 IP 的 Wrapper。在 Source 里找到生成的 ahb_axi，右键 Generate Output Products



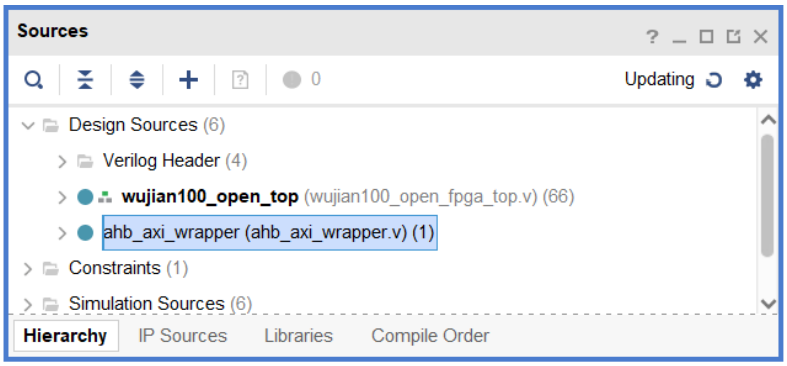
再右键点击 Create HDL Wrapper。



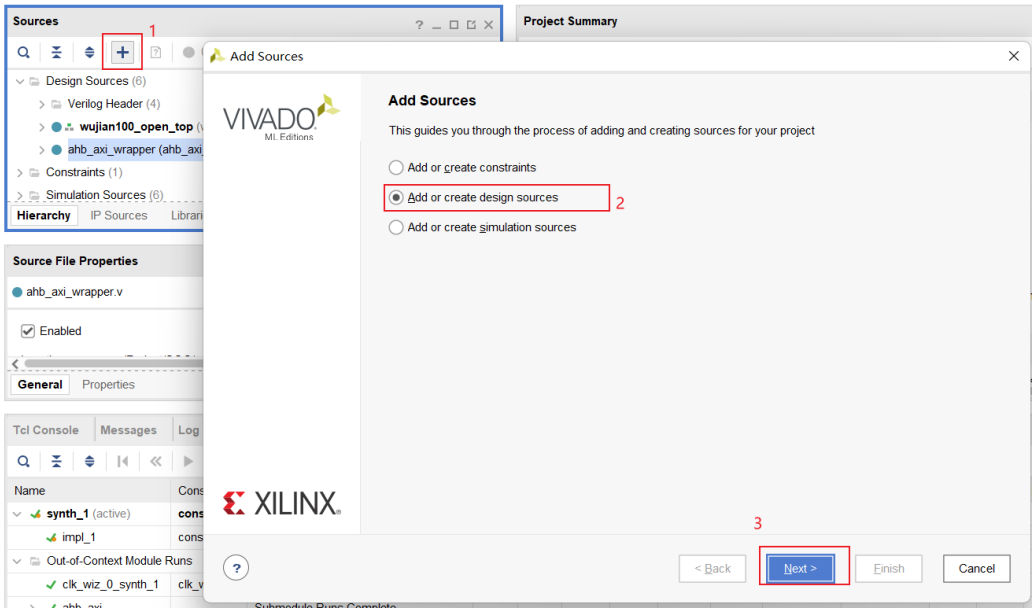
点击 OK。



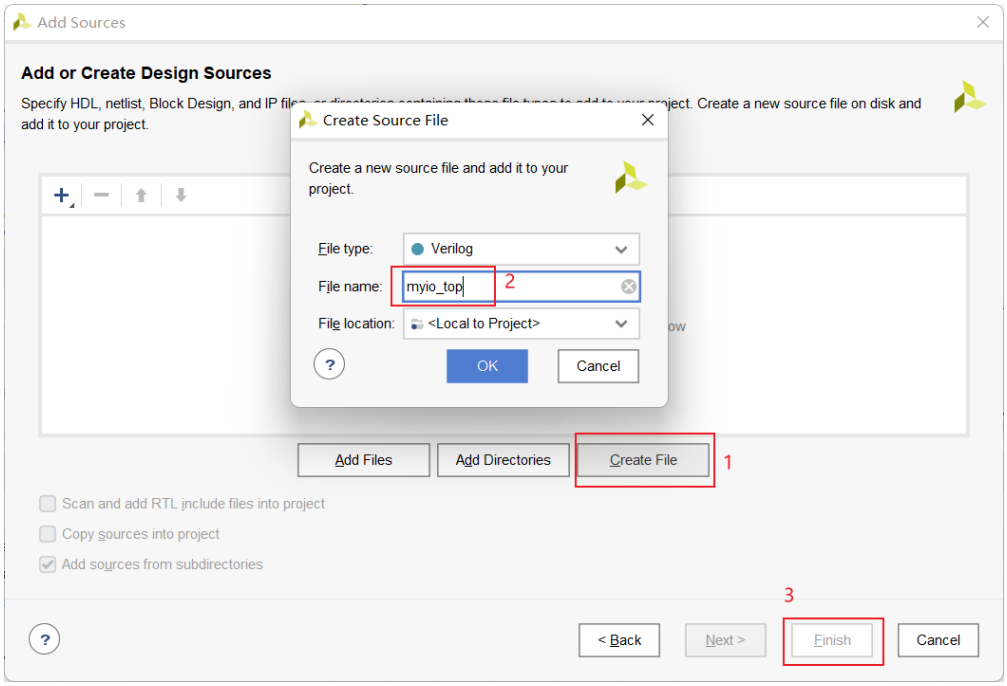
可以看到生成了 wrapper，可以被调用，或者再加一层文件进行调用。



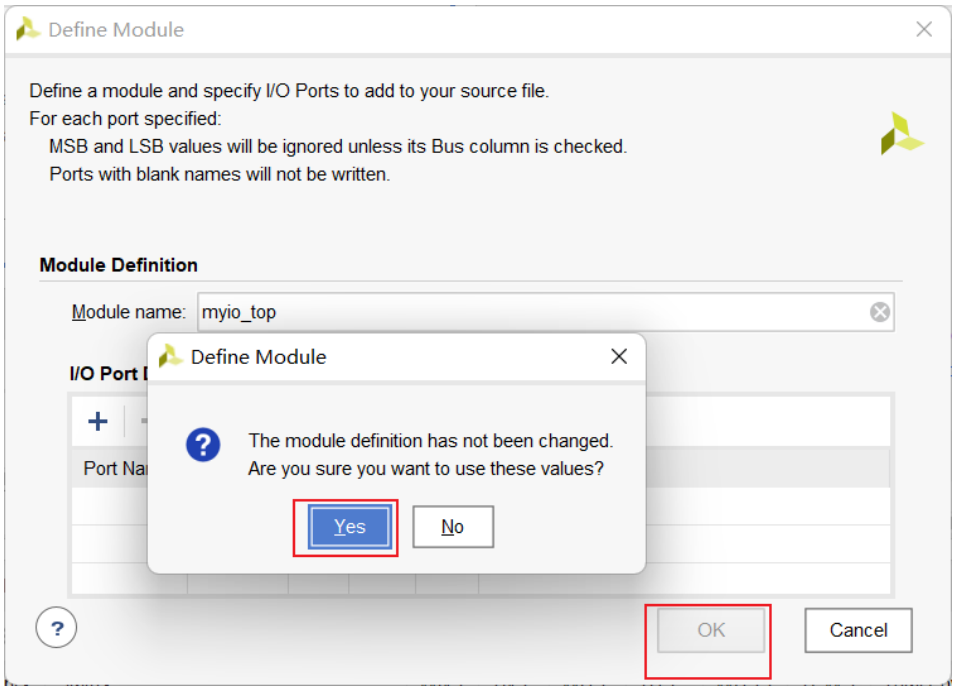
给自己的 design 加一层顶层文件。点击+号，新建设计源文件。



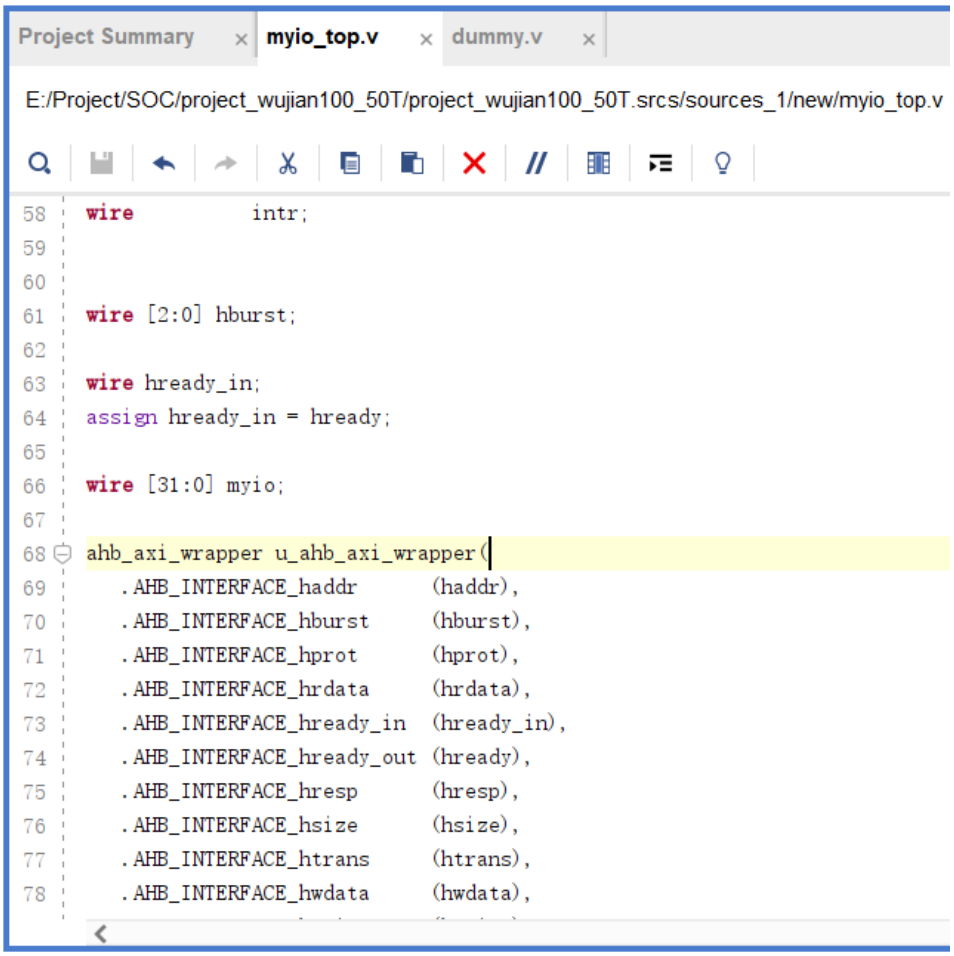
Create File。File name 为 myio_top。点击 Finish。



点击 OK，点击 Yes。



要将 main_dummy_top0 替换成 myio_top。在 myio_top 中例化 ahb_axi_wrappper，并添加与原 dummy 模块文件中相同的端口。



相关 RTL 代码：

```
module myio_top(haddr, hclk, hprot, hrdata, hready, hresp, hrst_b, hsel, hsize, htrans, hwdata,
hwrite, hburst,//新增
  intr,
  myio//新增
);
output [31:0] myio;
input   hburst;
input   [31:0]  haddr;
input          hclk;
input   [3 :0]  hprot;
input          hrst_b;
input          hsel;
input   [2 :0]  hsize;
```

```
input   [1 :0]  htrans;
input   [31:0]  hwdata;
input           hwrite;
output  [31:0]  hrdata;
output          hready;
output  [1 :0]  hresp;
output                intr;
wire  [31:0]  hrdata;
wire          hready;
wire  [1 :0]  hresp;
wire          intr;

wire [2:0] hburst;
wire hready_in;
assign hready_in = hready;
wire [31:0] myio;

ahb_axi_wrapper u_ahb_axi_wrapper(//例化 ahb_axi_wrapper 模块
  .AHB_INTERFACE_haddr      (haddr),
  .AHB_INTERFACE_hburst     (hburst),
  .AHB_INTERFACE_hprot      (hprot),
  .AHB_INTERFACE_hrdata     (hrdata),
  .AHB_INTERFACE_hready_in  (hready_in),
  .AHB_INTERFACE_hready_out (hready),
  .AHB_INTERFACE_hresp      (hresp),
  .AHB_INTERFACE_hsize      (hsize),
  .AHB_INTERFACE_htrans     (htrans),
  .AHB_INTERFACE_hwdata     (hwdata),
  .AHB_INTERFACE_hwrite     (hwrite),
  .AHB_INTERFACE_sel        (hsel),
```

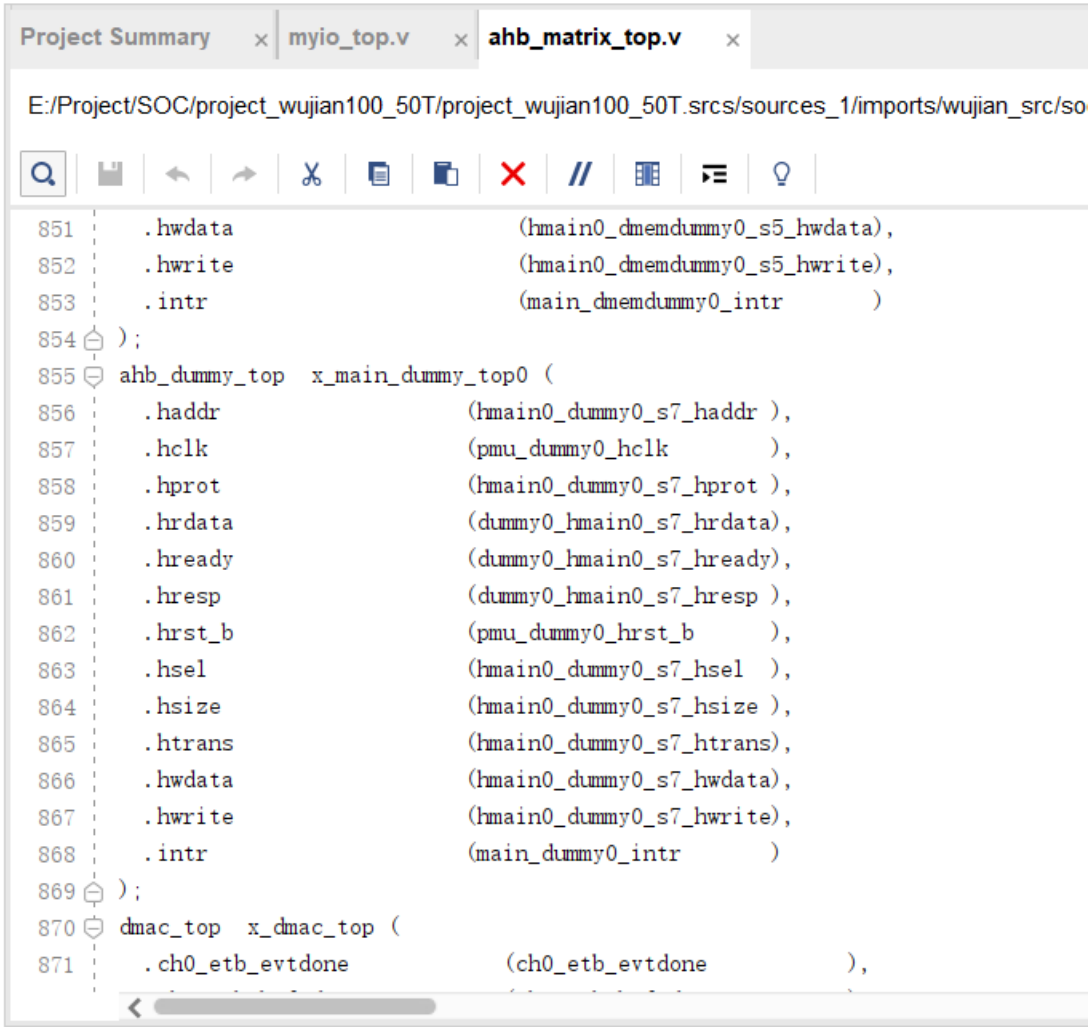
```
    .myio                (myio),
    .s_ahb_hclk          (hclk),
    .s_ahb_hresetn       (hrst_b)
);

endmodule
```

要将 main_dummy_top0 替换成 myio_top。依次修改 ahb_matrix_top，pdu_top，wujian100_open_top 中的例化。

双击打开修改 ahb_matrix_top 文件

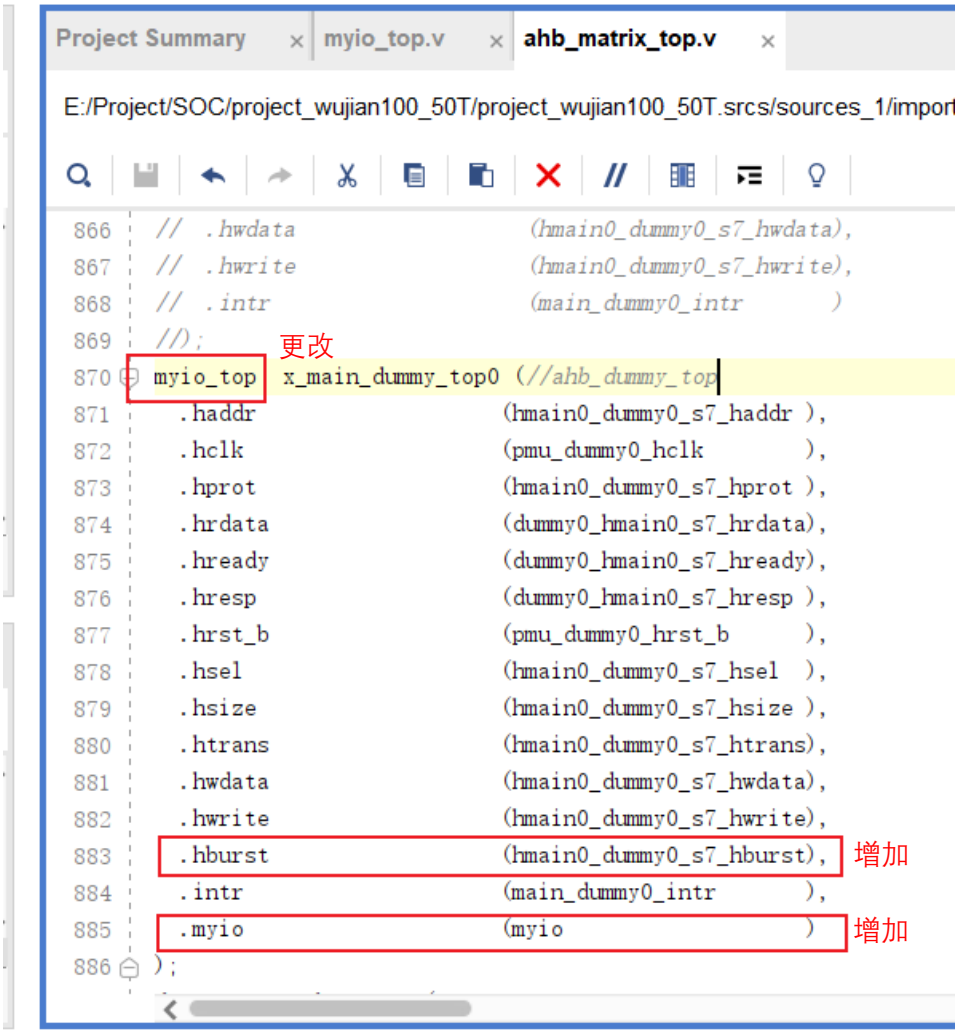
找到 x_main_dummy_top0 的例化



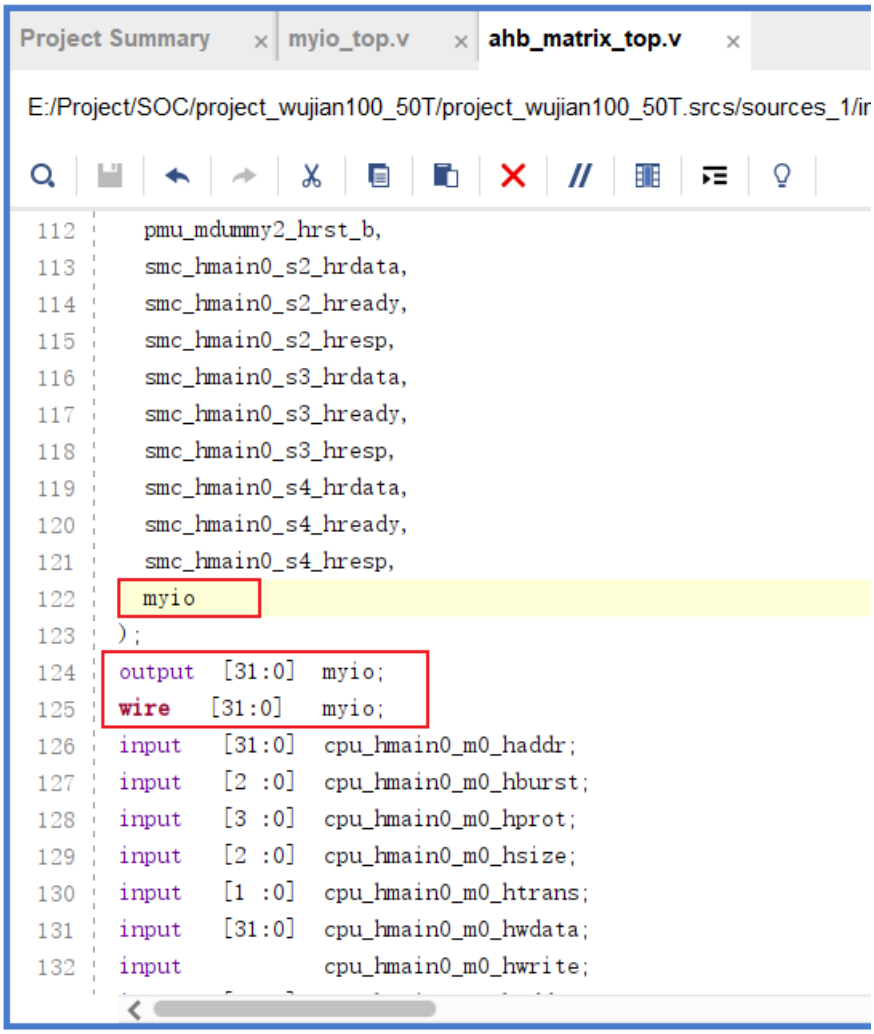
将例化更改为 myio_top，注意：

1.还要添加新增的 output 端口 myio，信号需要引出到最顶层，myio

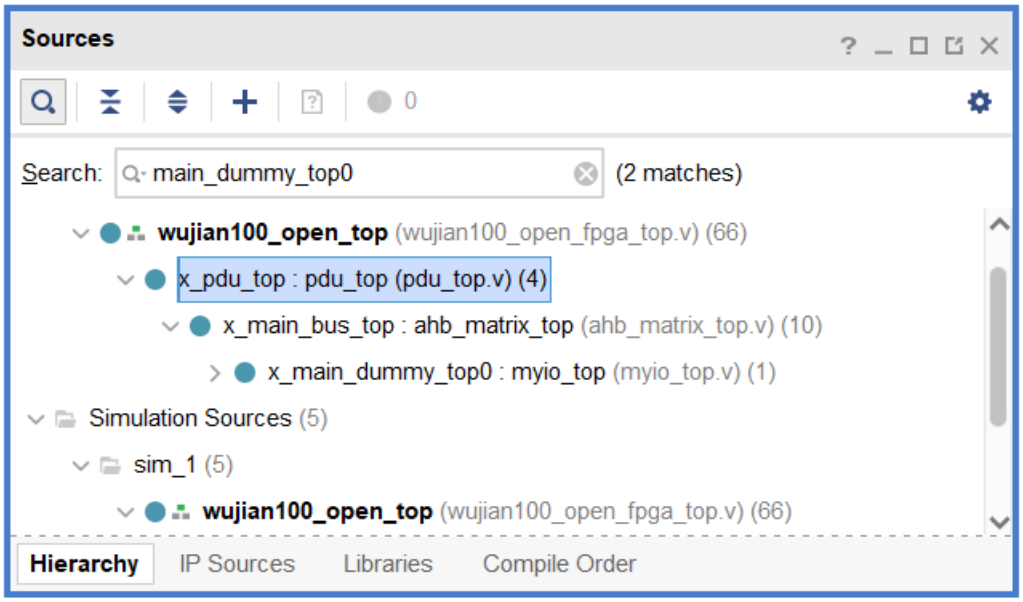
2.新增了 hburst,之前的 dummy 空模块中未使用，这里需要加上。



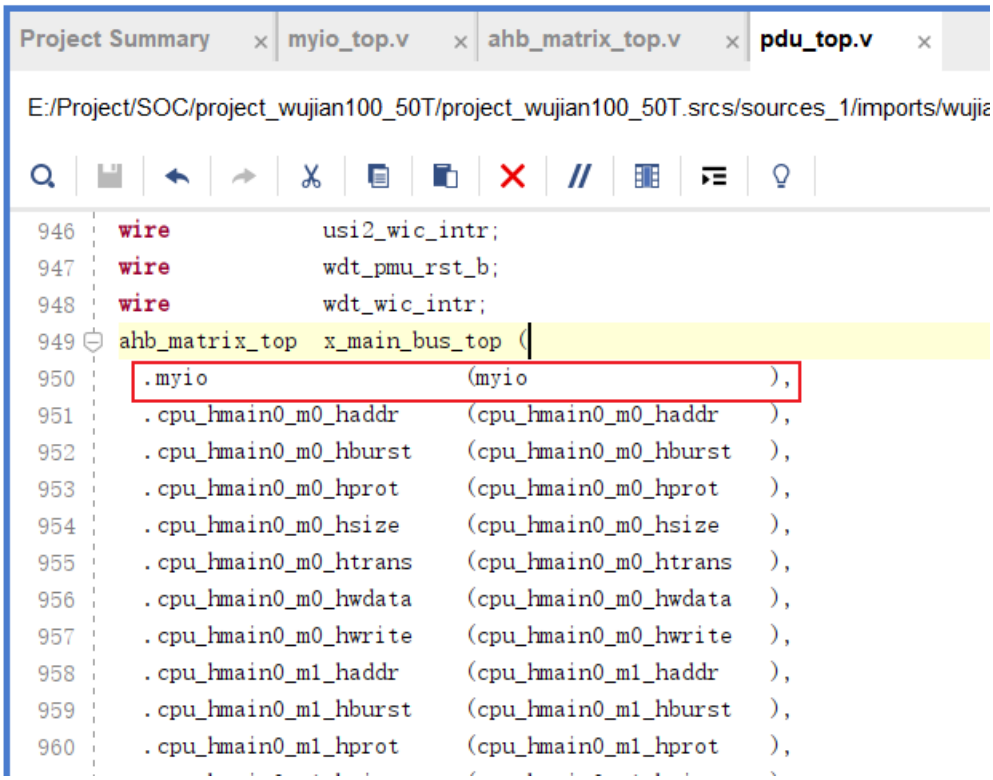
在改文件开始处，增加输出引脚和定义：



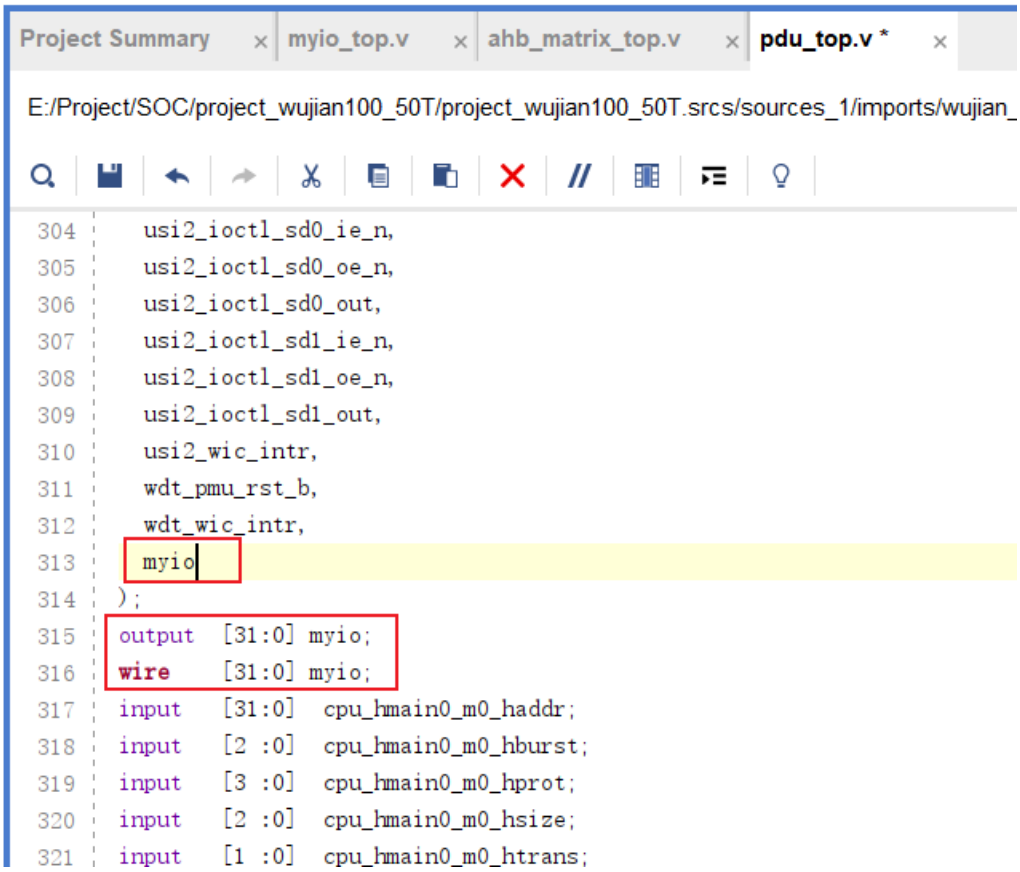
双击打开修改 pdu_top 文件



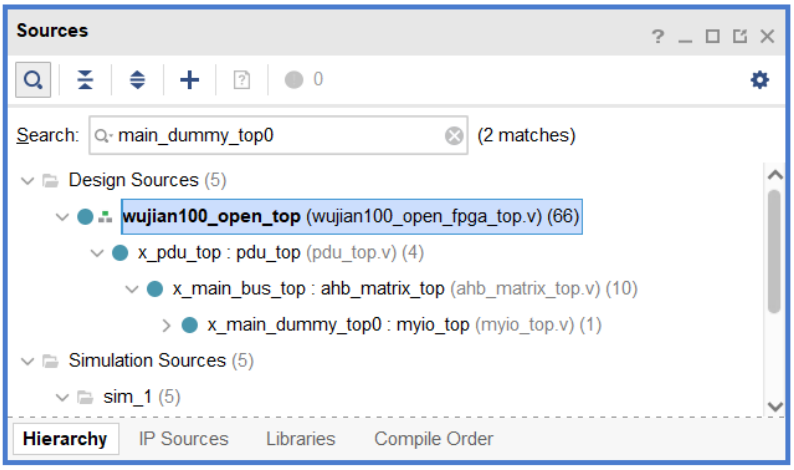
找到例化的 ahb_matrix_top，添加 myio 连接



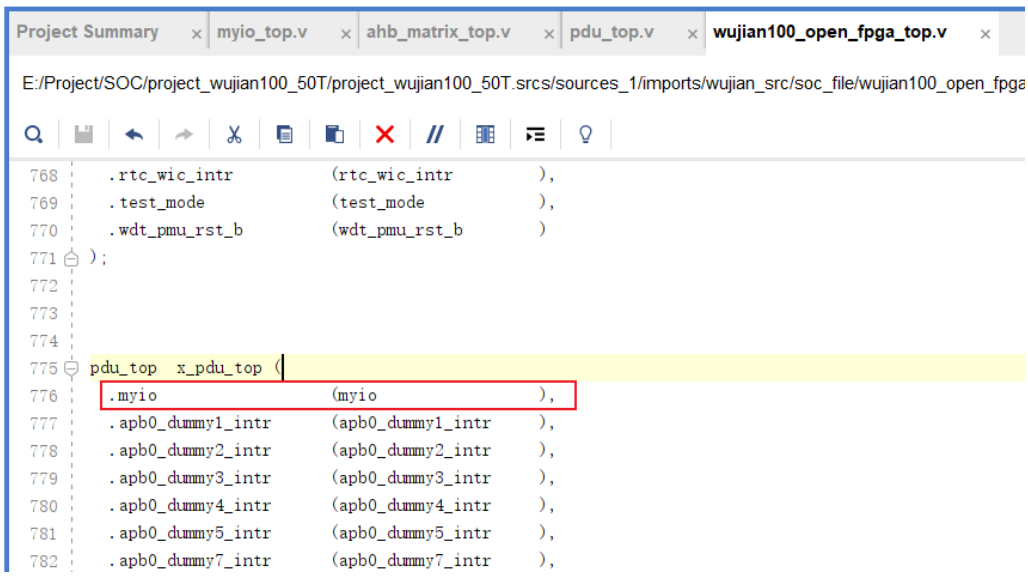
同样，增加输出引脚和定义



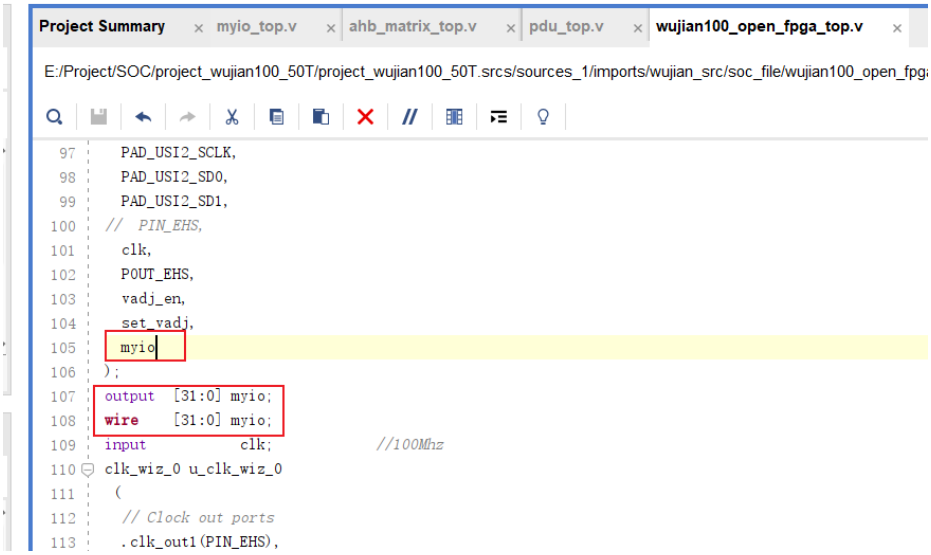
双击打开修改 wujian100_open_top 文件



找到例化的 pdu_top，添加 myio 连接

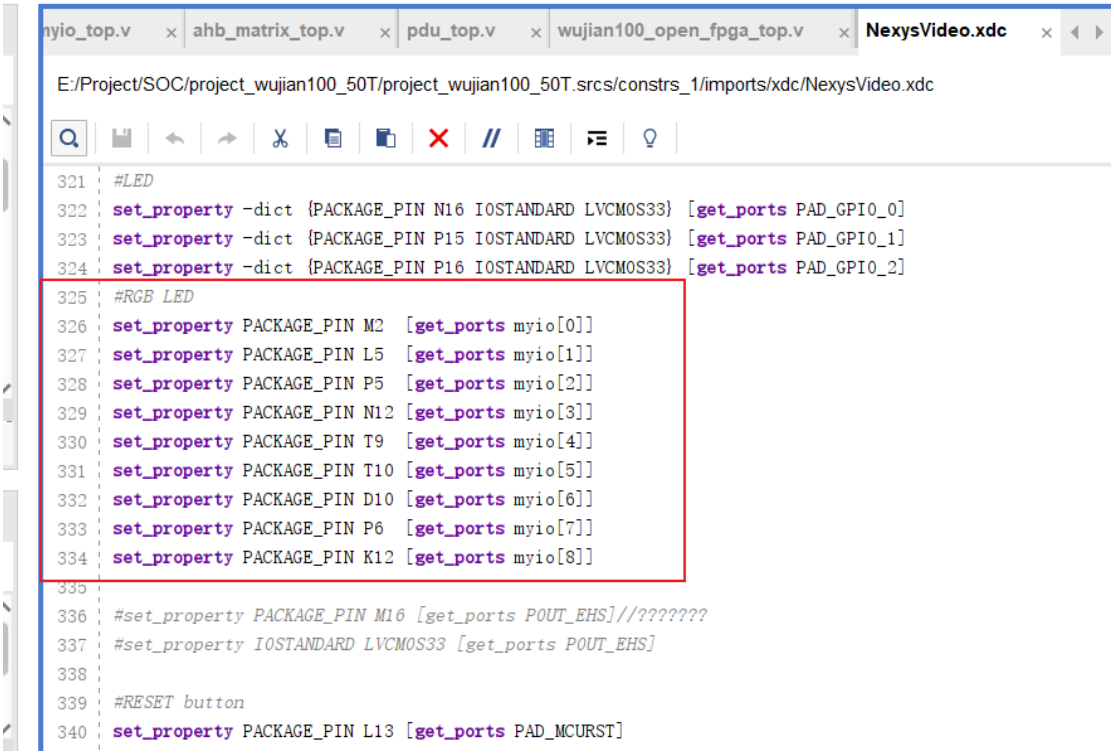


同样，增加输出引脚和定义

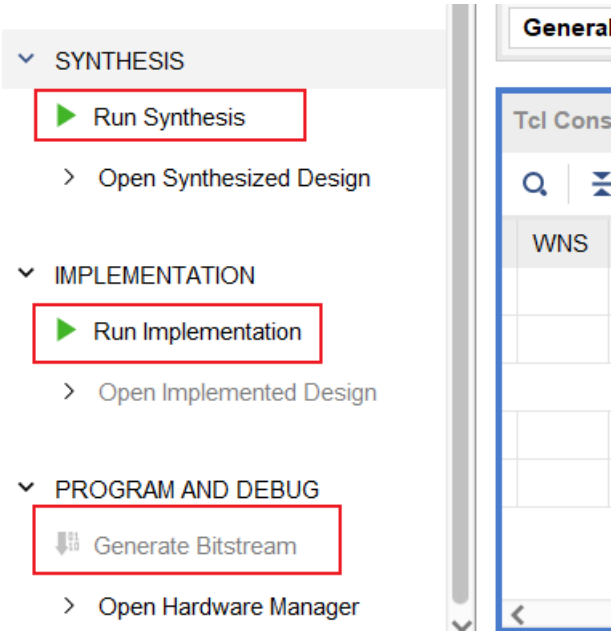


现在就实现了 IP 的挂载。

在约束文件中添加对应功能的引脚，这里将 myio 的低 9 位连接到三色灯的 9 个引脚上。

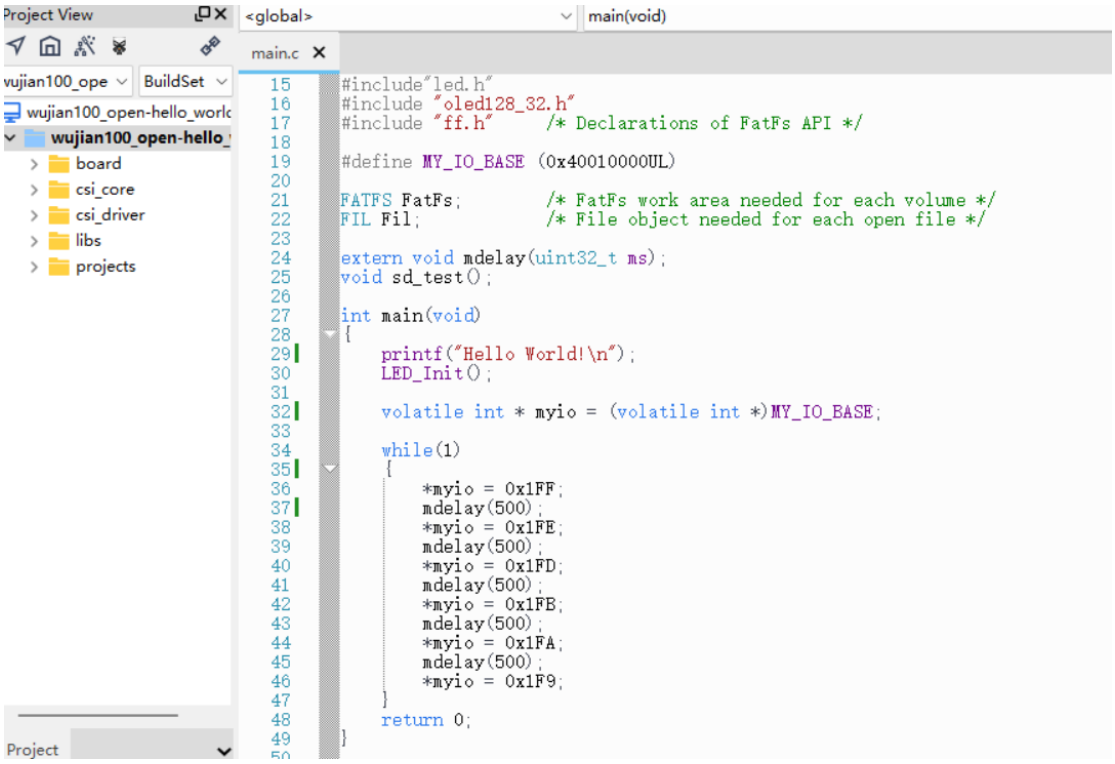


像之前的教程一样，综合，实现，再生成 BIT 文件，下载到开发板上。



三、 实现与验证

在 CDK 中编写简单的程序验证功能。



附：xdc 文件修改说明

说明：修改 xdc 文件（即该芯片的约束文件），要配置成所用板子一致的 xdc

需要的资料：开发板用户手册；wujian100 工程里的 xdc 文件

1. 开发板用户手册重点阅读内容：

1) 时钟：

2.4 有源晶振

开发板提供的 FPGA 系统时钟源为 50Mhz 有源晶振电路。晶振输出信号端 SYS_CLK 连接到 FPGA 的 BANK14 全局时钟管脚 N14（IO_L12P_T1_MRCC_14），这

2) IO 口电压：不同开发板用的电压不一样

XC7A35T-1FTG256C 共有 5 个 I/O Bank，其中 U2E 是 FPGA 专用的配置 Bank，其余 4 个 Bank I/O 的连接情况如下表。

Bank	用途
Bank14（3.3V）	LED/KEY/P1/P2
Bank15（3.3V）	P1/P2/JP1
Bank34（3.3V）	USER_JTAG/USER_FLASH/P1
Bank35（1.5V）	DDR3

设置电压的约束：

set_property CONFIG_VOLTAGE 3.3 [current_design] ：全部设定电压为 3.3

3) 引脚：以灯和开关为例子：

9 个 LED		
LED	信号名称	FPGA 引脚
D0	LED0	M16
D1	LED1	N16
D2	LED2	P15
D3	LED3	P16
D4	D4B	M2
	D4G	L5
	D4R	P5
D5	D5B	N12
	D5G	T9
	D5R	T10
D6	D6B	D10
	D6G	P6
	D6R	K12
D7	FPGA_DONE	H10
D8	电源指示灯	-

4) Flash

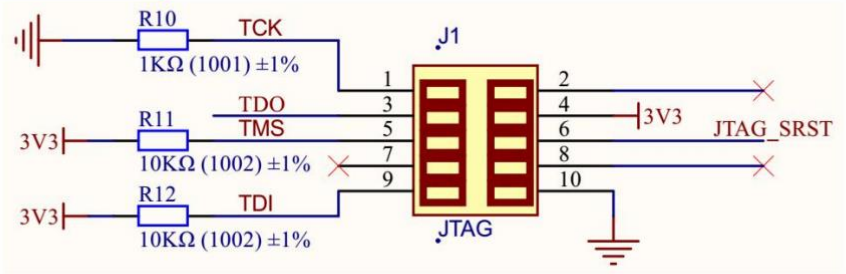
2.9 SPI FLASH

开发板上使用了两片 8MB(64Mbit)大小的 SPI FLASH 芯片，型号为 25Q064A，它

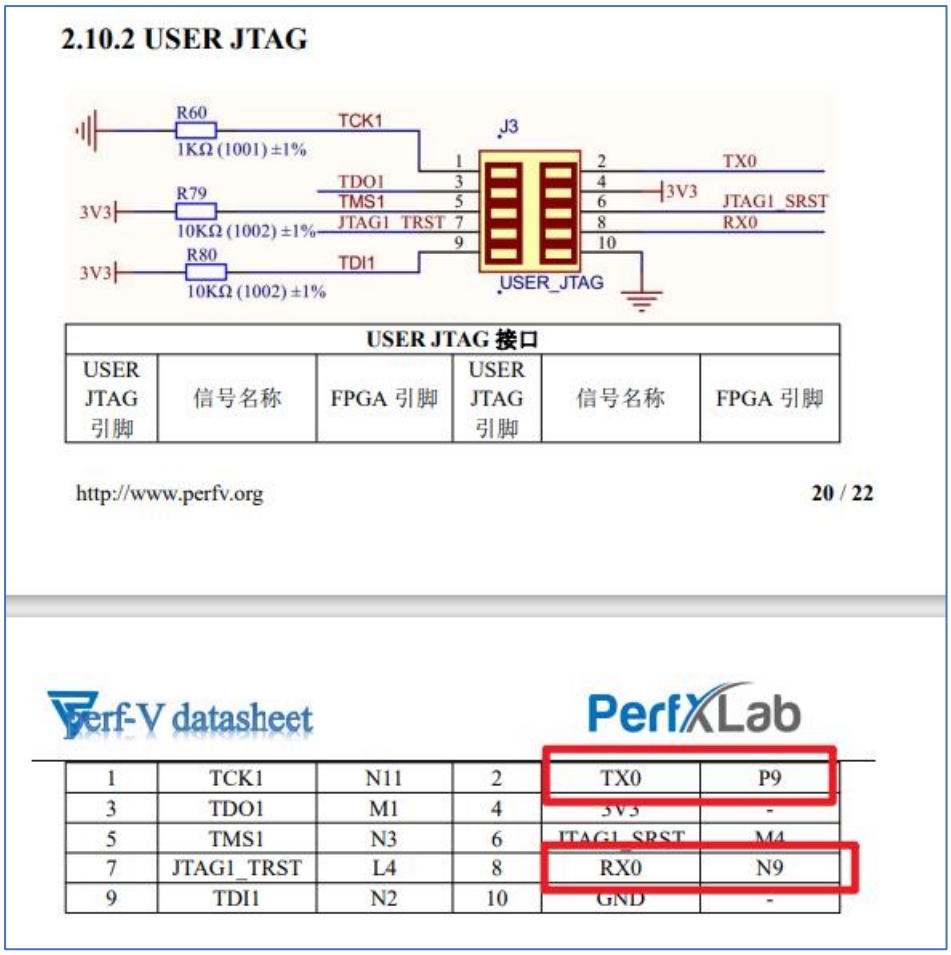
5) 接口：

2.10.1 JTAG

专用于下载 FPGA 程序或者固化程序到 FLASH。JTAG 线插拔的时候请断电，注意不要热插拔。



JTAG 接口					
JTAG 引脚	信号名称	FPGA 引脚	JTAG 引脚	信号名称	FPGA 引脚
1	TCK	L7	2	-	-
3	TDO	N8	4	3V3	-
5	TMS	M7	6	JTAG_SRST	P14
7	-	-	8	-	-
9	TDI	N7	10	GND	-



两种接口区别：

UART：是嵌入式开发所说的串口。通过它可以向设备烧写程序和调试程序。

JTAG：用于设备调试，需要硬件支持。主要用于芯片内部测试，调试程序。可以直接观察和修改寄存器和内存中的数据，方便找出程序中的问题。

2. xdc 约束更改

1) 时钟：

```
## Clock Signal
set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports { clk }]; #IO_L13P_T2_MRCC_34 Sch=sysclk
create_clock -add -name sys_clk_pin -period 20.00 waveform {0 10} [get_ports clk]
```

引脚　　周期

set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets PAD_JTAG_TCLK]

这一句必须加上

2) LED：

注意这三列，对照开发板资料给的引脚改

```
## LEDs
set_property -dict { PACKAGE_PIN M16   IOSTANDARD LVCMOS33 } [get_ports { POUT_EHS }]; #IO_L15P_T2_DQS_13 Sch=led[0]
set_property -dict { PACKAGE_PIN N16   IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_0 }]; #IO_L15N_T2_DQS_13 Sch=led[1]
set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_1 }]; #IO_L17P_T2_13 Sch=led[2]
set_property -dict { PACKAGE_PIN P16   IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_2 }]; #IO_L17N_T2_13 Sch=led[3]
set_property -dict { PACKAGE_PIN M2   IOSTANDARD LVCMOS33 } [get_ports { PAD_USI0_NSS }]; #IO_L14N_T2_SRCC_13 Sch=led[4]
set_property -dict { PACKAGE_PIN P5   IOSTANDARD LVCMOS33 } [get_ports { PAD_USI0_SD1 }]; #IO_L16N_T2_13 Sch=led[5]
set_property -dict { PACKAGE_PIN K12   IOSTANDARD LVCMOS33 } [get_ports { PAD_PWM_CH9 }]; #IO_L16P_T2_13 Sch=led[6]
set_property -dict { PACKAGE_PIN H10   IOSTANDARD LVCMOS33 } [get_ports { PAD_USI2_NSS }]; #IO_L5P_T0_13 Sch=led[7]


## Buttons
set_property -dict { PACKAGE_PIN R15  IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_8 }]; #IO_L20N_T3_16 Sch=btnc
set_property -dict { PACKAGE_PIN R16  IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_9 }]; #IO_L22N_T3_16 Sch=btnd
set_property -dict { PACKAGE_PIN T14  IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_10 }]; #IO_L20P_T3_16 Sch=btnl
set_property -dict { PACKAGE_PIN M15  IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_11 }]; #IO_L6P_T0_16 Sch=btnr
#set_property -dict { PACKAGE_PIN F15  IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_12 }]; #IO_0_16 Sch=btnu
set_property -dict { PACKAGE_PIN L13   IOSTANDARD LVCMOS33 } [get_ports { PAD_MCURST }]; #IO_L12N_T1_MRCC_35 Sch=cpu_resetn
```

3) 复位:

```
set_property PACKAGE_PIN L13 [get_ports PAD_MCURST]
set_property IOSTANDARD LVCMOS33 [get_ports PAD_MCURST]
```

复位引脚

4) UART JTAG：

```
set_property PACKAGE_PIN T8 [get_ports PAD_USI0_SD0]
set_property IOSTANDARD LVCMOS33 [get_ports PAD_USI0_SD0]
set_property PACKAGE_PIN T7 [get_ports PAD_USI0_SCLK]
set_property IOSTANDARD LVCMOS33 [get_ports PAD_USI0_SCLK]
```

串口

5) JTAG：

```
set_property PACKAGE_PIN F12 [get_ports PAD_JTAG_TCLK]
set_property IOSTANDARD LVCMOS33 [get_ports PAD_JTAG_TCLK]
set_property PACKAGE_PIN F13 [get_ports PAD_JTAG_TMS]
set_property IOSTANDARD LVCMOS33 [get_ports PAD_JTAG_TMS]
```

CLINK连接时的引脚设置

6）SD 卡：

注意这两列，对照开发板资料给的引脚改

```
#SD card
set_property PACKAGE_PIN L4 [get_ports { PAD_GPIO_6 }]; #Sch=sd_cclk        sclk       PAD_GPIO_6
set_property PACKAGE_PIN T2 [get_ports { PAD_GPIO_18 }]; #Sch=sd_cd         card detect
set_property PACKAGE_PIN R1 [get_ports { PAD_GPIO_5 }]; #Sch=sd_cmd         mosi       PAD_GPIO_5
set_property PACKAGE_PIN H16 [get_ports { PAD_GPIO_7 }]; #Sch=sd_d[0]       miso       PAD_GPIO_7
set_property PACKAGE_PIN R2 [get_ports { PAD_GPIO_19 }]; #Sch=sd_d[3]       cs         PAD_GPIO_19
# set_property PACKAGE_PIN F12 [get_ports { PAD_GPIO_20 }]; #Sch=sd_reset    rst        PAD_GPIO_20
```

3.其他

```
## Switches  29 30 31
set_property -dict { PACKAGE_PIN T15  IOSTANDARD LVCMOS33 } [get_ports { PAD_PWM_CH10 }]; #IO_L22P_T3_16 Sch=sw[0]
set_property -dict { PACKAGE_PIN L14  IOSTANDARD LVCMOS33 } [get_ports { PAD_PWM_CH11 }]; #IO_25_16 Sch=sw[1]
set_property -dict { PACKAGE_PIN M14  IOSTANDARD LVCMOS33 } [get_ports { PAD_PWM_CH0 }]; #IO_L24P_T3_16 Sch=sw[2]
set_property -dict { PACKAGE_PIN K13  IOSTANDARD LVCMOS33 } [get_ports { PAD_PWM_CH1 }]; #IO_L24N_T3_16 Sch=sw[3]
#set_property -dict { PACKAGE_PIN H17  IOSTANDARD LVCMOS33 } [get_ports { PAD_PWM_CH2 }]; #IO_L6P_T0_15 Sch=sw[4]
#set_property -dict { PACKAGE_PIN J16  IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_31 }]; #IO_0_15 Sch=sw[5]
#set_property -dict { PACKAGE_PIN K13  IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_30 }]; #IO_L19P_T3_A22_15 Sch=sw[6]
#set_property -dict { PACKAGE_PIN M17  IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_29 }]; #IO_25_15 Sch=sw[7]

## OLED Display
set_property -dict { PACKAGE_PIN W22   IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_21 }]; #IO_L7N_T1_D10_14 Sch=oled_dc
set_property -dict { PACKAGE_PIN U21   IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_22 }]; #IO_L4N_T0_D05_14 Sch=oled_res
set_property -dict { PACKAGE_PIN W21   IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_23 }]; #IO_L7P_T1_D09_14 Sch=oled_sclk
set_property -dict { PACKAGE_PIN Y22   IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_24 }]; #IO_L9N_T1_DQS_D13_14 Sch=oled_sdin
set_property -dict { PACKAGE_PIN P20   IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_25 }]; #IO_0_14 Sch=oled_vbat
set_property -dict { PACKAGE_PIN V22   IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_26 }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=oled_vdd

## HDMI in
#set_property -dict { PACKAGE_PIN AA5   IOSTANDARD LVCMOS33 } [get_ports { hdmi_rx_cec }]; #IO_L10P_T1_34 Sch=hdmi_rx_cec
#set_property -dict { PACKAGE_PIN W4    IOSTANDARD TMDS_33  } [get_ports { hdmi_rx_clk_n }]; #IO_L12N_T1_MRCC_34 Sch=hdmi_rx_clk_n
#set_property -dict { PACKAGE_PIN V4    IOSTANDARD TMDS_33  } [get_ports { hdmi_rx_clk_p }]; #IO_L12P_T1_MRCC_34 Sch=hdmi_rx_clk_p
#set_property -dict { PACKAGE_PIN AB12  IOSTANDARD LVCMOS33 } [get_ports { hdmi_rx_hpa }]; #IO_L7N_T1_13 Sch=hdmi_rx_hpa
#set_property -dict { PACKAGE_PIN Y4    IOSTANDARD LVCMOS33 } [get_ports { hdmi_rx_scl }]; #IO_L11P_T1_SRCC_34 Sch=hdmi_rx_scl
#set_property -dict { PACKAGE_PIN AB5   IOSTANDARD LVCMOS33 } [get_ports { hdmi_rx_sda }]; #IO_L10N_T1_34 Sch=hdmi_rx_sda
#set_property -dict { PACKAGE_PIN R3    IOSTANDARD LVCMOS33 } [get_ports { hdmi_rx_txen }]; #IO_L3P_T0_DQS_34 Sch=hdmi_rx_txen
```

这些根据自己板子的资源多少和想实现的功能改；比如只想实现点灯，那么这些就可以全注释掉；但如果要实现 OLED，那先看自己板子有没有 OLED 的配置，然后根据开发板手册改 xdc 里相对应的引脚；

以下为移植到 A7 50T 的 xdc 代码：

```
set_property IOSTANDARD LVCMOS33 [get_ports]

#clock
set_property -dict {PACKAGE_PIN N14 IOSTANDARD LVCMOS33} [get_ports clk]
create_clock -period 20.000 -name sys_clk_pin -waveform {0.000 10.000} -add [get_ports clk]

set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets PAD_JTAG_TCLK]

#LED
set_property -dict {PACKAGE_PIN N16 IOSTANDARD LVCMOS33} [get_ports PAD_GPIO_0]
set_property -dict {PACKAGE_PIN P15 IOSTANDARD LVCMOS33} [get_ports PAD_GPIO_1]
set_property -dict {PACKAGE_PIN P16 IOSTANDARD LVCMOS33} [get_ports PAD_GPIO_2]

#RESET button
set_property PACKAGE_PIN L13 [get_ports PAD_MCURST]

#USI0
```

```
set_property PACKAGE_PIN T7 [get_ports PAD_USI0_SCLK]
set_property PACKAGE_PIN T8 [get_ports PAD_USI0_SD0]



#JTAG
set_property PACKAGE_PIN F12 [get_ports PAD_JTAG_TCLK]
set_property PACKAGE_PIN F13 [get_ports PAD_JTAG_TMS]

#SD card
set_property PACKAGE_PIN L4 [get_ports { PAD_GPIO_6 }]; #Sch=sd_cclk sclk PAD_GPIO_6
set_property PACKAGE_PIN T2 [get_ports { PAD_GPIO_18 }]; #Sch=sd_cd card detect
set_property PACKAGE_PIN R1 [get_ports { PAD_GPIO_5 }]; #Sch=sd_cmd mosi PAD_GPIO_5
set_property PACKAGE_PIN H16 [get_ports { PAD_GPIO_7 }]; #Sch=sd_d[0] miso PAD_GPIO_7
set_property PACKAGE_PIN R2 [get_ports { PAD_GPIO_19 }]; #Sch=sd_d[3] cs PAD_GPIO_19
# set_property PACKAGE_PIN F12 [get_ports { PAD_GPIO_20 }]; #Sch=sd_reset rst PAD_GPIO_20

set_property CFGBVS VCCO [current_design]
```