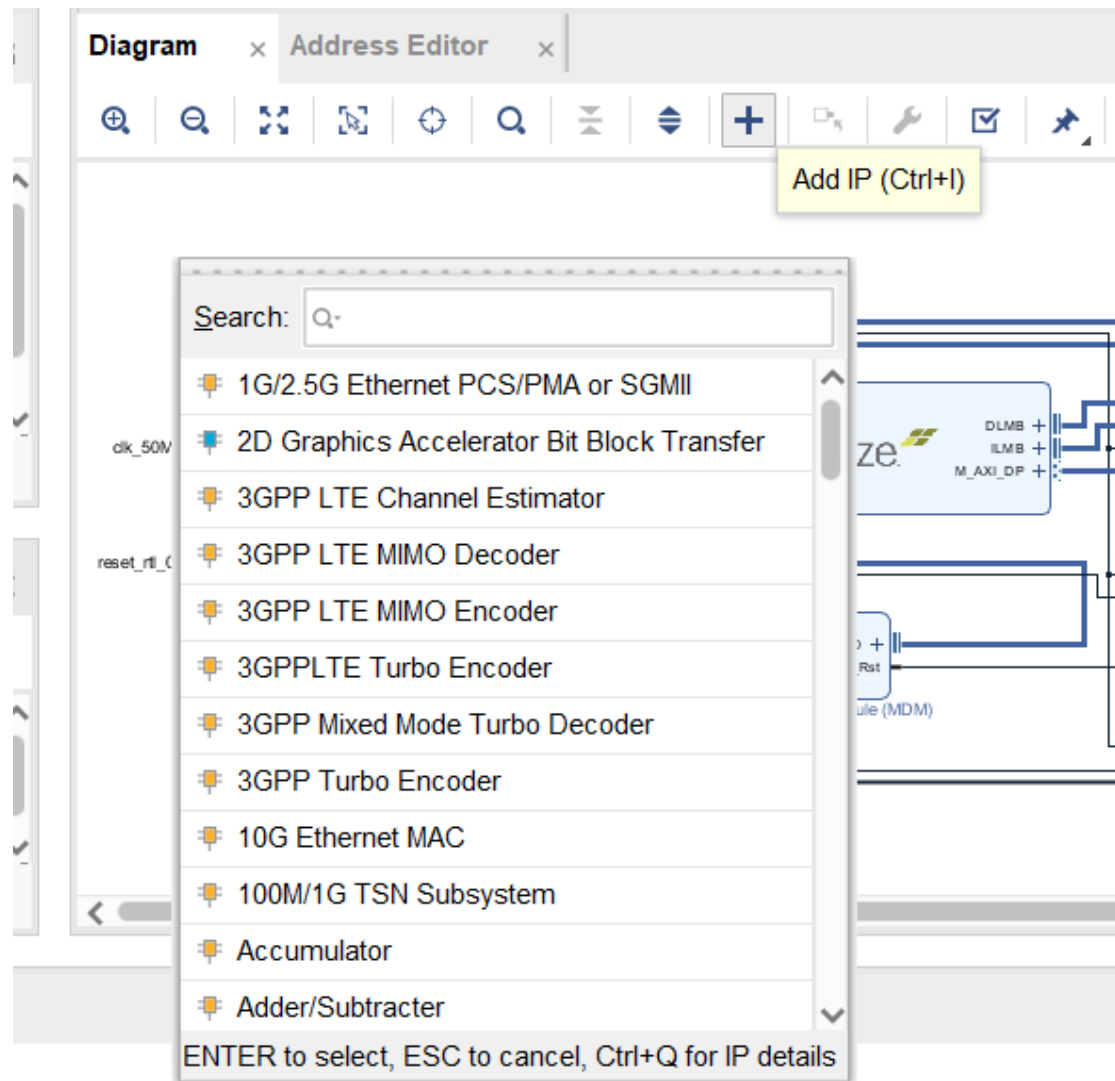


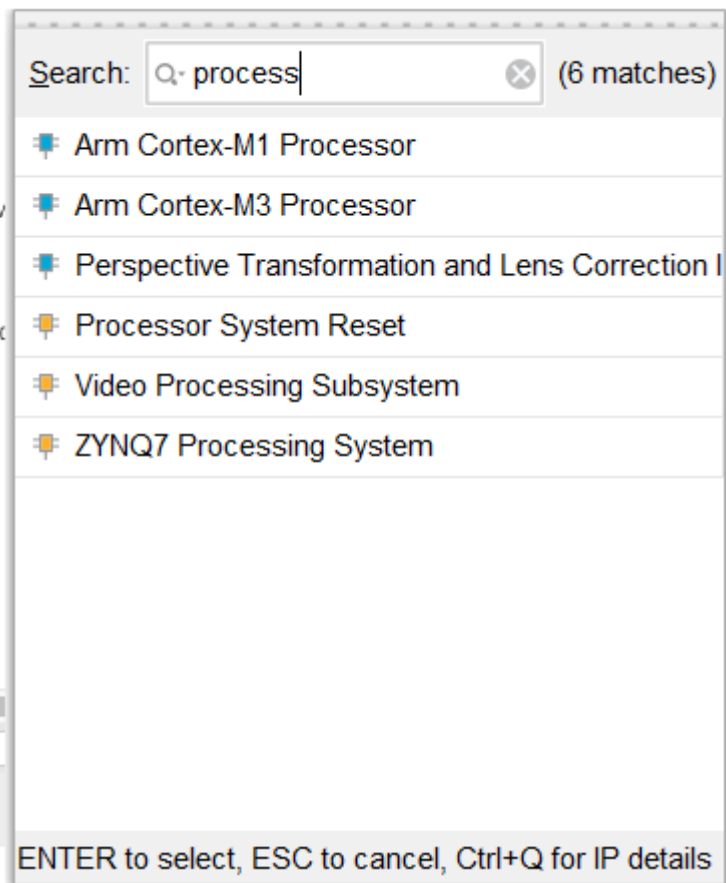
vivado2019+vitis2019 的 microblaze 软核生成

vivado 上的操作请参考老师发的 ic3\_soc\_demo.pdf 的前 9 页, 由于 2019 版本 vitis 存在 bug, 需要在 block diagram 中添加一个 ARM 硬核来完成编译, 操作如下:

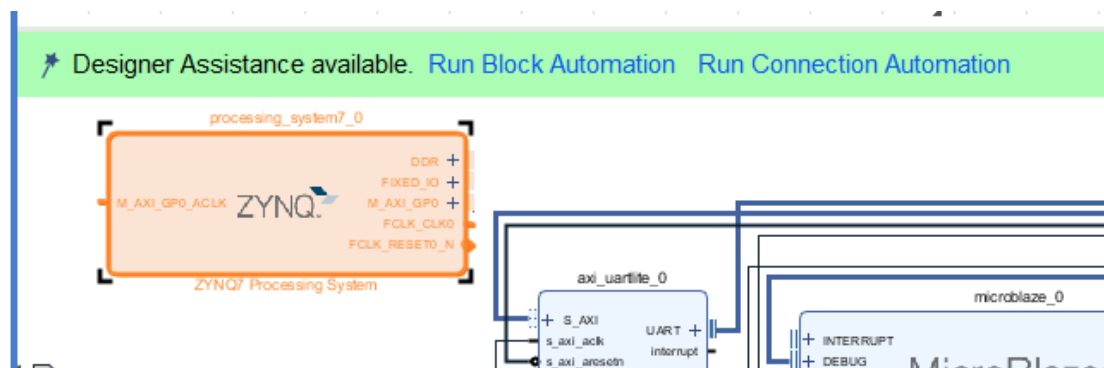
在做到 ic3\_soc\_demo.pdf 中第 7 页后, 点框图上方加号



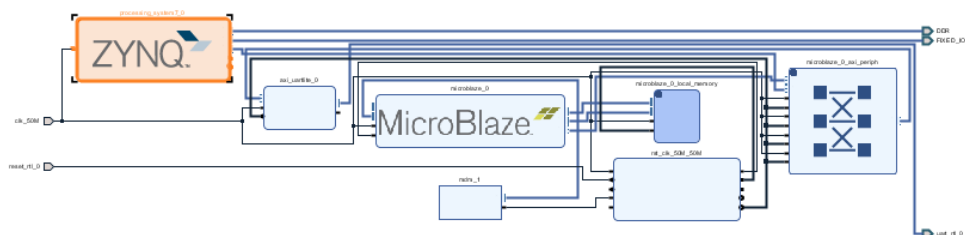
搜索模块 ZYNQ7 processor



添加后依次 run block automation（使用默认设置，直接点“OK”）以及 run connection automation，完成连线



连线后模块如下图, 最后的工程中仅调用这个硬核, 使编译正常进行, 并不使用它处理任务。

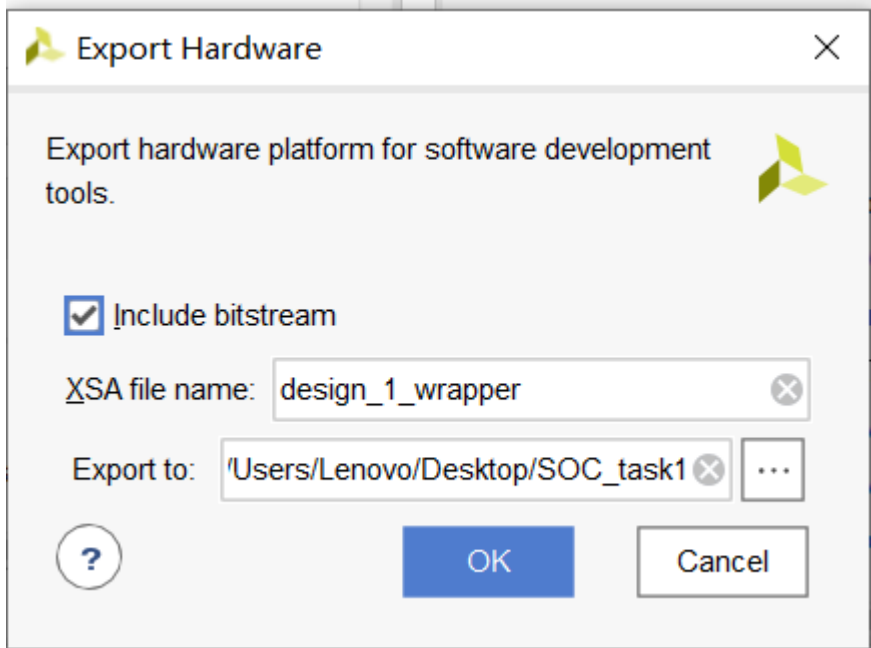


绑定引脚时注意使用 PL 端引脚, clk 使用 PL 端 50M 晶振, reset 按键也使用 PL 端按键。串

口 TX、RX 引脚绑定任意一个 PL 端 IO 即可。

All ports (4)							
CLK.CLK_50M_54576 (1)	IN			<input checked="" type="checkbox"/>	34	LVC MOS33*	3.300
Scalar ports (1)							
clk_50M	IN		U18	<input checked="" type="checkbox"/>	34	LVC MOS33*	3.300
RST.RESET_RTL_0_54576 (1)	IN			<input checked="" type="checkbox"/>	35	LVC MOS33*	3.300
Scalar ports (1)							
reset_rtl_0	IN		N15	<input checked="" type="checkbox"/>	35	LVC MOS33*	3.300
uart_rtl_0_54576 (2)	(Multiple)			<input checked="" type="checkbox"/>	35	LVC MOS33*	3.300
Scalar ports (2)							
uart_rtl_0_rxd	IN		B20	<input checked="" type="checkbox"/>	35	LVC MOS33*	3.300
uart_rtl_0_txd	OUT		C20	<input checked="" type="checkbox"/>	35	LVC MOS33*	3.300
Scalar ports (0)							

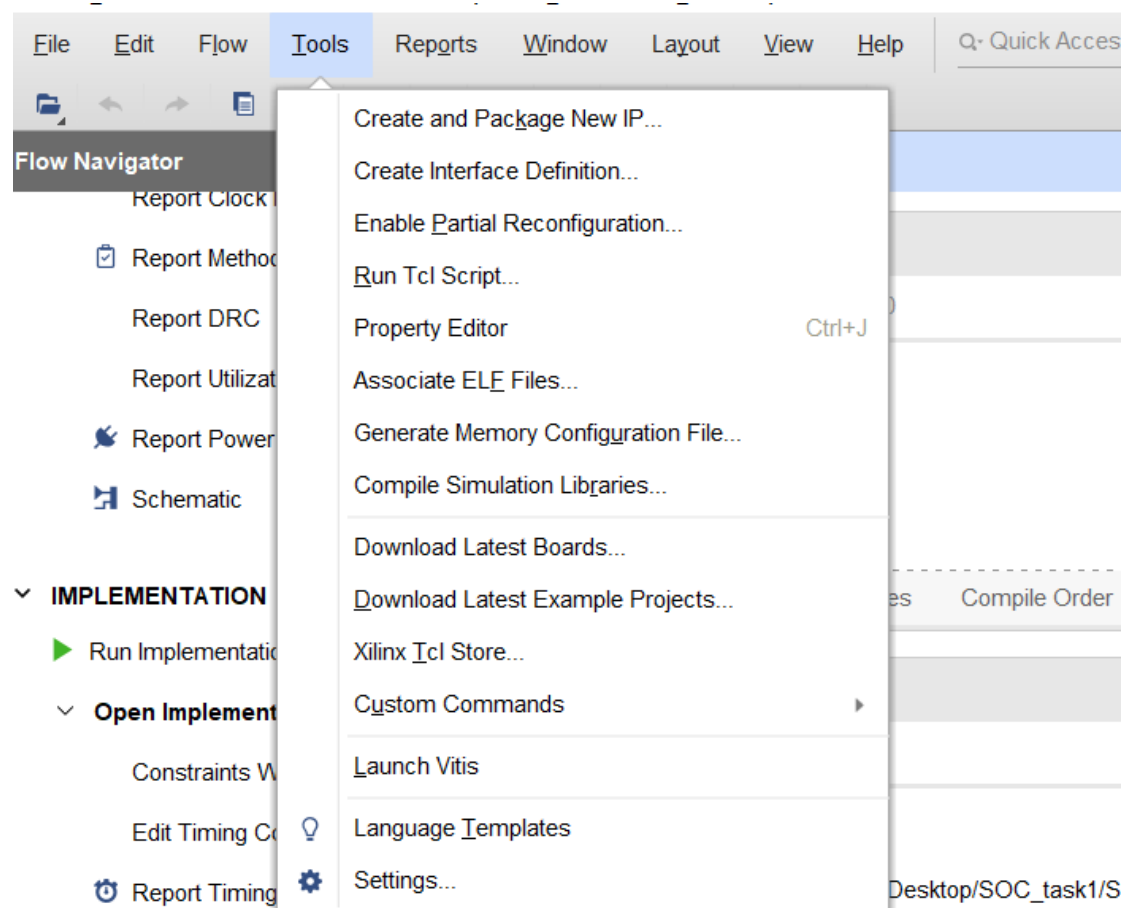
然后继续按照 PDF 中步骤做到第 9 页，在导出硬件时注意勾选 include bitstream 选框。



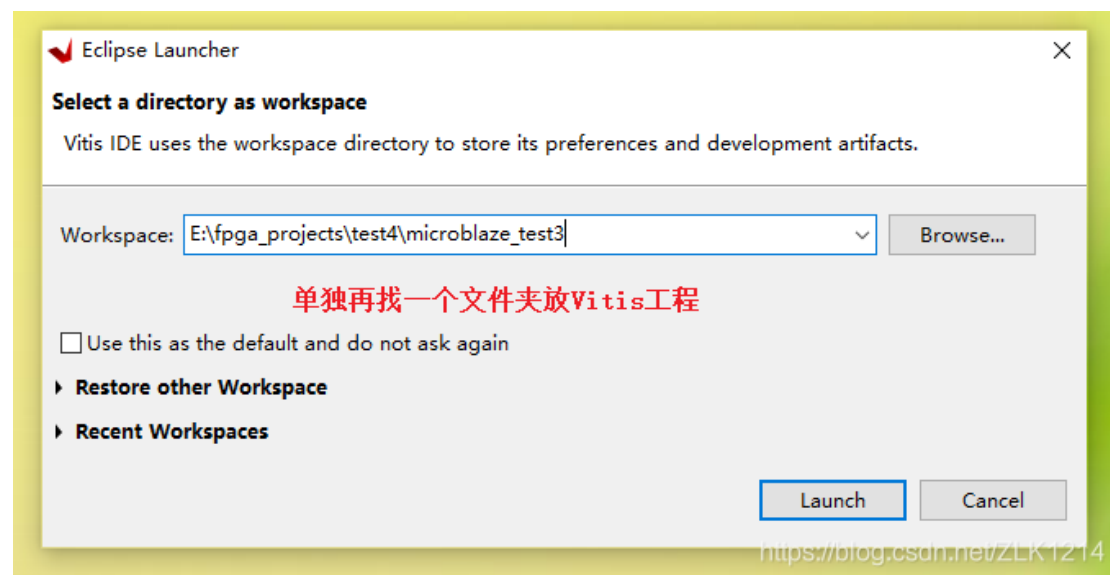
Vivado2019 及以上版本导出的 Hardware 文件为.xsa 后缀的打包文件，与 2018 以下版本的.hdf 不同

SOC_task1.srcs	2023/11/11 21:30	文件夹	
design_1_wrapper.xsa	2023/11/11 21:44	XSA 文件	120 KB
SOC_task1.xpr	2023/11/11 21:43	Vivado Project F...	39 KB

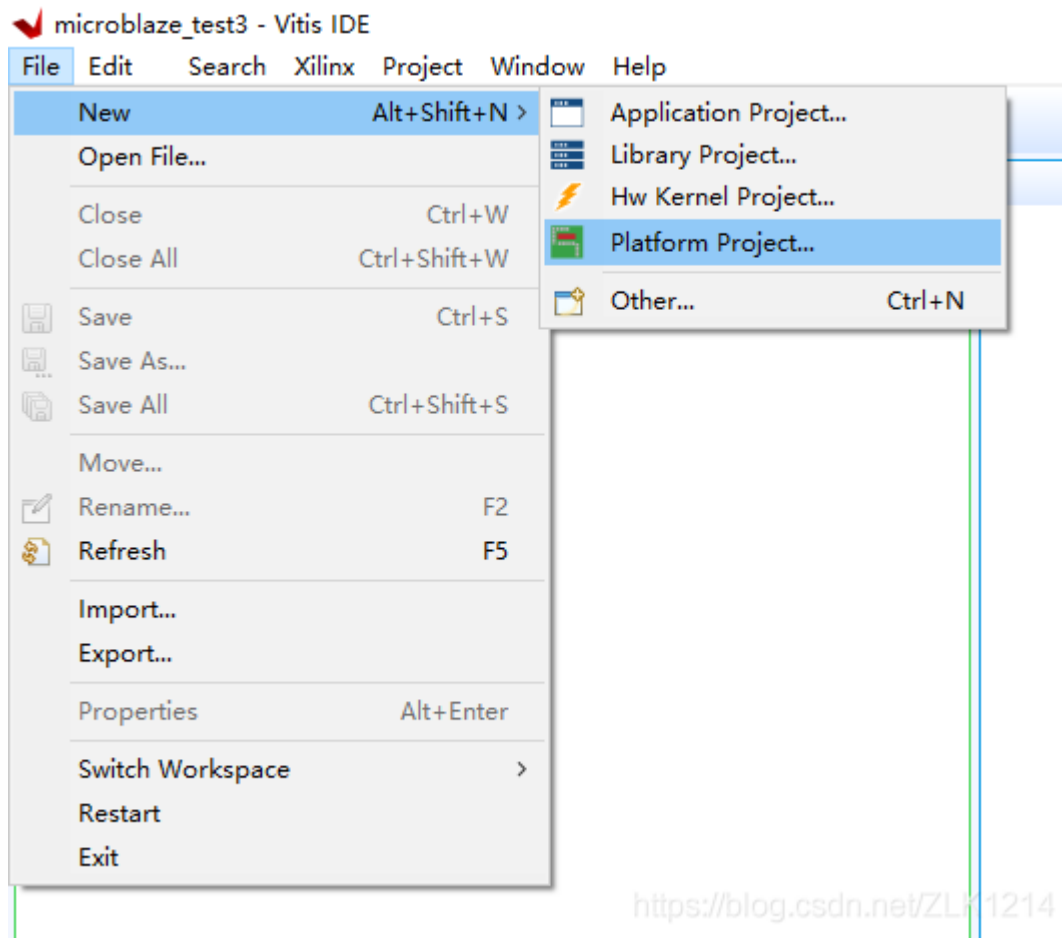
VITIS 部分



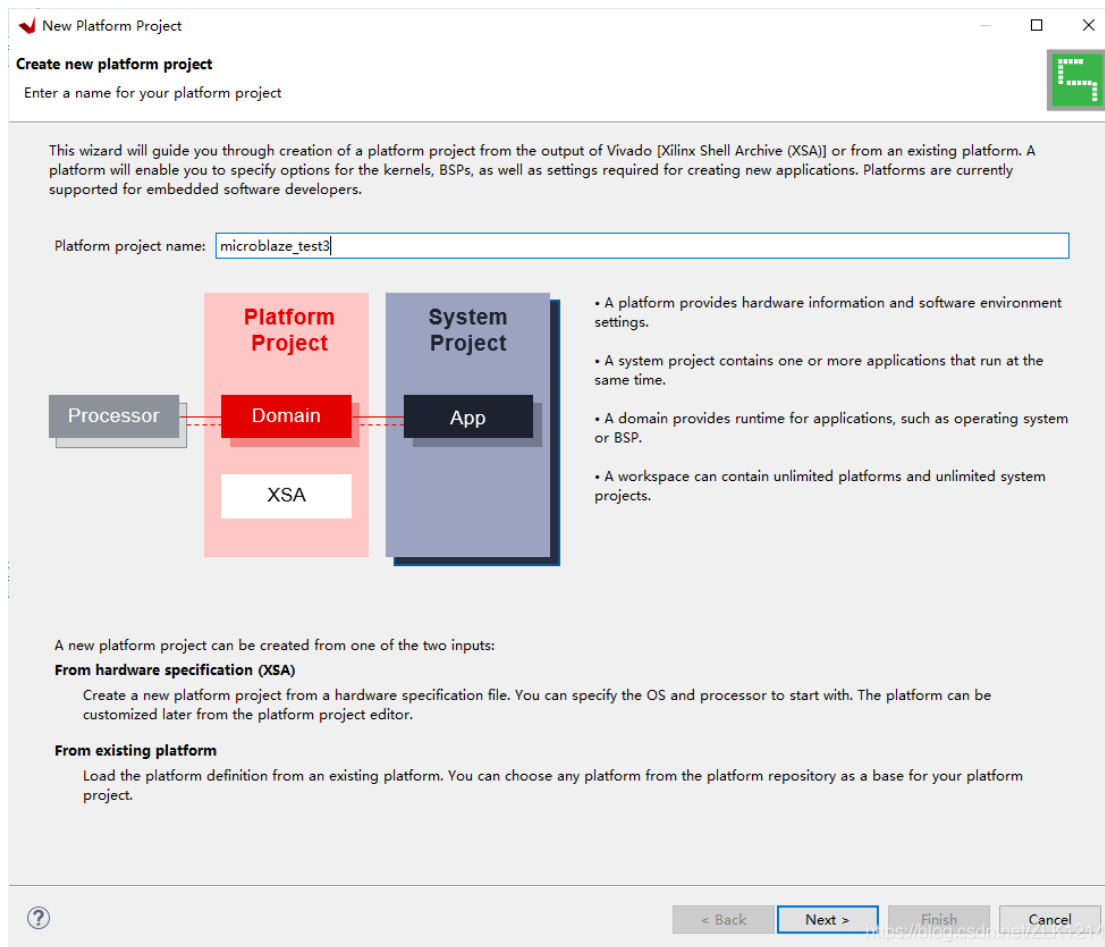
在 vivado 上方菜单栏的 tools 中选择 launch vitis，打开 vitis



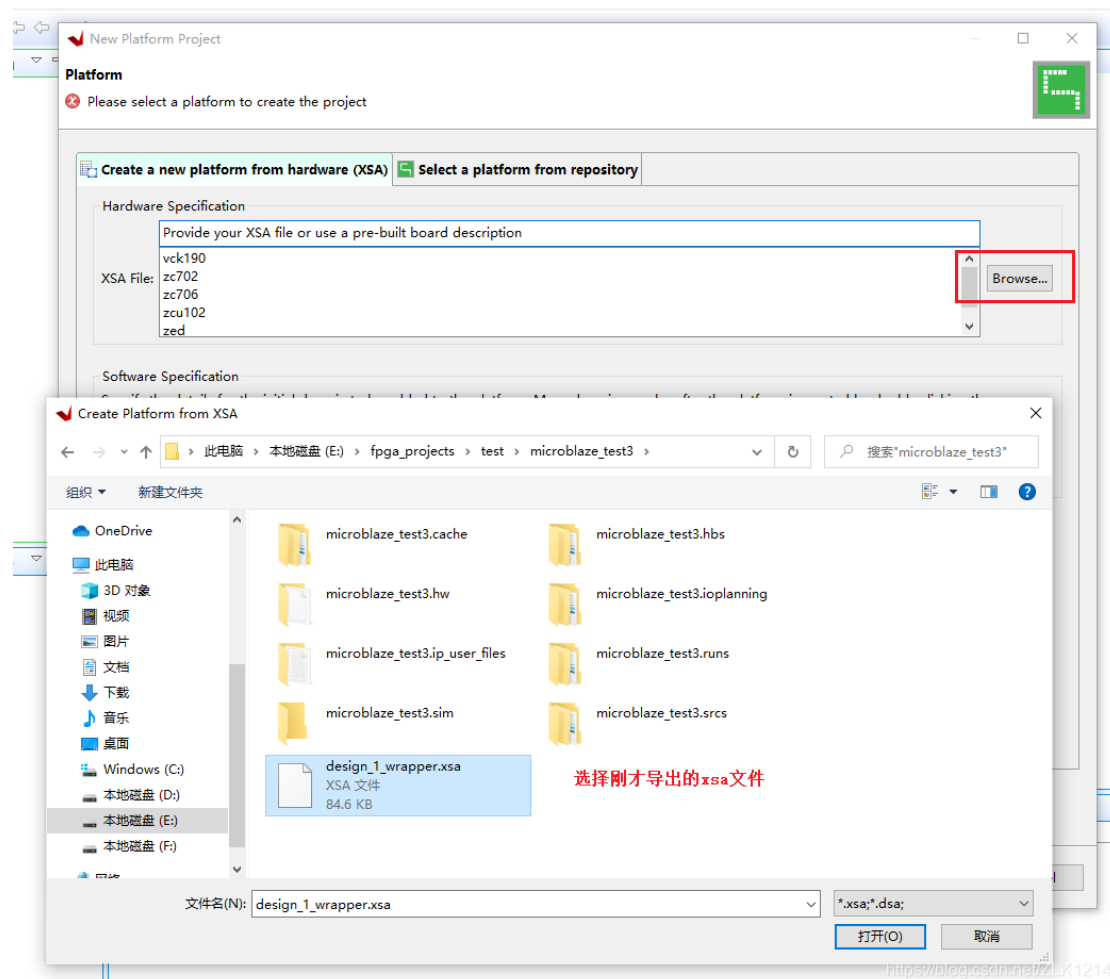
进入时会选择一个文件夹作为 vitis 的工程文件，在 vivado 的工程文件夹下新建一个空文件夹即可。



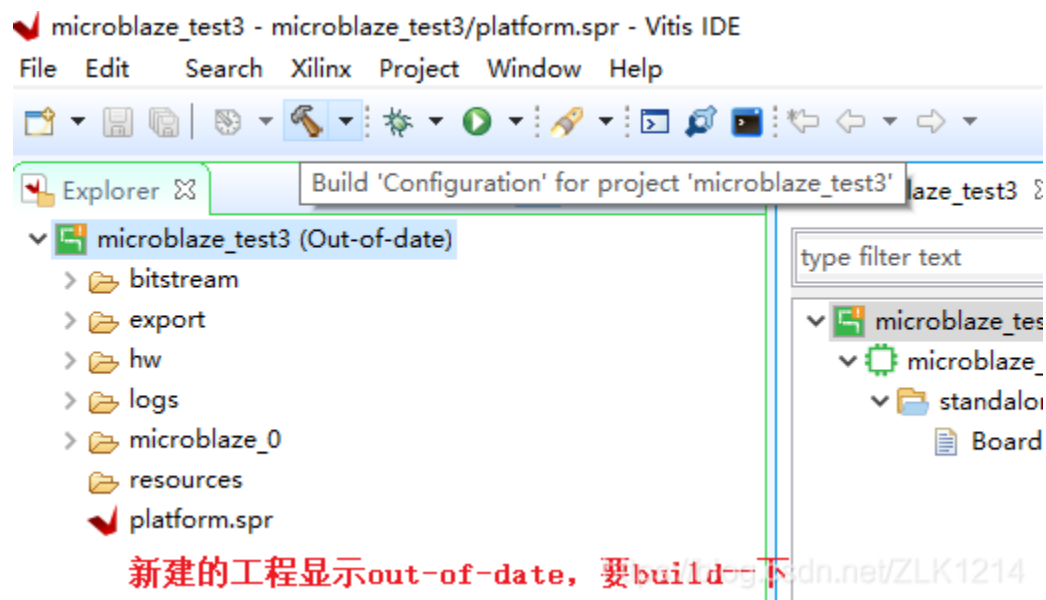
进去之后在 file->new->platform project 中新建平台工程，将 vivado 中设置好的软核包装成处理器，进入之后先命名。



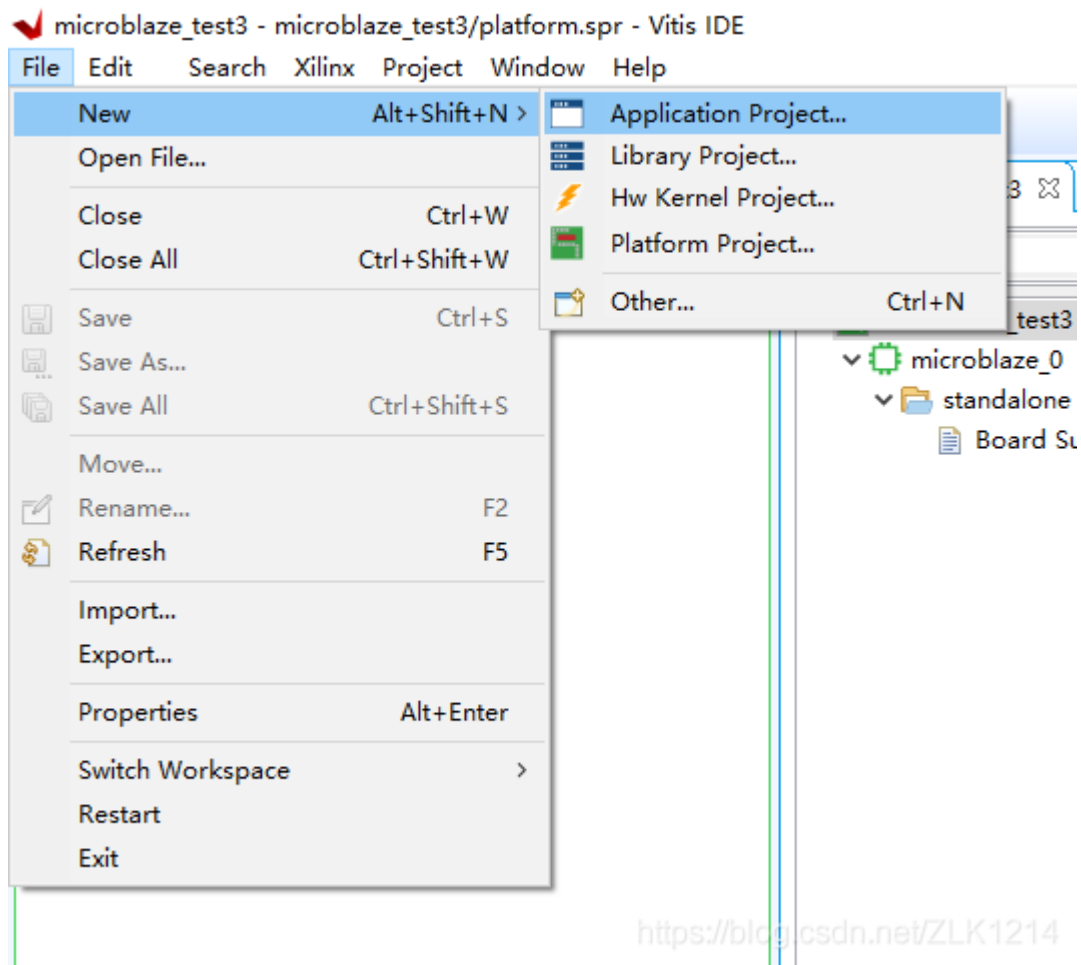
点击 next



选择刚才在 vivado 工程目录下生成的.xsa 文件，之后左侧会出现这个绿色的 platform 图标

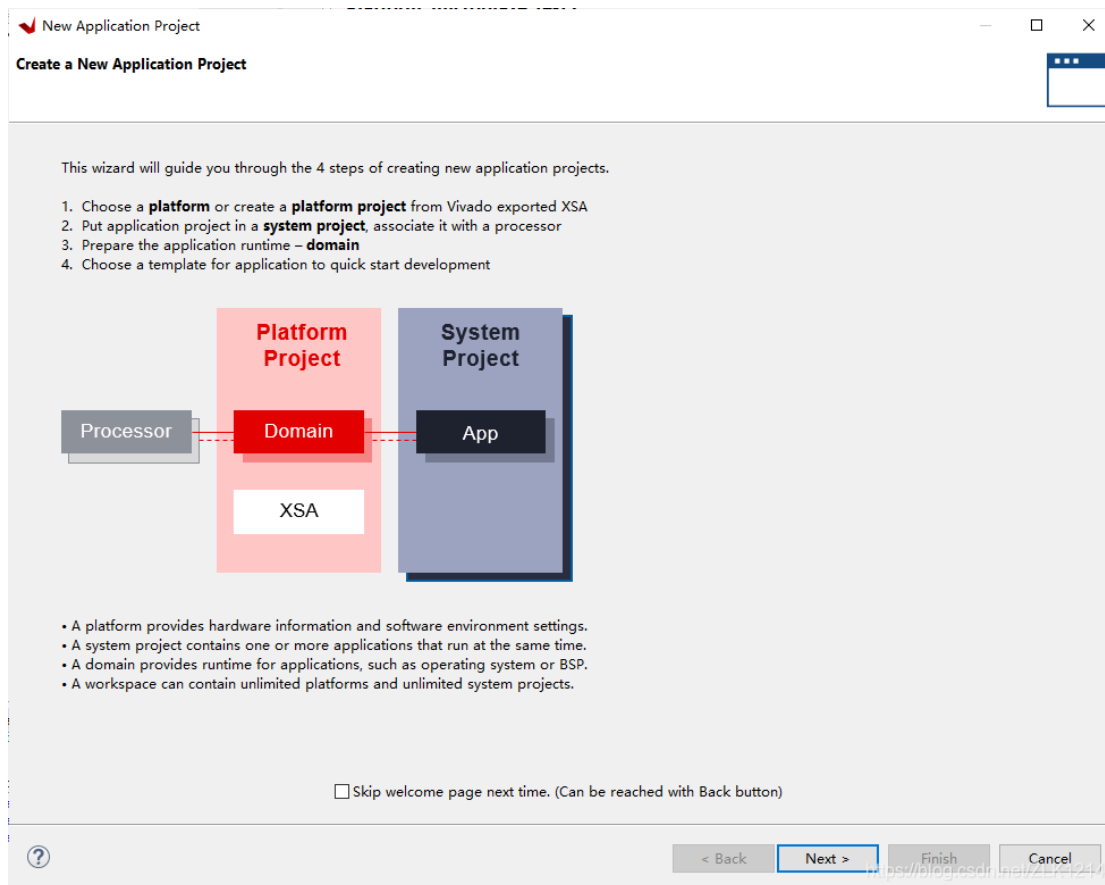


点击上面小锤子编译 platform

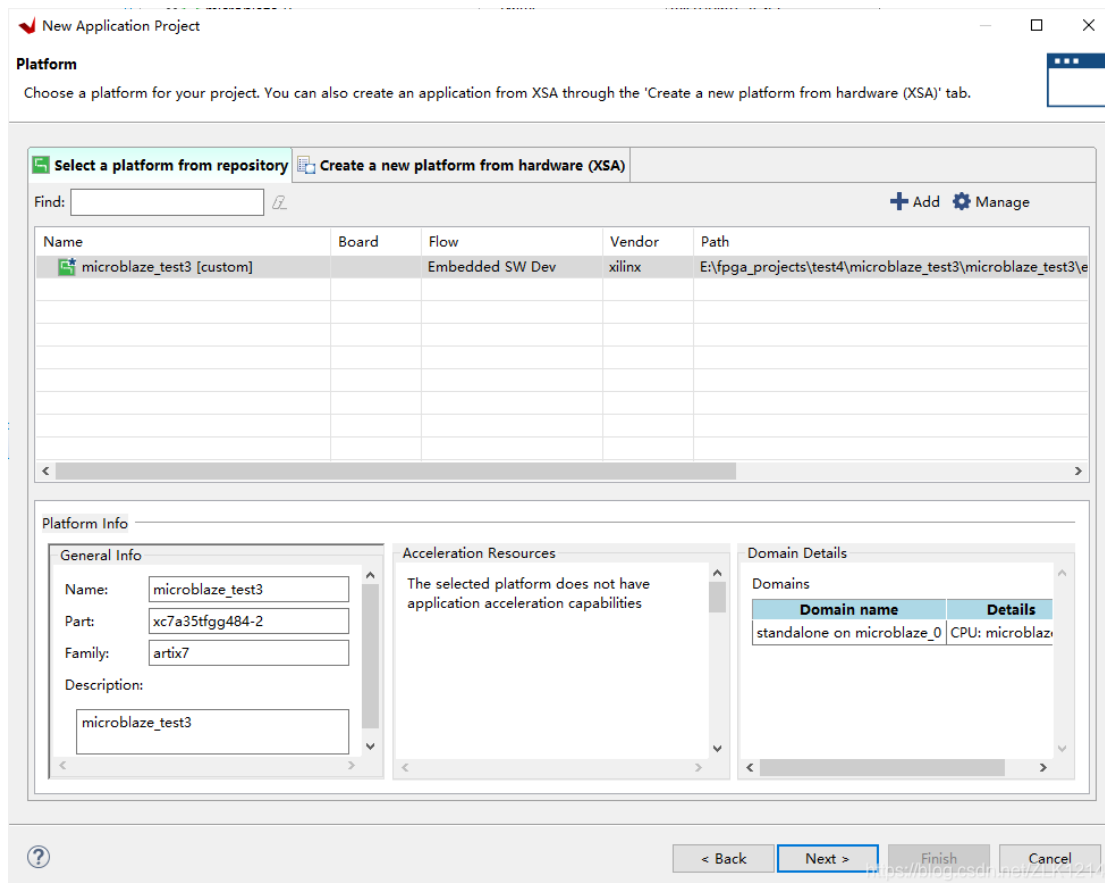


再新建 application project 作为这个 soc 的工程文件。





Next



选择刚刚生成的 platform 文件

New Application Project

Application Project Details

Specify the application project name and its system project properties

Application project name: hello\_world

System Project

Create a new system project for the application or select an existing one from the workspace

Select a system project

+ Create new...

System project details

System project name: hello\_world\_system

Target processor

Select target processor for the Application project.

Processor	Associated applications
microblaze_0	hello_world

Show all processors in the hardware specification ☐

< Back Next > Finish Cancel

命名它为 uart\_echo

New Application Project

**Domain**

Select a domain for your project or create a new domain

Select the domain that the application would link to or create a new domain

Note: New domain created by this wizard will have all the requirements of the application template selected in the next step

Select a domain

standalone on microblaze\_0

+ Create new...

Domain details

Name: standalone\_domain

Display Name: standalone on microblaze\_0

Operating System: standalone

Processor: microblaze\_0

< Back Next > Finish Cancel

Next

New Application Project

**Templates**

Select a template to create your project.

Available Templates:

Find:

SW development templates

Empty Application

Empty Application (C++)

Hello World

lwIP Echo Server

lwIP TCP Perf Client

lwIP TCP Perf Server

lwIP UDP Perf Client

lwIP UDP Perf Server

mba\_fs\_boot

Memory Tests

Peripheral Tests

SREC Bootloader

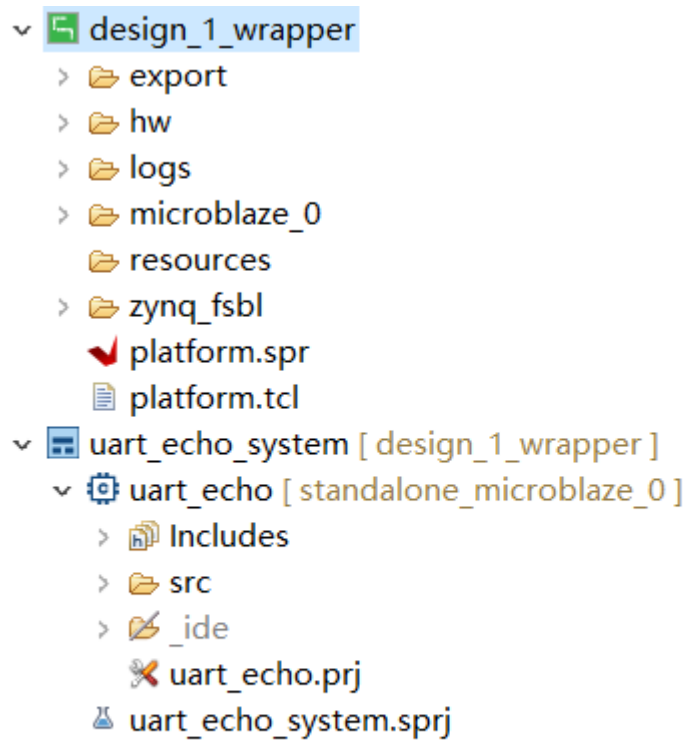
SREC SPI Bootloader

**Hello World**

Let's say 'Hello World' in C.

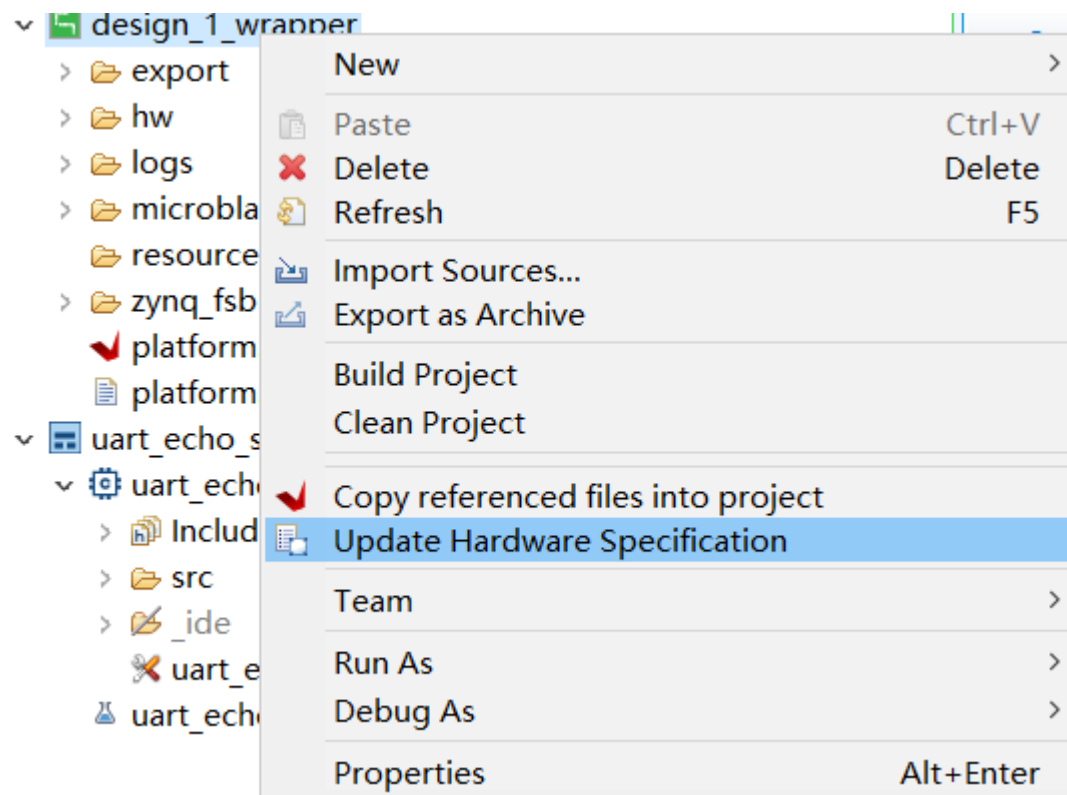
< Back Next > Finish Cancel

选择 empty application

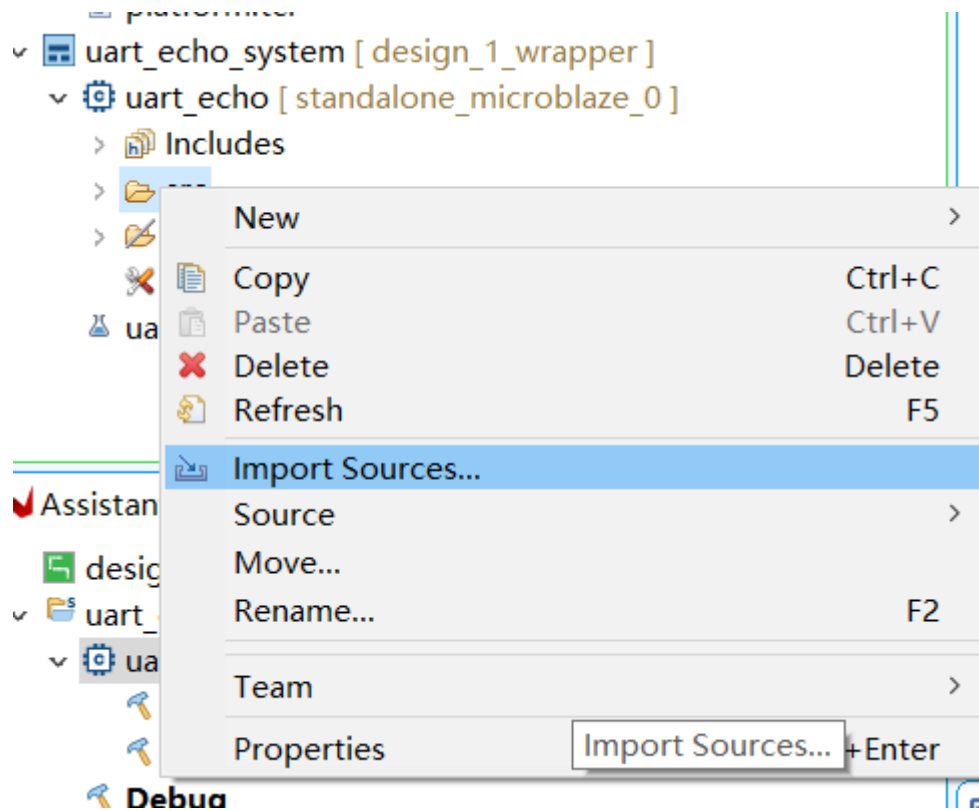


生成的目录如上图所示

PS: 如果后面在 vivado 中对.xsa 文件进行修改后重新 export, 请点击 update 更新 vitis 中的 platform



然后添加 uart\_echo.c 文件



这里我已将 pdf 中 c 代码 copy 下，编辑成 c 文件，可直接添加。没有文本编辑器的同学可以点 new->file 生成源文件

## Create New File

### File

Create a new file resource.


Enter or select the parent folder:

uart\_echo/src

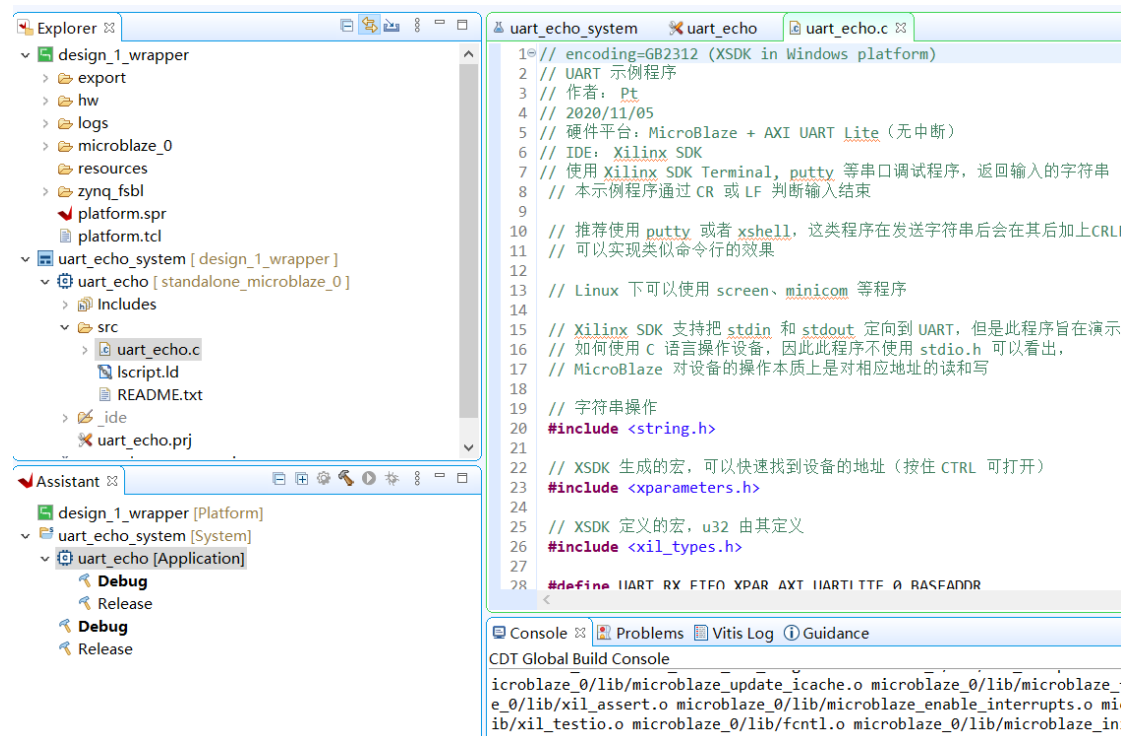
- > design\_1\_wrapper
  - RemoteSystemsTempFiles
  - ▼ uart\_echo [ standalone\_microblaze\_0 ]
    - > \_ide
      - src
    - uart\_echo\_system [ design\_1\_wrapper ]

File name:

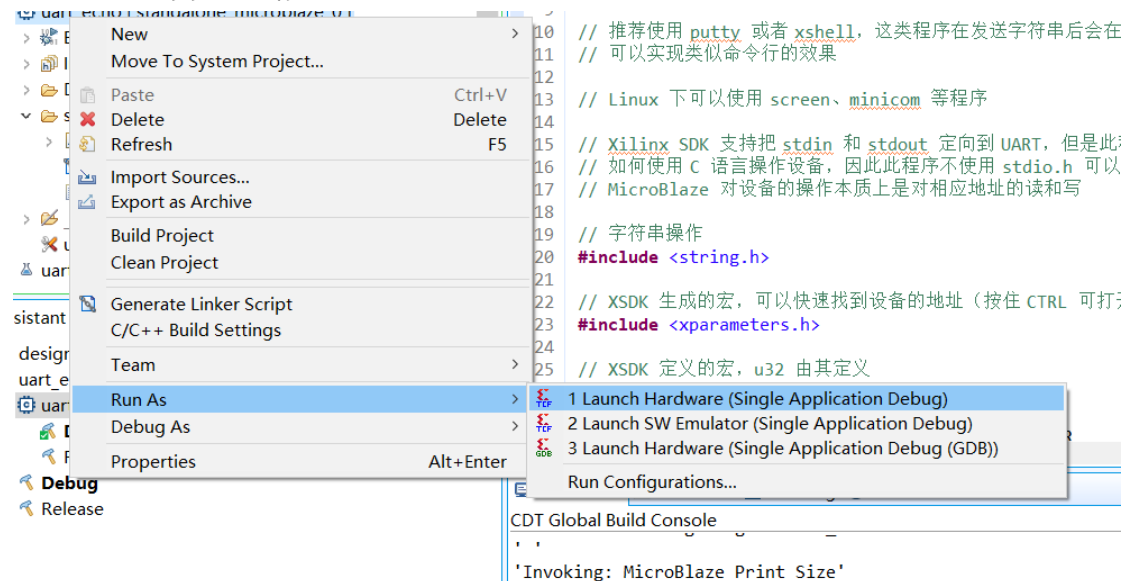
Advanced >>



在出现的编辑器中修改复制下的代码



再次点击上方的小锤子编译



右键 `uart_echo` 工程, 选择 `launch hardware`, 此时连接上 fpga 会直接对它编程。

在连接 ttl 转 USB 串口模块 (长得像 U 盘的那个东西) 时, 注意串口模块上的 TX 连接在 FPGA 上定义的 RX 引脚上, RX 连在 FPGA 定义的 TX 引脚, 并将串口模块的 GND 引脚和 FPGA 的 GND 相连, 否则收不到也发不出。

另外, 串口通讯助手注意选择“发送新行”, 因为串口接收函数是通过判断发送字符串结尾的回车符, 判断接收完毕, 否则无法进行回环通讯测试。另外一个有趣的现象是由于换行“\n”和回车符“\r”分别对应两个 HEX 字符 0x0A 和 0x0D, 都可以触发串口接收完毕, 因此选择“发送新行”会触发两次串口接收完毕, 在回环测试时会出现发送一个空行, 这是正常现象。