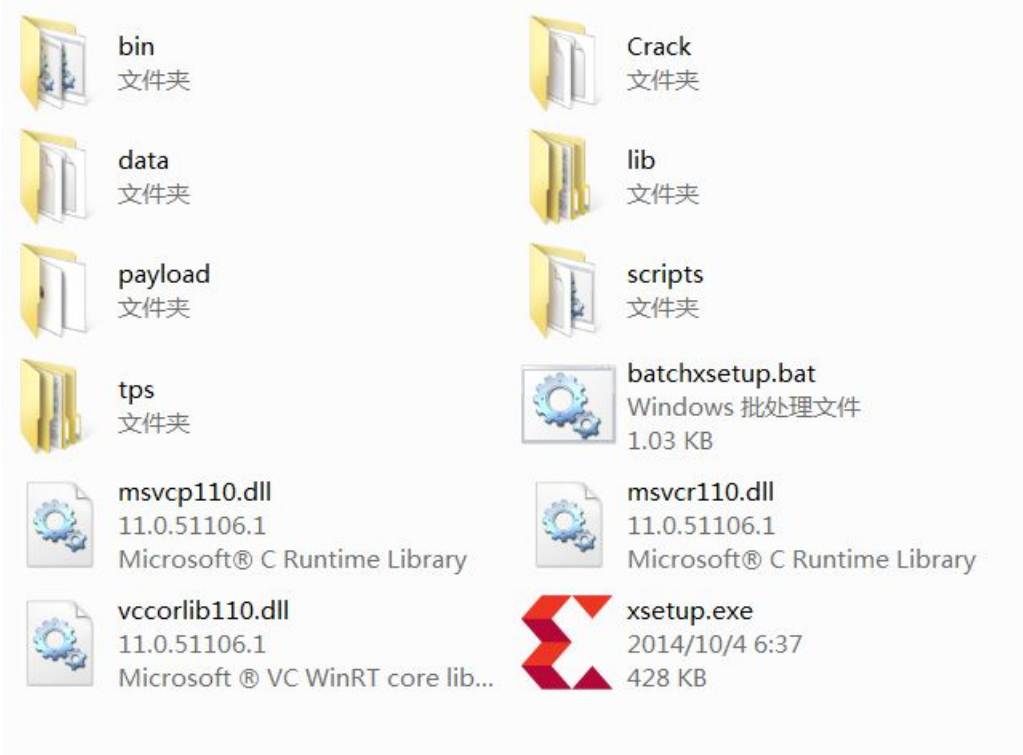
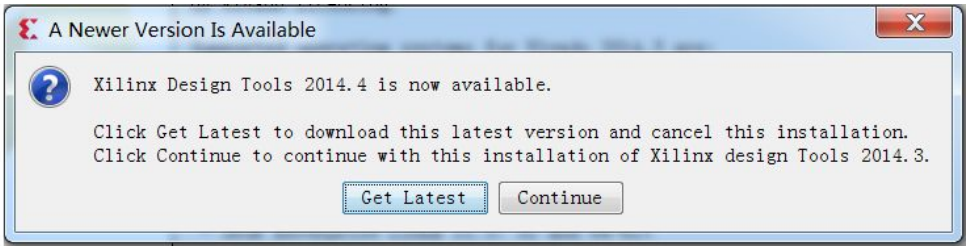
使用 Vivado 制作 FPGA 的简要流程

一、在 Windows 下安装 Xilinx Vivado Design Suite：

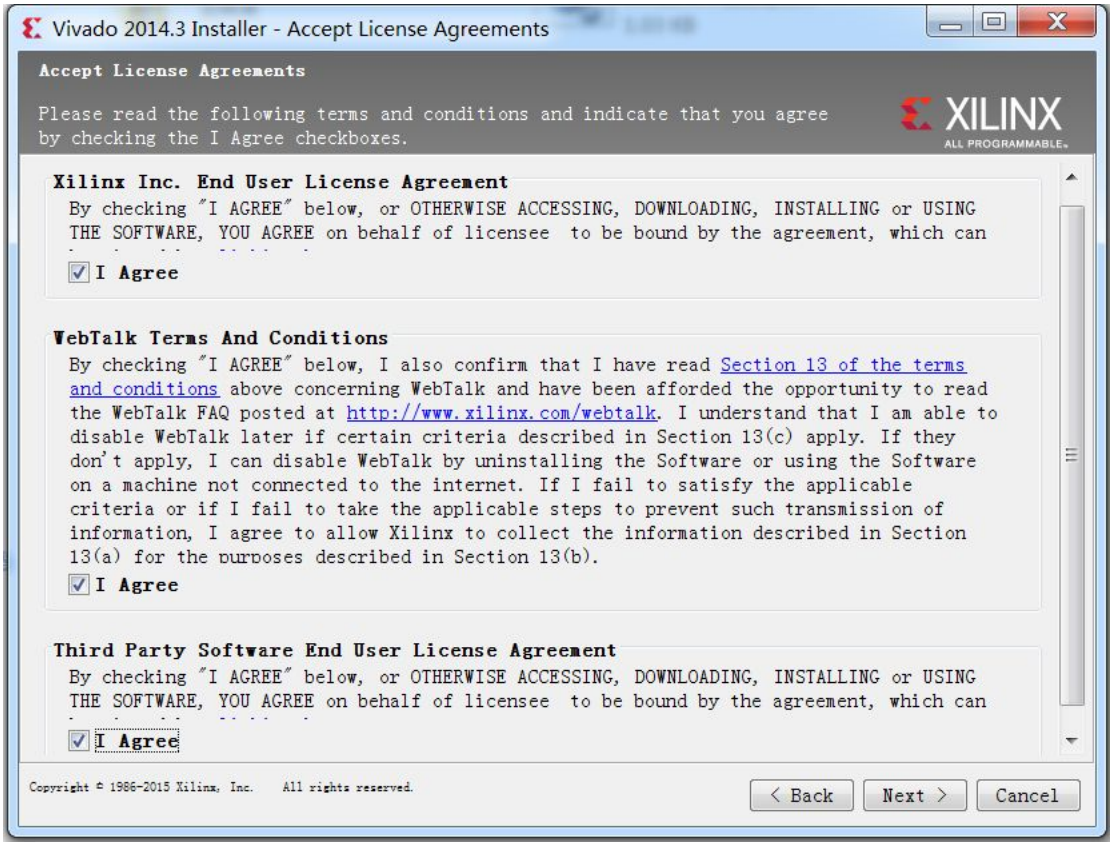
1.1. Xilinx Vivado Design Suite 安装文件，解压后得到安装目录：



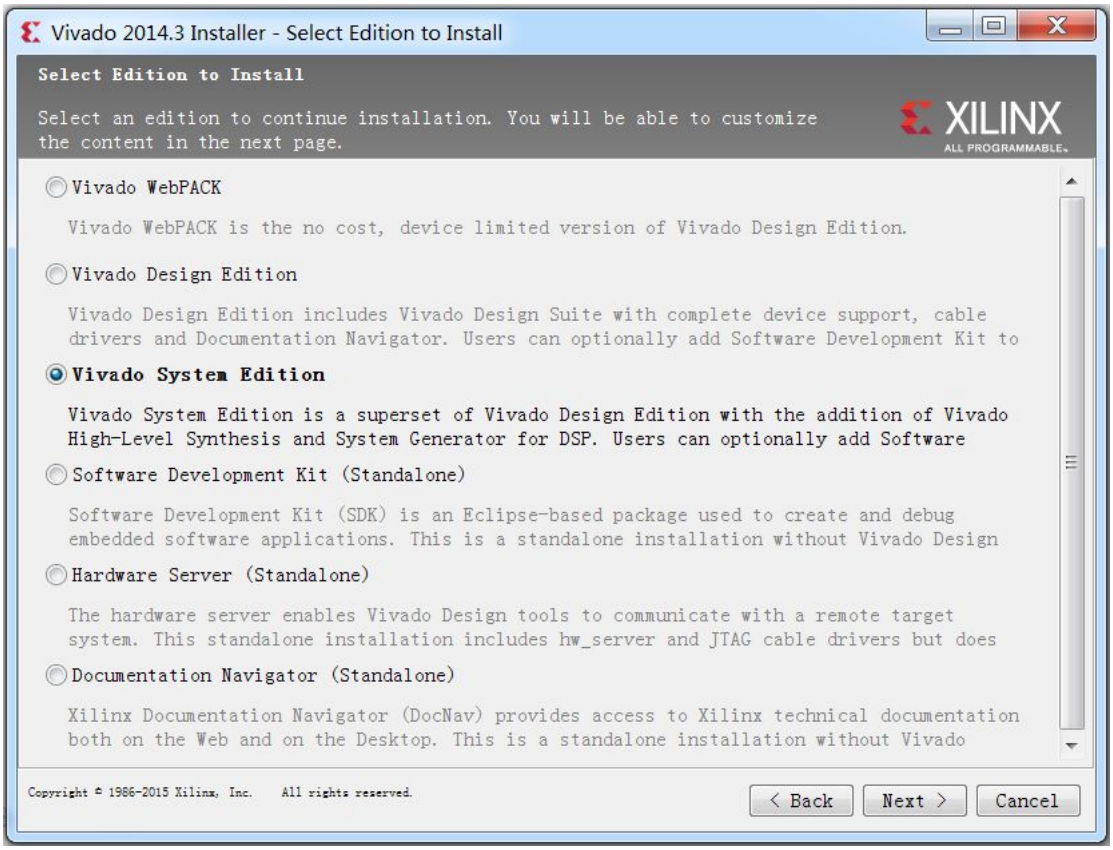
1.2. 运行 xsetup.exe 文件，进入安装程序。如果提示要更新就直接点 continue 关掉。



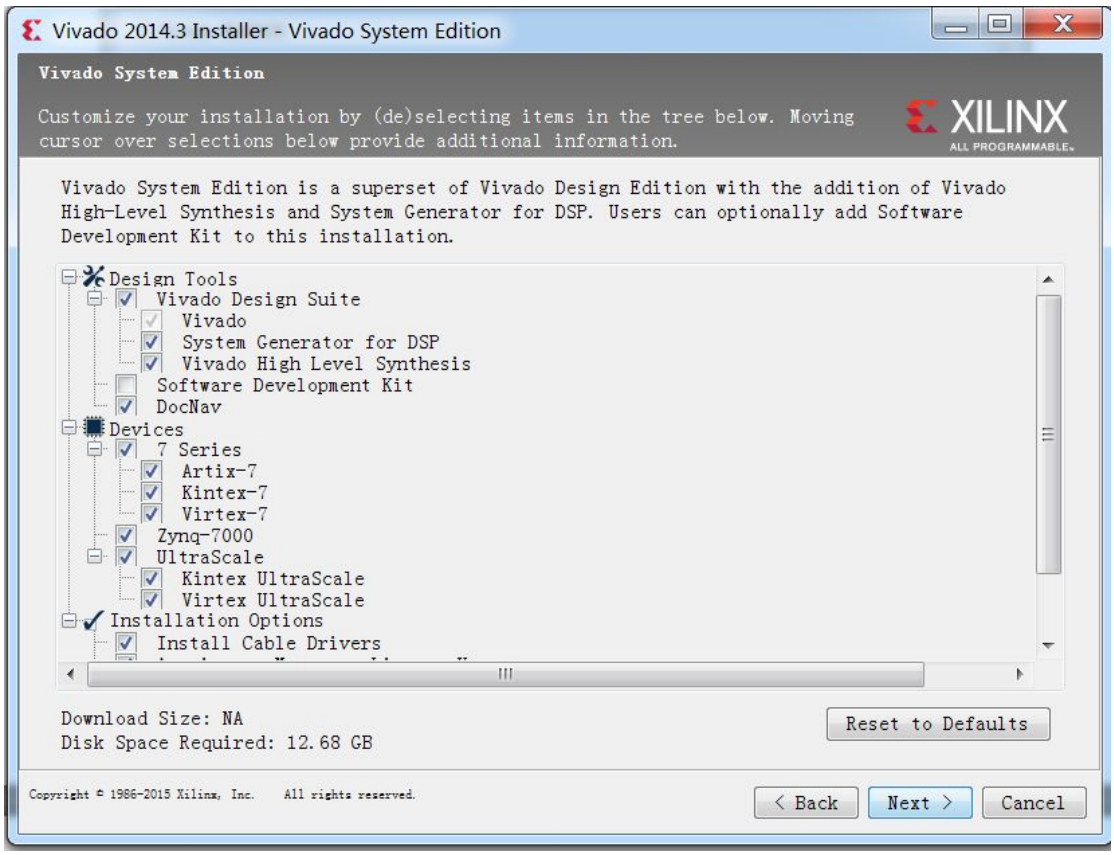
1.3. 选一些根本看都不会看的 I agree.



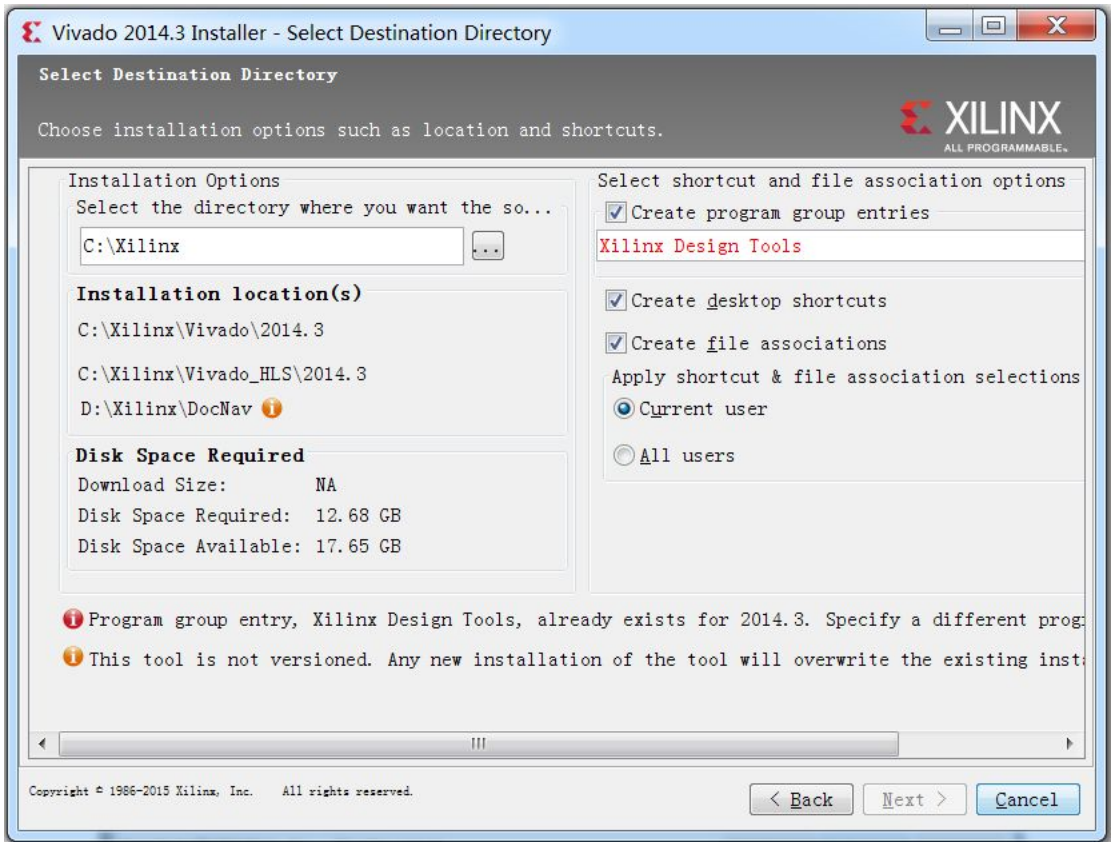
1.4. 选第二个或者第三个应该都可以。我感觉第三个看起来更加高大上一点，我就选了第三个：



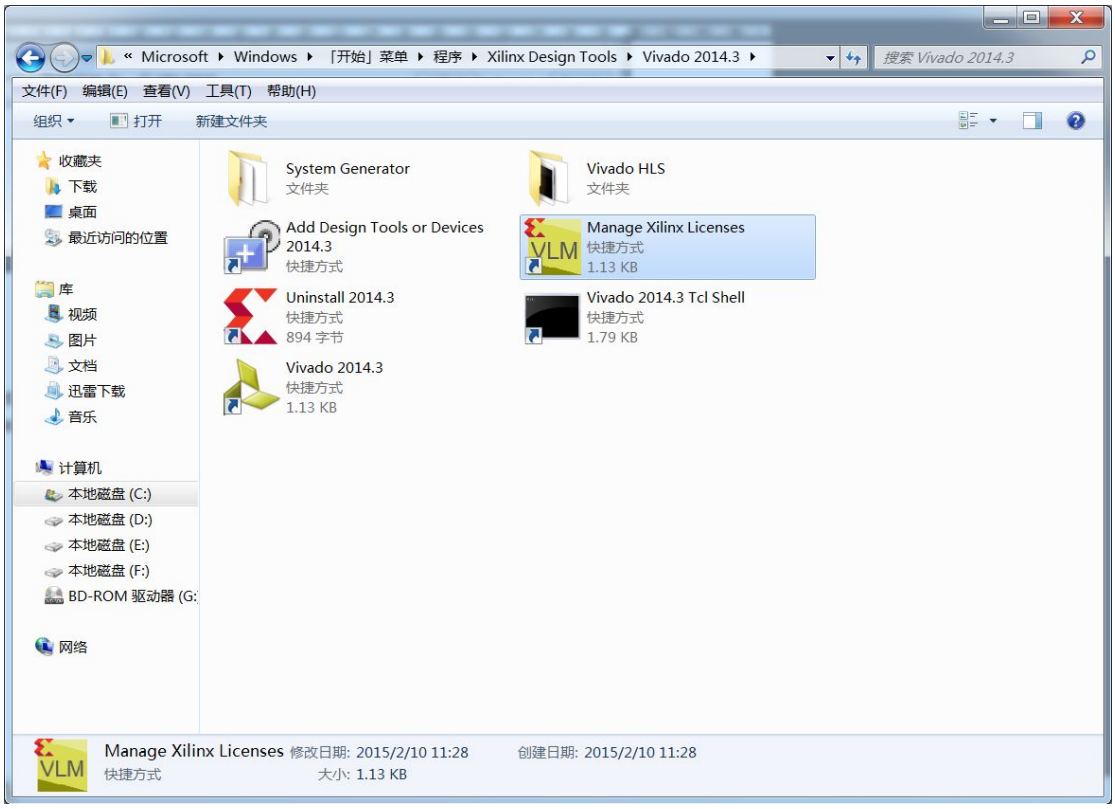
1.5. 直接点 next：



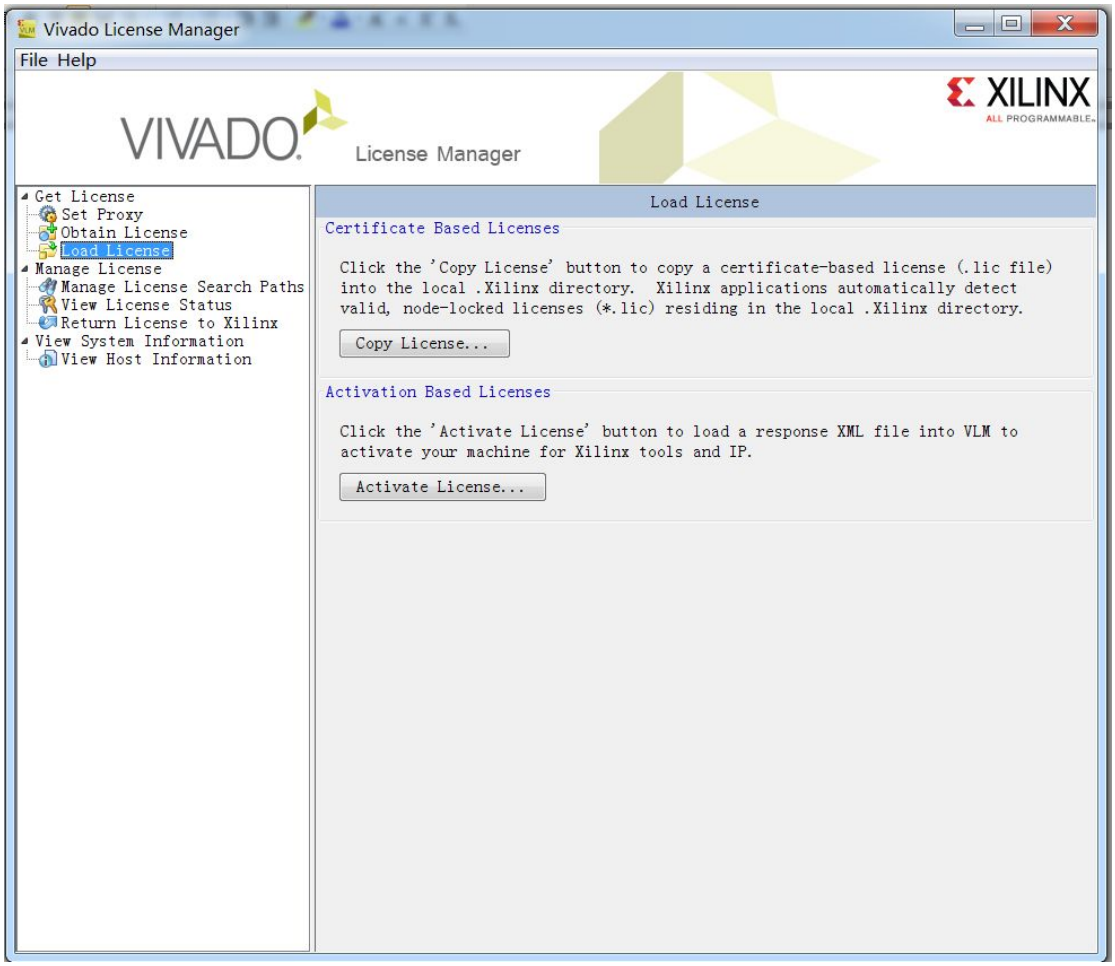
1.6. 选择路径，稍等片刻就能安装完成：



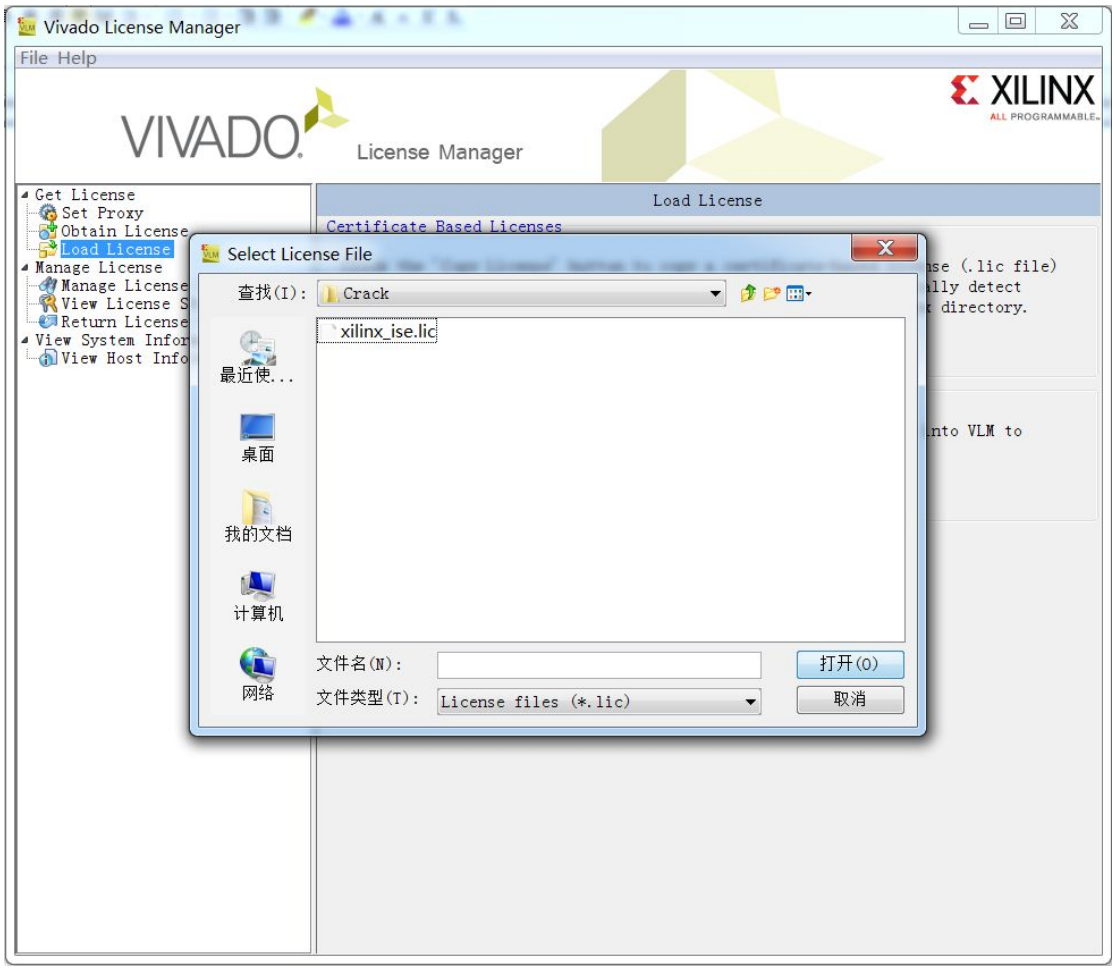
1.7. 安装完成后，在开始菜单找到 Xilinx Design Tools\Vivado 2014.3 文件夹，打开 Manage Xilinx Licenses：



1.8. 打开后选择左边的 load license 选项卡：



1.9. 点击 copy license，选中刚才安装目录中的 crack 文件夹中的 license.lic：



1.10. 至此 Windows 下的 Xilinx Vivado Design Suite 已经全部安装完成。

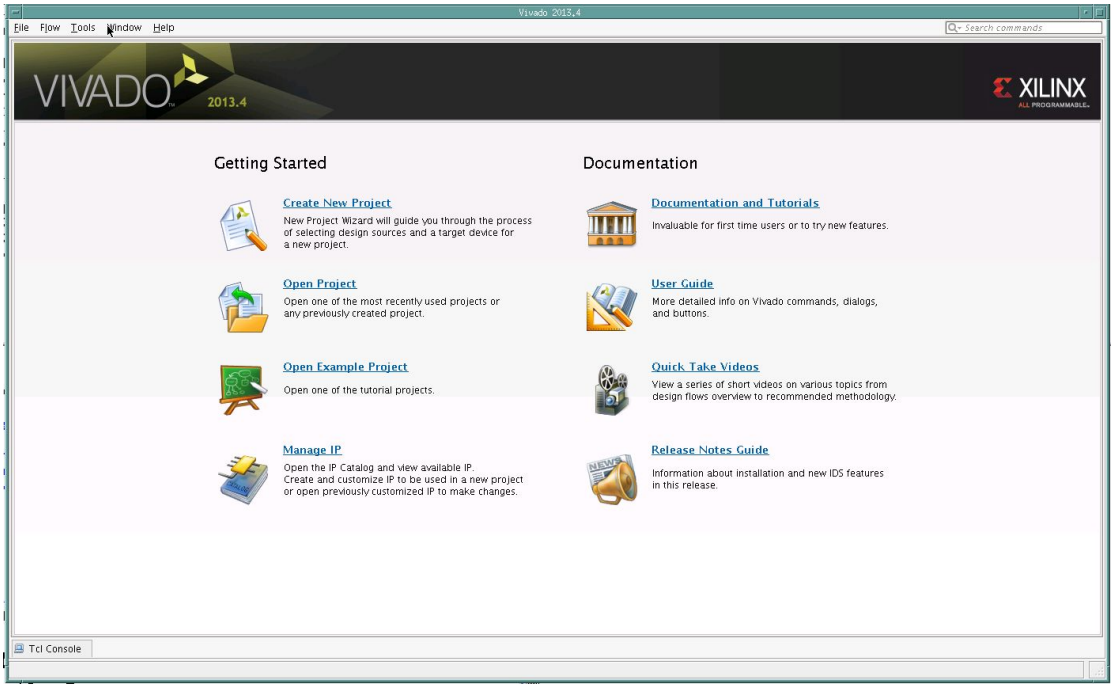
二、在服务器中使用 Vivado 生成 bit 文件：

2.0. 由于综合和布局布线需要较好的硬件资源，所以本次流程中综合和布局布线在 linux 环境的服务器中完成。在 linux 环境中运行 vivado 请确保正确安装 JVM，在 Windows 环境下图形界面流程完全一致。

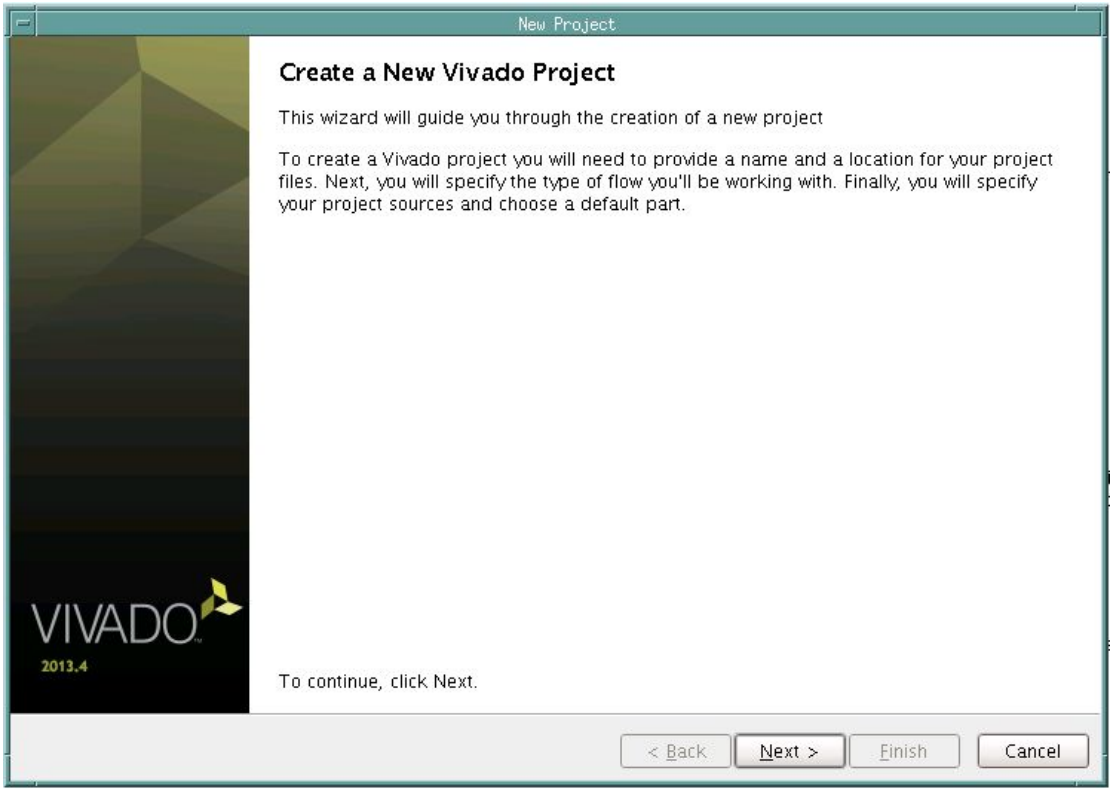
2.1. 在服务器上正确安装 JVM 后执行以下指令：

```
1 source /tools/fpga/Xilinx_Vivado_SDK_2013.4_1210_1/Vivado/2013.4/settings64.csh
```

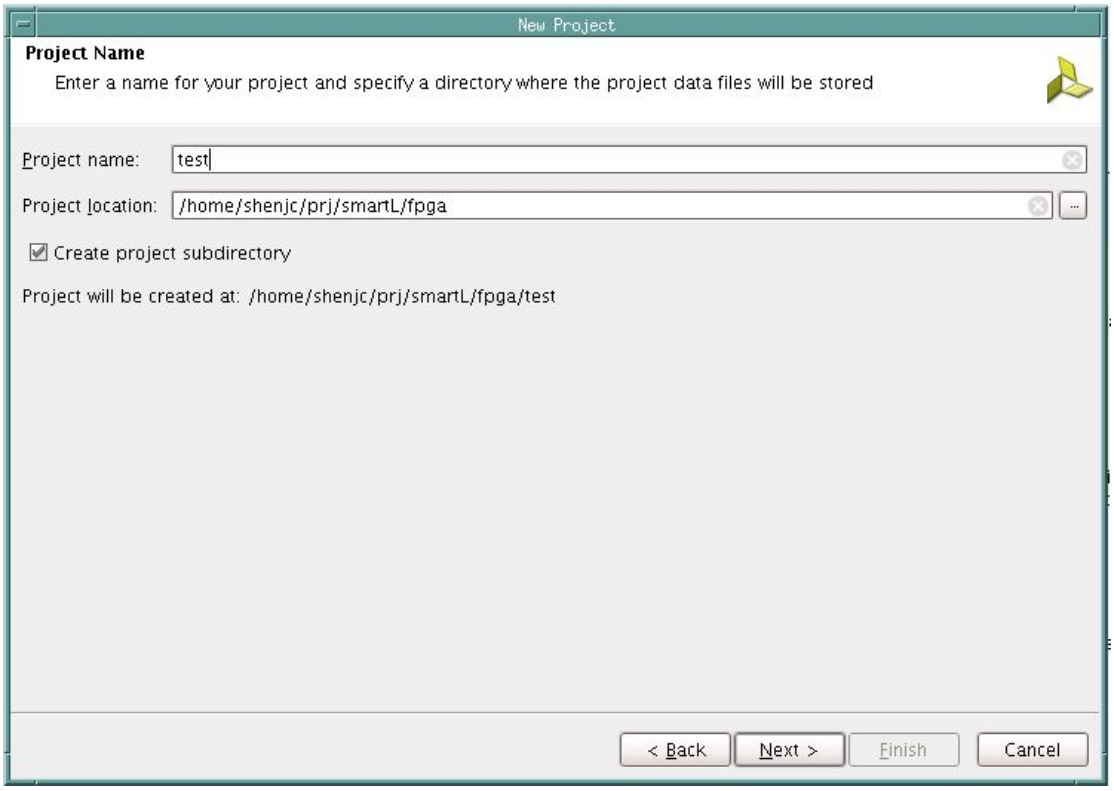
2.2. 执行“Vivado &”打开 Vivado。注意 Vivado 会在你执行这条命令的目录下生成一些 log 信息，所以最好新建一个目录再打开 Vivado：



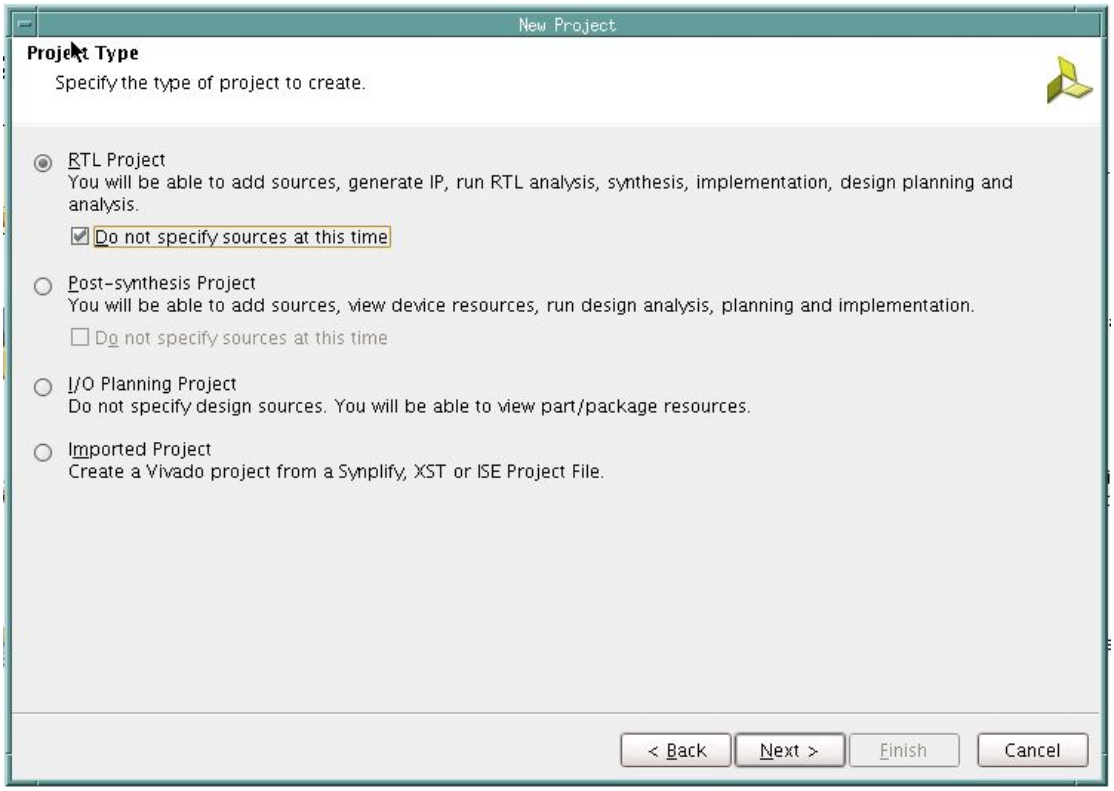
2.3. 点击 Create New Project 建立新的项目，在弹出的对话框中点 Next：



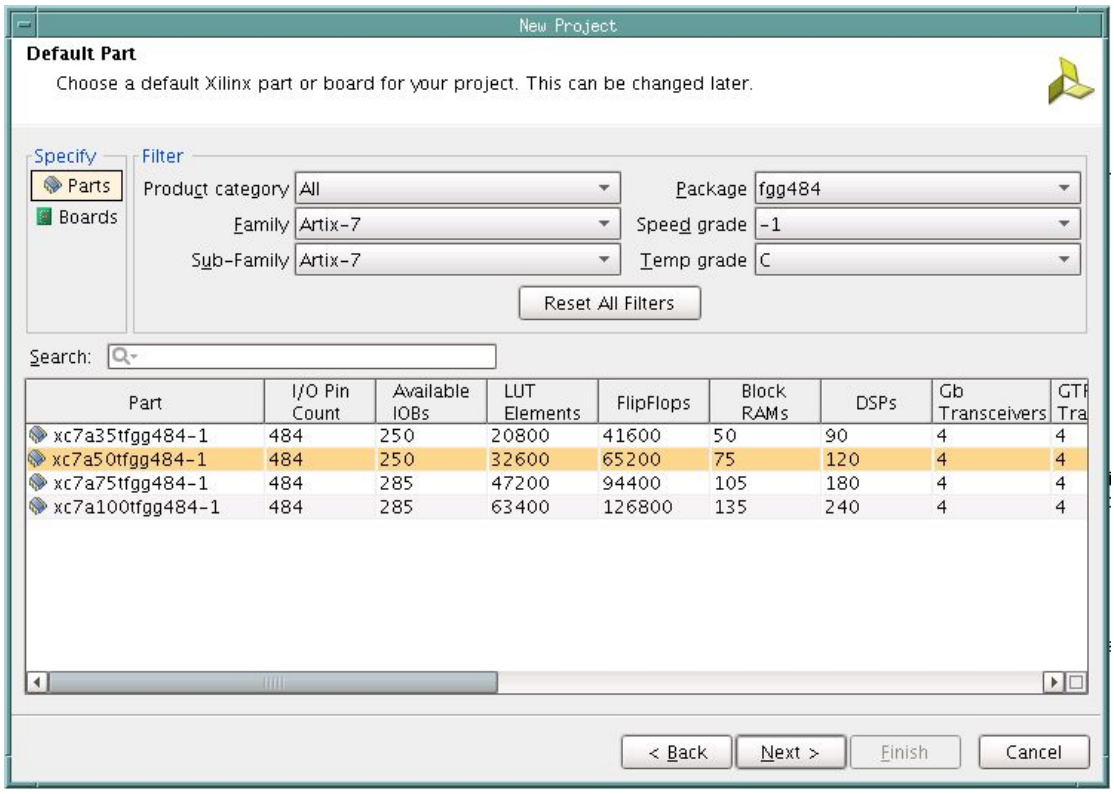
2.4. 输入项目名称，然后再点击 Next：



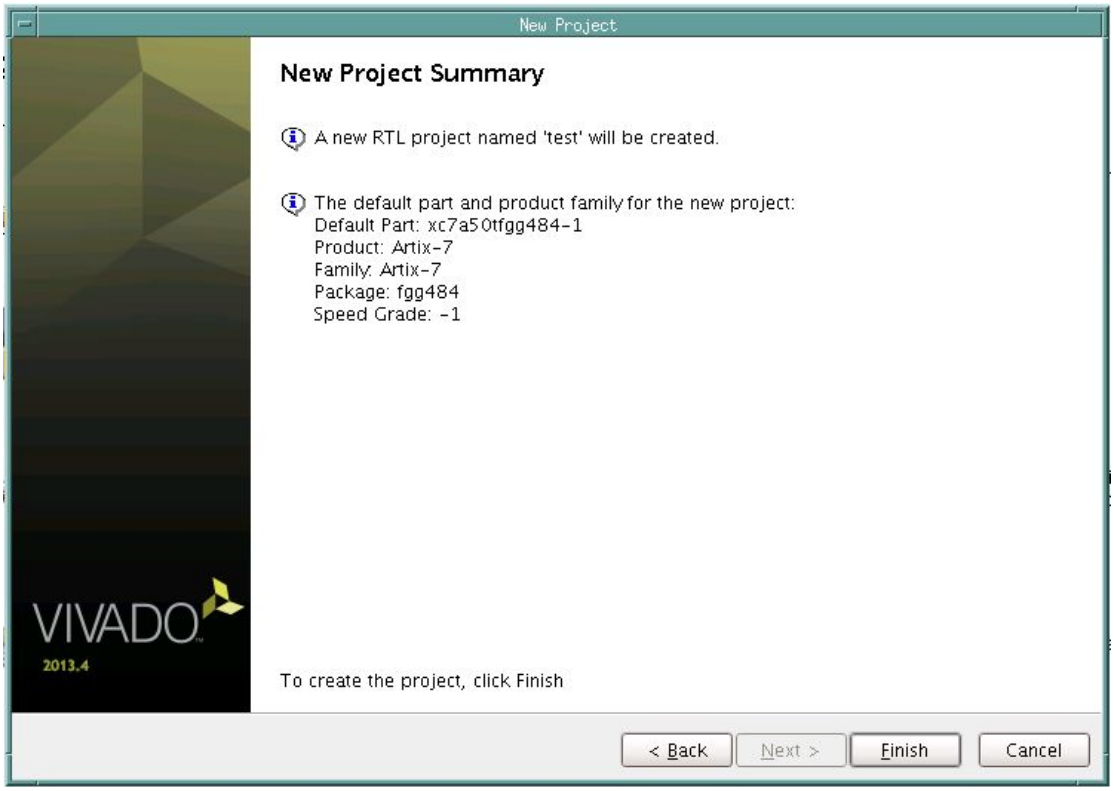
2.5. 选择项目类型，因为我们要从 RTL 代码开始综合，因此选择 RTL Project。下面的 Do not specify source at this time 的勾也可以打上。如果不打上，下一步会进入添加 source file：



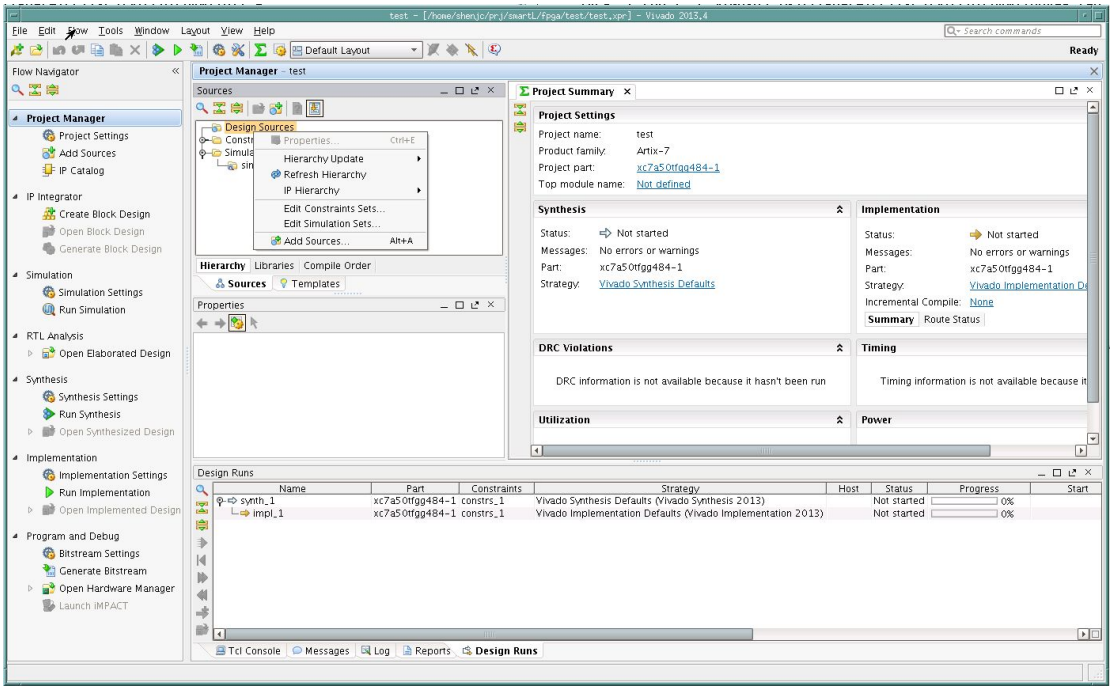
2.6. 选择板子的型号，然后点击 Next。本次流程使用 Artix-7 板子的具体型号如下：



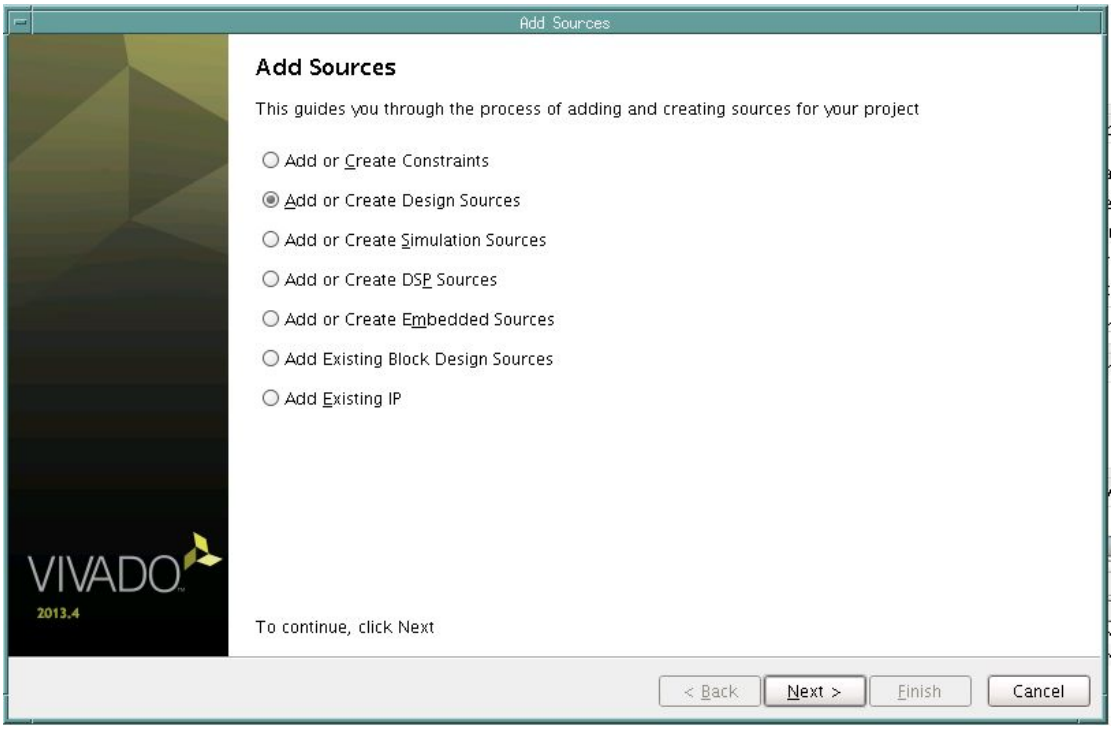
2.7. 再次确认一下板子型号有没有选对，然后点击 Finish 完成项目创建工作：



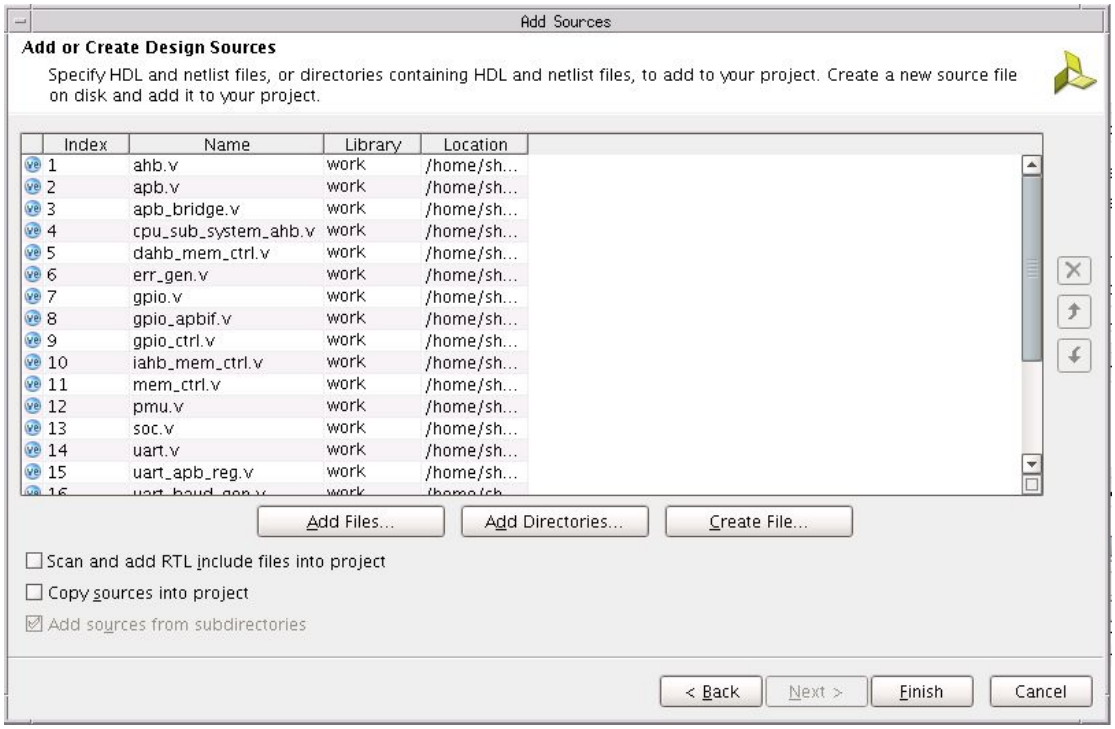
2.8. 右键 Design Sources 或者使用快捷键 Alt+A 开始添加 Source files：



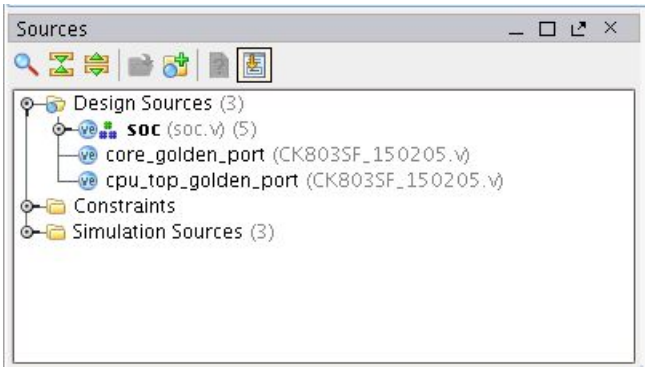
2.9. 选择 Add or Create Design Sources，再点击 Next：



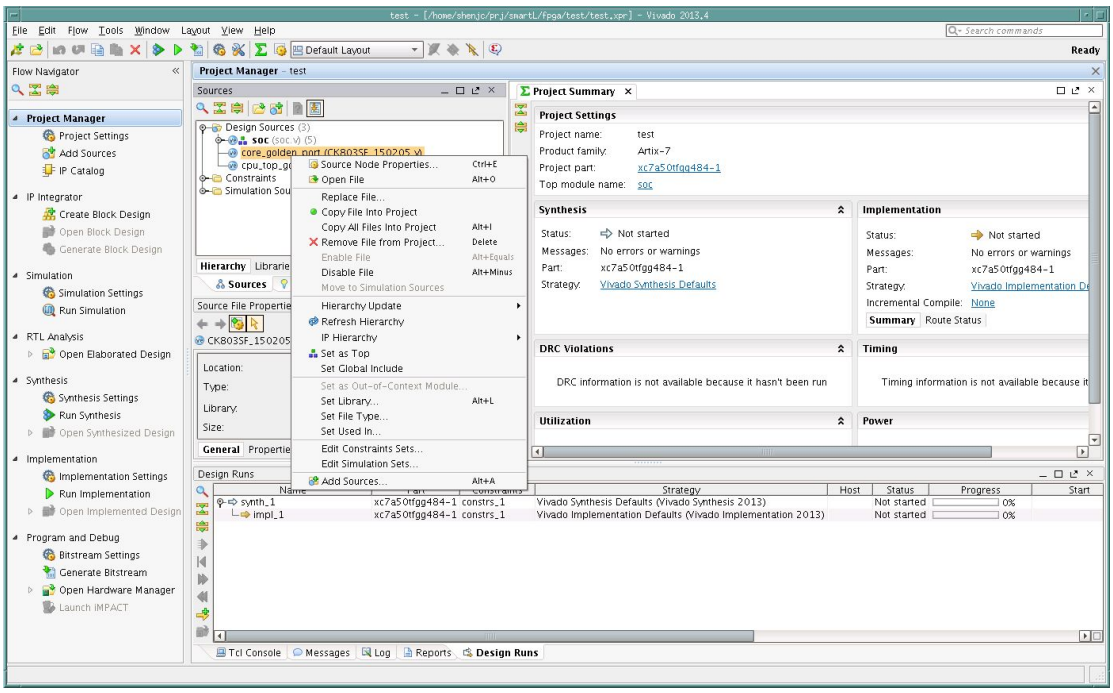
2.10. 点击 Add Files 可以一个个添加源文件，点击 Add Directories 可以按目录添加源文件。在这里加入所有需要的.v 文件和.h 文件。完成后点击 Finish：



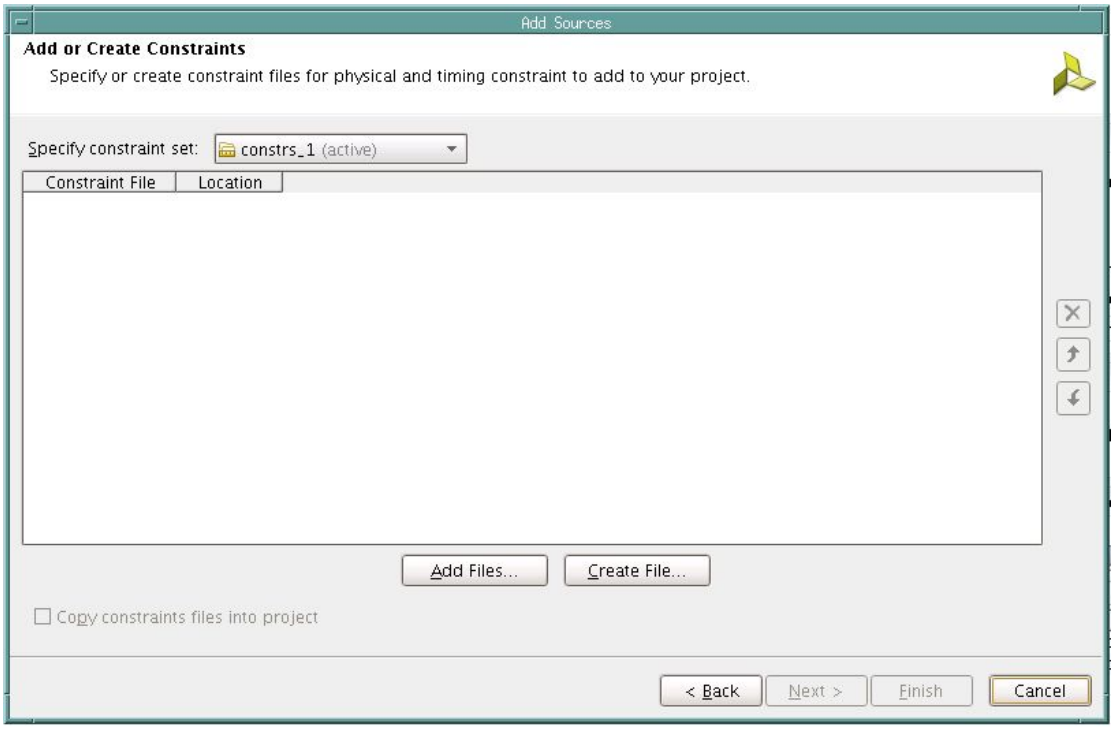
2.11. 如果刚才正确添加了源文件，在这个 Sources 窗口中，Vivado 会自动加粗识别出来的 top module：



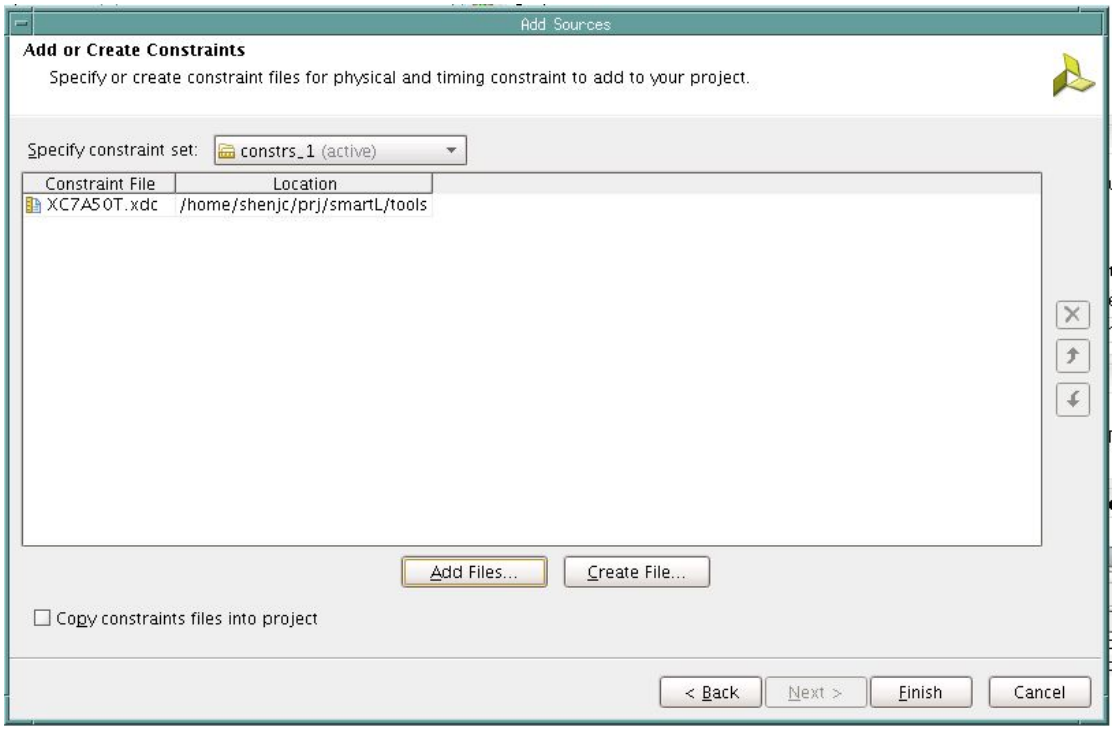
2.12. 有时候难免软件也会有识别错误的时候，右键一个 module，点击 Set as Top 可以手动将其变成 top module：



2.13. 右键Constrains，点击 Add Sources，在接下来弹出的窗口中选择 Add or Create Constrains 后再点击 Finish 来添加约束文件：

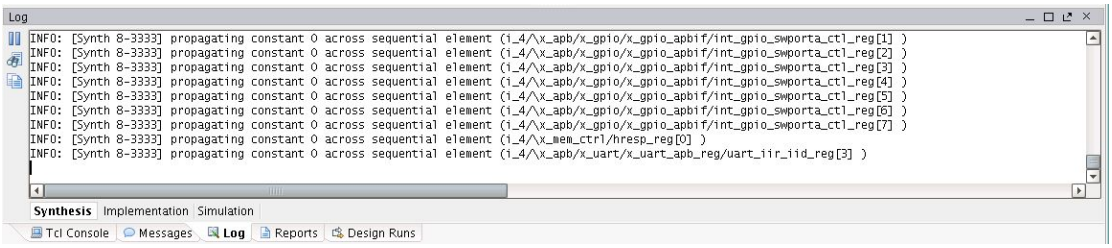


2.14. 要注意的是 Vivado 使用的约束文件格式为 xdc，和 ISE 的约束文件并不能通用，添加完成后点击 Finish：



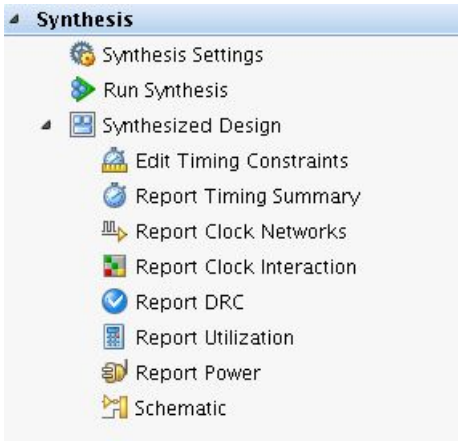
2.15. 完成后，点击 Run Synthesis，即可开始综合并生成网表文件：

2.16. 右上角可以查看现在正在干什么，不开心了可以点 Cancel，下面还可以看一些 report 和 log：

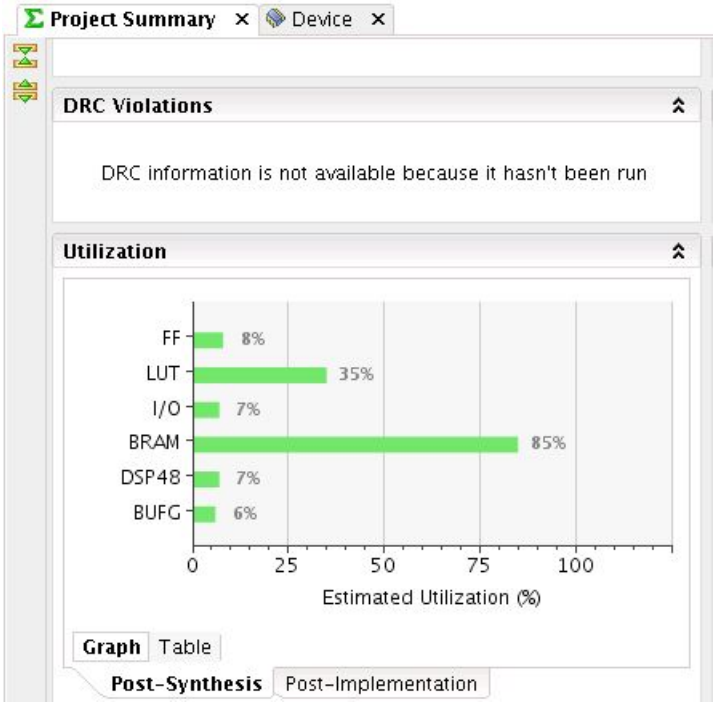


2.17. 综合完成后，会弹出这样一个提示小窗口。如果没什么问题可以直接点击 Run Implementation，在这里我们先点 Open Synthesized Design，看看有什么东西：

2.18. 在这边可以看一些 report：

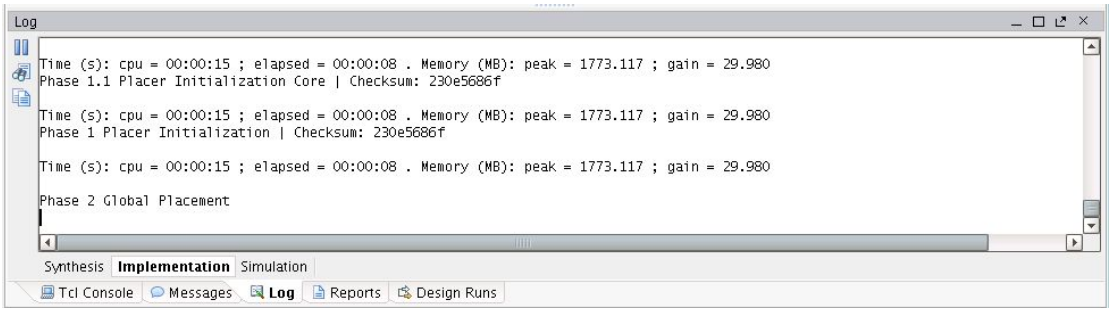


2.19. 在右上角点 Project Summary 可以看一下这次综合以后大约会占用多少板上资源：



2.20. 如果没什么问题就可以点这里的 Run Implementation 来开始布局布线：

2.21. 和综合的时候一样，右上角可以查看现在正在干什么，不开心了可以点 Cancel，下面还可以看一些 report 和 log：



2.22. 完成之后，在 [project_name].runs/impl_1/ 这个目录下会生成 [top_module_name]_routed.dcp 这个文件，继续点击 generate bit file 即可生成 bit 文件：

```
[shenjc@zhiweiguan test.runs]$ d impl_1/
runme.bat       place_design.pb                 soc_drc_routed.rpt
soc.dcp         route_design.pb                 soc_io_placed.rpt
soc_opt.dcp     soc_drc_routed.pb               soc_power_routed.rpt
soc_placed.dcp  soc_power_summary_routed.pb     soc_route_status.rpt
soc_routed.dcp  soc_route_status.pb             soc_timing_summary_routed.rpt
vivado.jou      soc_timing_summary_routed.pb    soc_utilization_placed.rpt
ISEWrap.js*     soc_utilization_placed.pb       ISEWrap.sh*
rundef.js       vivado.pb                       runme.sh*
runme.log       soc.rdi                         soc.tcl
init_design.pb  soc_clock_utilization_placed.rpt htr.txt
opt_design.pb   soc_control_sets_placed.rpt     project.wdf
[shenjc@zhiweiguan impl_1]$ pwd
/home/shenjc/prj/smartL/fpga/test/test.runs/impl_1
```

2.23. 完成 Implementation 之后，可以查看 Implemented Design，在这里可以看到板子上实际资源的使用：

1.1 On-Chip Components

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.010	3	---	---
Slice Logic	0.010	19534	---	---
LUT as Logic	0.009	11316	32600	34.71
CARRY4	<0.001	311	8150	3.81
Register	<0.001	5537	65200	8.49
F7/F8 Muxes	<0.001	654	32600	2.00
BUFG	<0.001	1	32	3.12
Others	0.000	250	---	---
Signals	0.013	16470	---	---
Block RAM	0.043	64	75	85.33
DSPs	<0.001	8	120	6.66
I/O	<0.001	17	250	6.80
Static Power	0.073			
Total	0.149			

2.24. 我们板子上有 75 块 Block Ram，每块 4KB，所以一共有 300KB 的 Block Ram，上面的例子中调用了 256KB Ram，下图是同样的设计，调用 96KB Ram 的资源使用报告，通过

两张图比对也可以证明板子的 Block Ram 资源总量确实是 300KB（其实资源为 337.5KB，但是这是 9bits 的 Block RAM 位宽，若生成 8bits 的 RAM 则多出的 1bit 无法利用，故按 byte 设计的话实际可以利用资源为 300KB）：

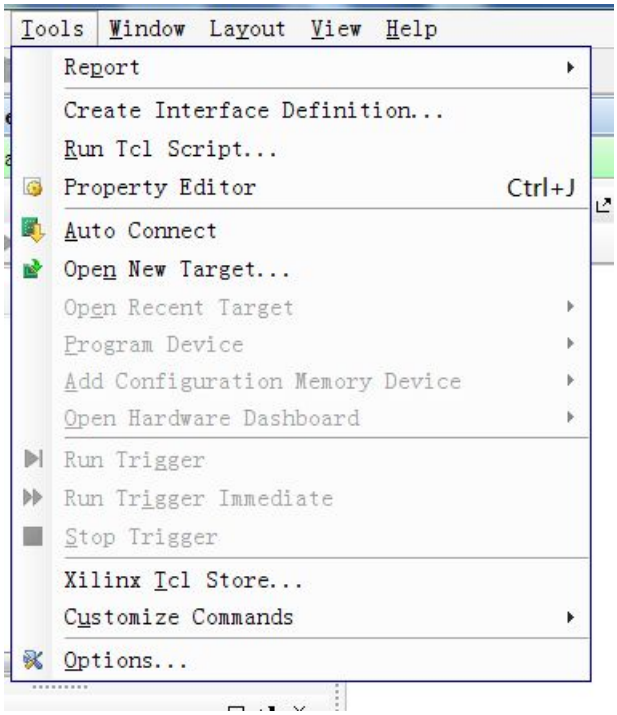
1.1 On-Chip Components

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.010	3	---	---
Slice Logic	0.008	19460	---	---
LUT as Logic	0.008	11304	32600	34.67
CARRY4	<0.001	311	8150	3.81
Register	<0.001	5533	65200	8.48
F7/F8 Muxes	<0.001	601	32600	1.84
BUFG	<0.001	1	32	3.12
Others	0.000	251	---	---
Signals	0.011	16512	---	---
Block RAM	0.015	24	75	32.00
DSPs	<0.001	8	120	6.66
I/O	<0.001	17	250	6.80
Static Power	0.072			
Total	0.116			

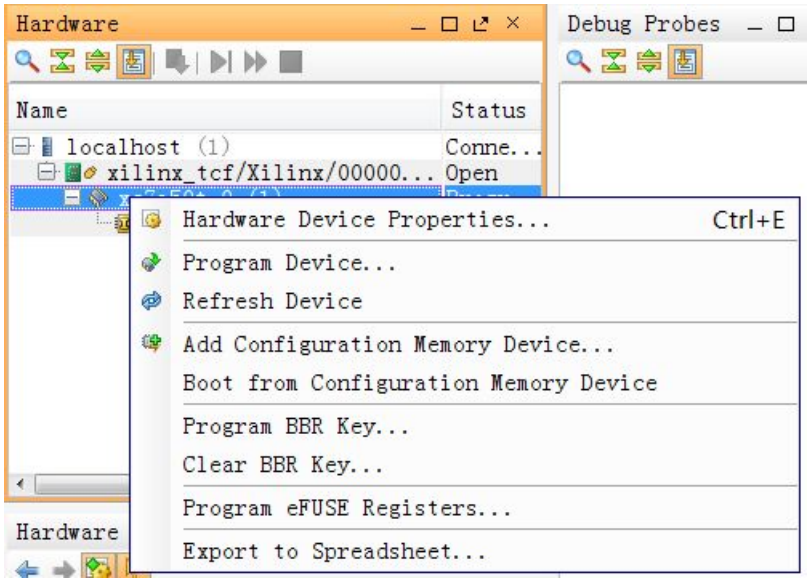
三、使用 Vivado 制作 FPGA：

3.1. 打开 Windows 下的 Vivado，点击 Open Hardware Manager：

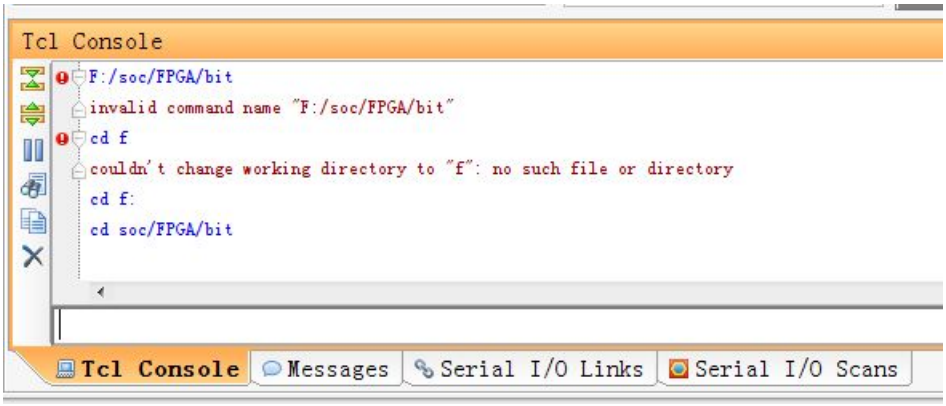
3.2. 在 Hardware Manager 中点击 Tools，再点击 Auto Connect：



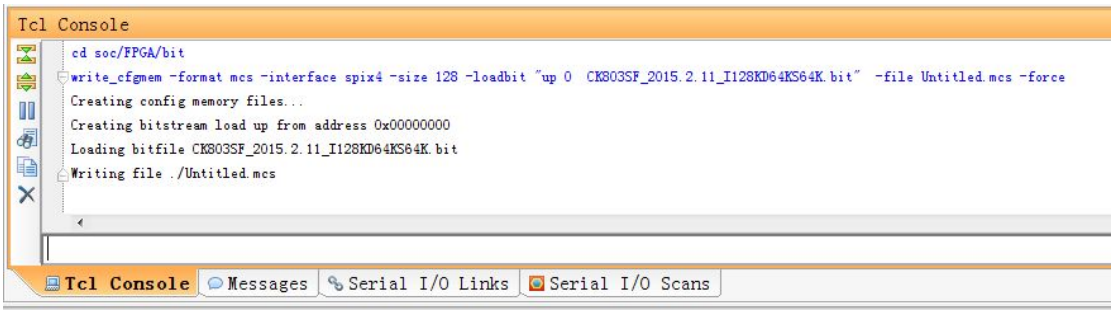
3.3. 连接成功后就会在 Hardware Manager 中看到板子的型号，右键点击板子，再点击 Program Device 就可以烧录 bit 文件：



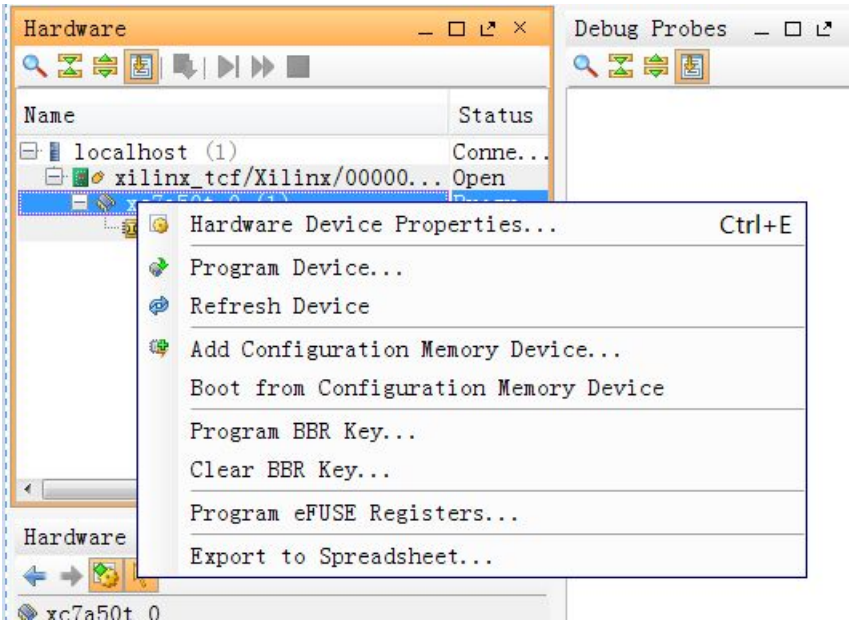
3.4. 如果要烧录 eflash，首先在下面的 Tcl Console 中，通过 cd 命令进入 bit 文件所在的目录，要注意的是，在 Windows 下的目录符是\，而在这里依然要用/作为目录符：



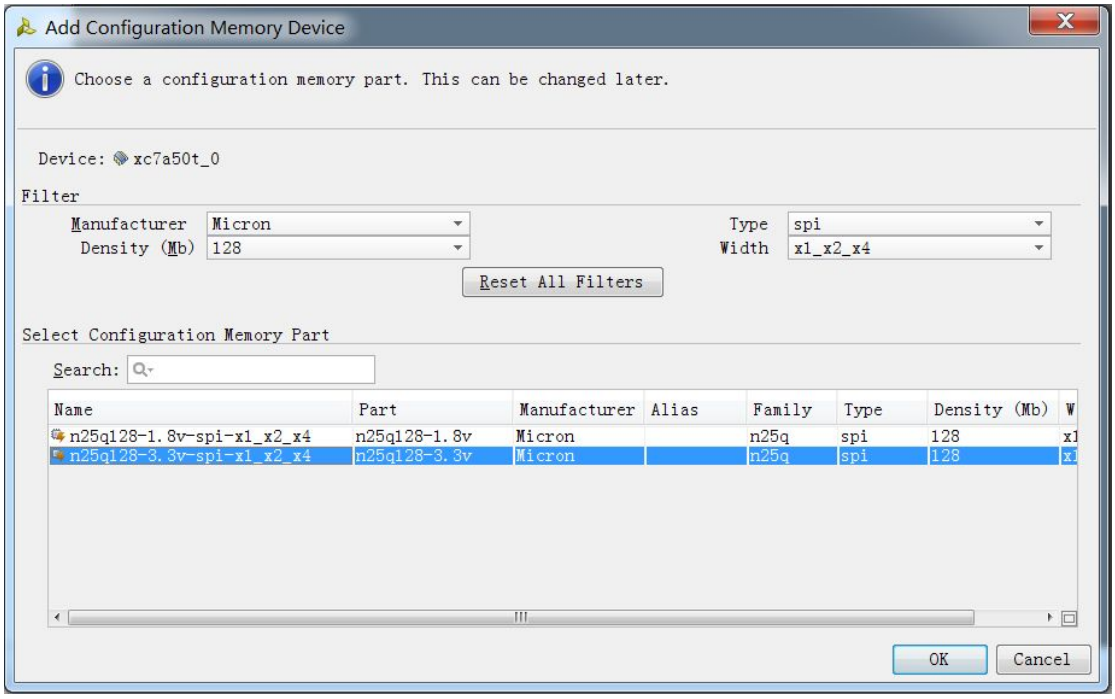
3.5. 输入指令 write_cfgmem -format mcs -interface spix4 -size 128 -loadbit "up 0 xxx.bit" -file Untitled.mcs -force 来生成 mcs 文件，其中 xxx 为你的 bit 文件的名字：



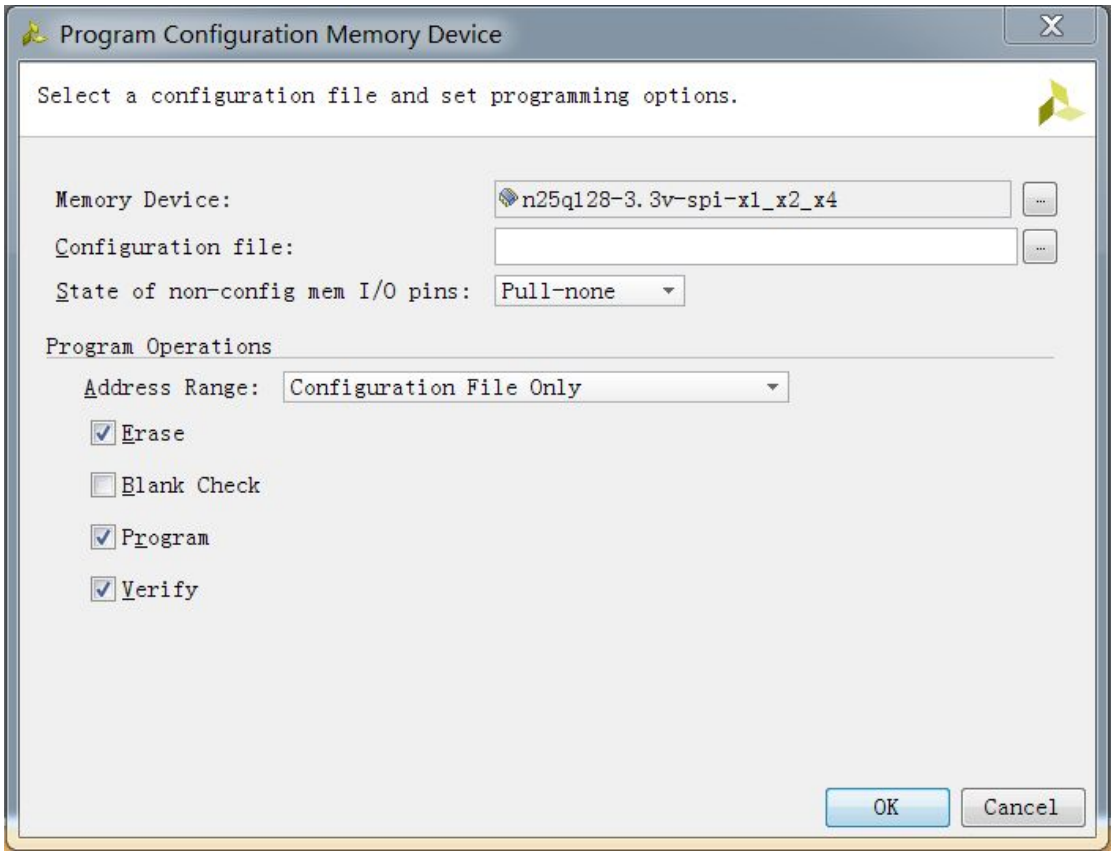
3.6. 右键板子，点击 Add Configuration Memory Device：



3.7. 选择正确的 memory 型号，点击 ok：



3.8. 选择正确的 mcs 文件，点击 OK 即可：



3.9. 至此即完成 FPGA flash 的烧录工作，流程全部完成，撒花~