# 一、简介

以最简单的变量加法器作为例子，从**HLS**到**Vivado**再到**SDK**走一遍流程。

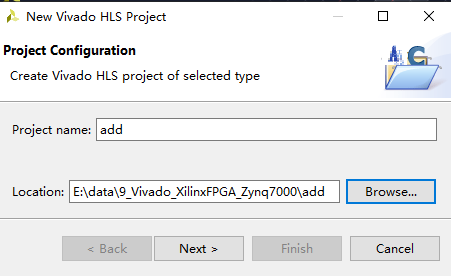
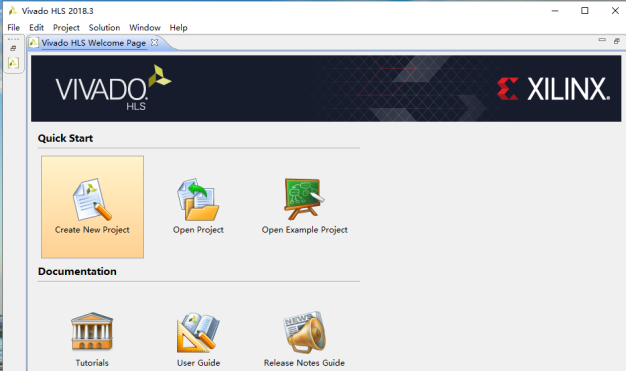
# 二、HLS设计

首先要做的是在Vivado HLS上设计我们所需要的加法器IP核。

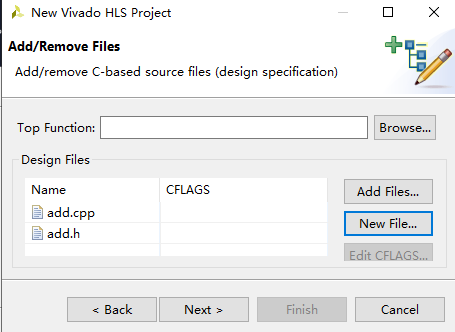
这里科普一下IP核：IP核就是知识产权核或知识产权模块的意思，在EDA技术开发中具有十分重要的地位。美国著名的Dataquest咨询公司将半导体产业的IP定义为“用于ASIC或FPGA中的预先设计好的电路功能模块”。IP主要分为**软IP、固IP和硬IP**。软IP是用Verilog/VHDL等硬件描述语言描述的功能块，但是并不涉及用什么具体电路元件实现这些功能。固IP是完成了综合的功能块。硬IP提供设计的最终阶段产品——掩膜（来自百度百科）。我们平时做C程序设计总是会自定义一些子函数方便主函数的调用并使得主函数更精简，这些函数一般都是在CPU上跑的，而IP核可以理解成在硬件上通过电路实现出来的子函数，起着数据处理或交换等作用。

言归正传，我们从最基本的开始一步一步做。

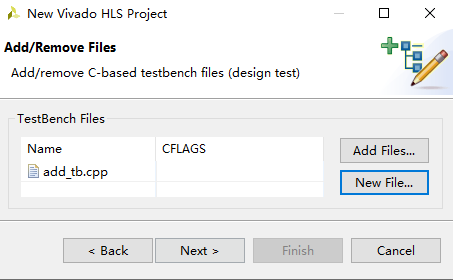
## 1、创建工程



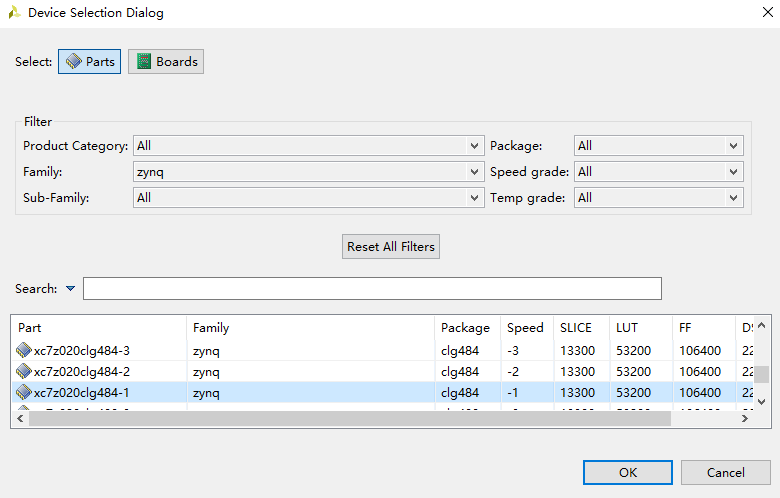
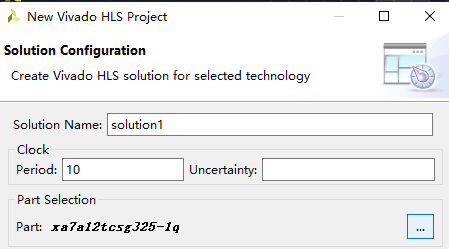
添加新源文件和头文件：



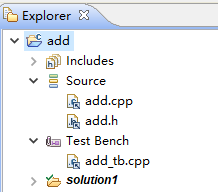
然后添加testbench测试文件，这在做C综合过程中也是必不可少的，添加方法更上面添加顶层文件和头文件是一样的。



选择硬件：



创建成功后，如下图所示：



## 2、程序设计

add.cpp文件下添加代码：

int add(int a, int b)

{

return (a+b);

}

add.h下添加代码：

#ifndef add\_h\_

#define add\_h\_

int add(int, int);

#endif

add\_tb.cpp下添加代码

#include "add.h"

#include <iostream>

#include <cstdio>

#include <fstream>

int main()

{

int a, b;

a = 5;

b = 10;

int res;

res = add(a, b);

if(res == 15)

{

printf("OK");

return 0;

}

else

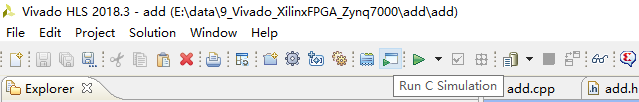
return 1;

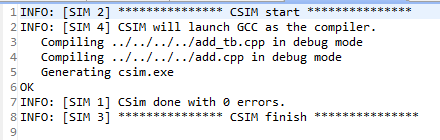
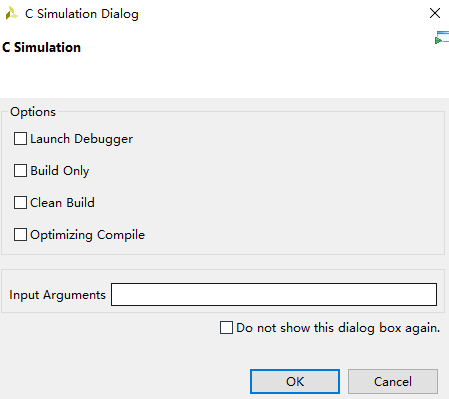
}

testbench文件中一般都会有一个验证顶层函数是否正确的语句，比如这里我用的 res == 15表示加法器的输出与预想一致，那么返回值为0。如果一个 main 函数的返回值是0的话，程序就是正确的，如果返回值是1的话（条件是我们自己设定的），程序在做C仿真的时候是会出现错误的。

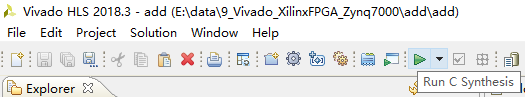
## 3、四步曲

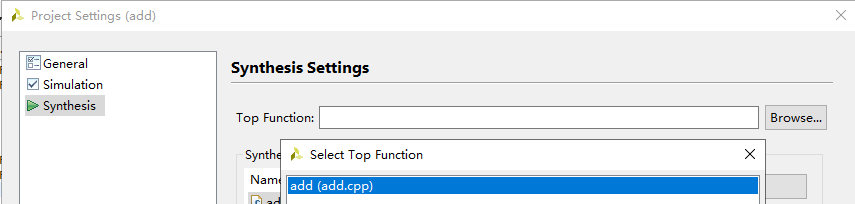
**（1）C仿真：**



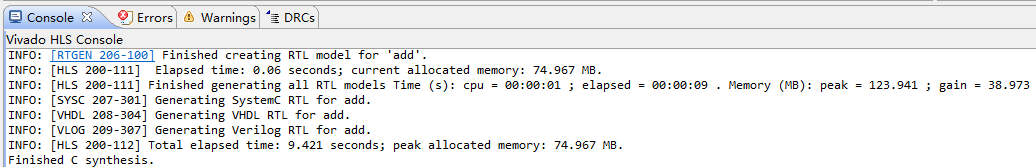


**（2）C综合：**

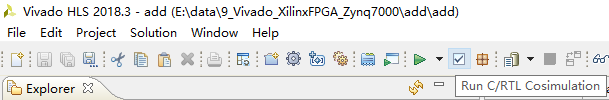




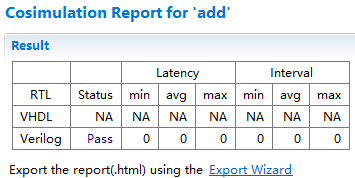
设置完再次点击综合按键：



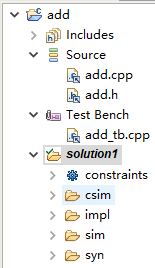
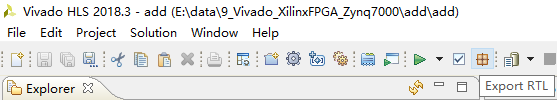
**（3）C/RTL联合仿真：**



联合仿真之后出现下面的结果就是没有问题的。



**（4）导出RTL（Export RTL）也就是我们一直在说的IP核：**

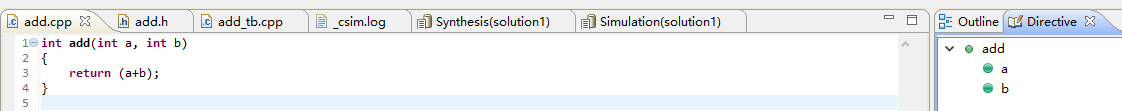


上面右边的图片csim\impl\sim\syn分别对应四个步骤。

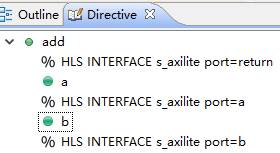
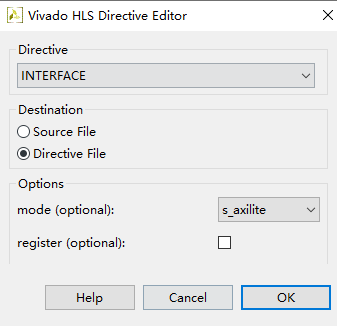
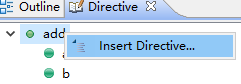
## \*、HLS番外

上面的设计存在一个问题，就是这样综合出来的一个 IP核，ZYNQ处理系统是没法去用它的，因为数据不知道怎么放到IP核的输入，更不知道怎么把输出取出来，因此这里加入一个番外，阐述最重要的一件事情——添加AXI接口协议。这一块在初学的时候查阅了很多资料，大多都没有提及，因此走了很多弯路，这里单独放到一个番外章节旨在提醒大家不要忘记这个重要的步骤。

点击到 add.cpp 顶层文件，再点击 Directive （我将其称为综合引导，用来引导HLS按照我们希望的综合方向去走），如下图所示。

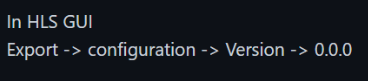
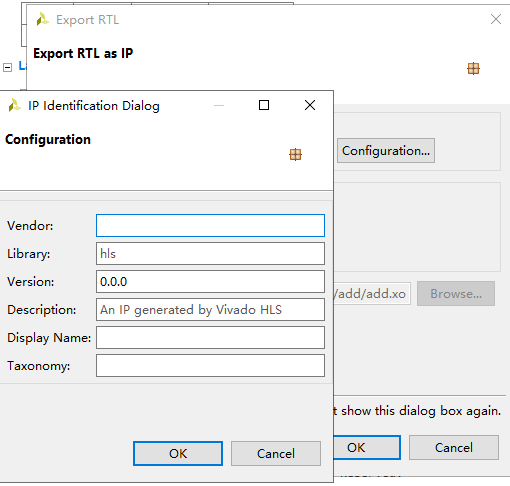


然后点击 Directive 下面的三个绿点对应到的对象，分别是 add 顶层函数（顶层模块）、变量a和变量 b，分别用右键点击这三个对象，然后点 Insert Directive… 。然后按照下图选择 Directive 类型 INTERFACE ，再选择 mode 类型为s\_axiite。这里选择接口协议为AXI协议，关于AXI协议是什么之后再补充。



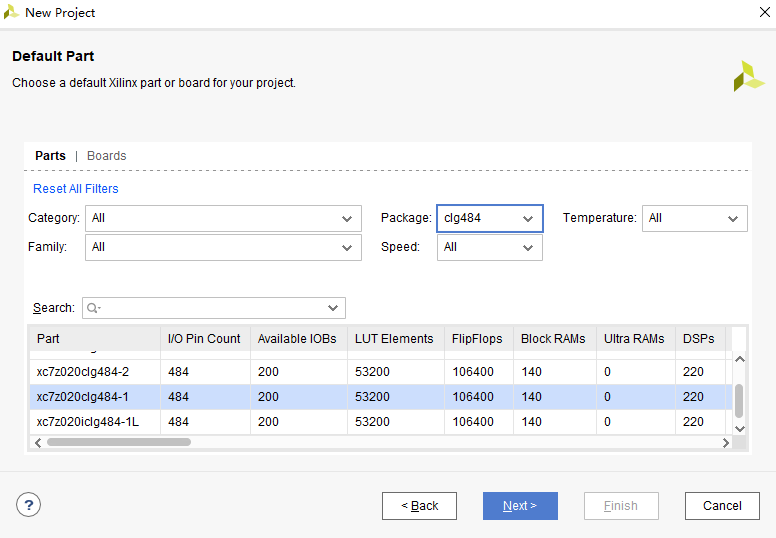
**最后再做一下四步曲就可以了，其实可以直接做C综合和导出RTL，但是保险起见，而且速度也很快，我们就都做一遍。**再补充一点，C仿真、C综合和导出RTL是必不可少的，但是C/RTL联合综合对一些比较大的模块可能会非常耗时，如果读者对自己写的程序有信心的话，其实是可以跳过的。现在我们已经有了加法器的IP核了，接下来就是把它用起来。

**【注】：导出RTL要修改版本号。**



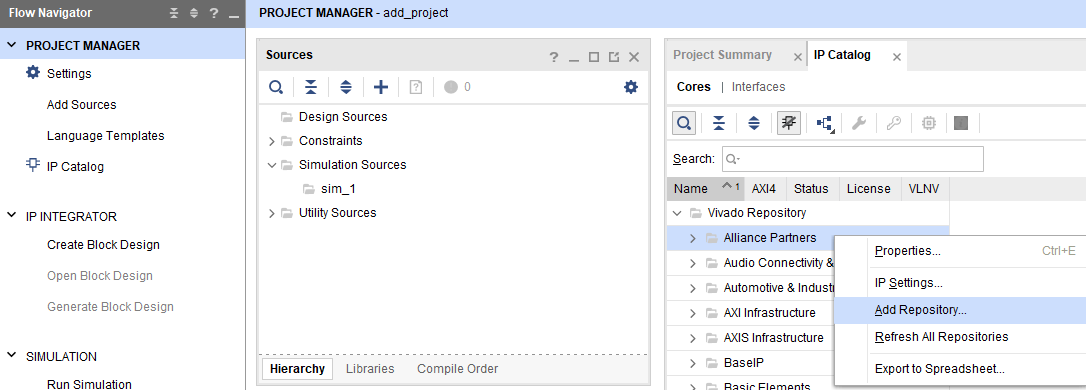
# 三、Vivado设计

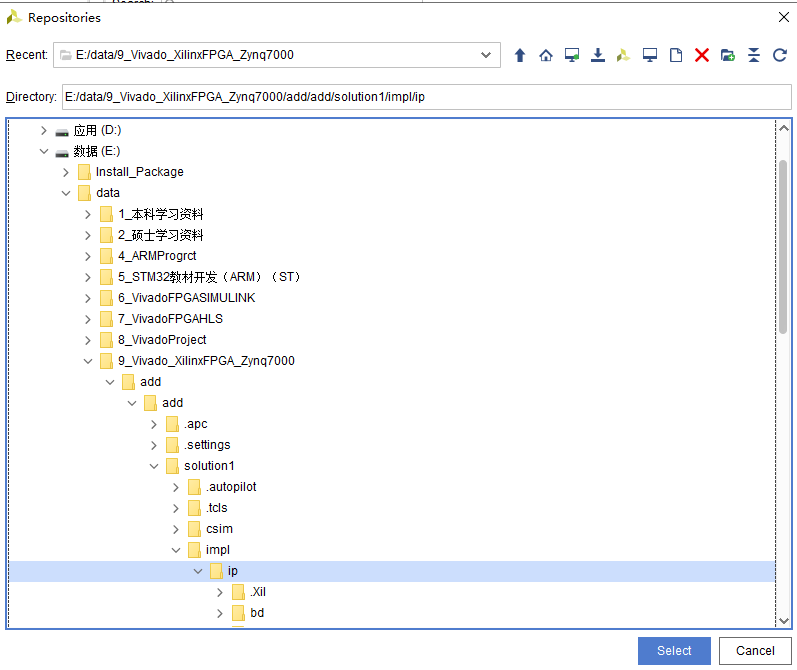
## 1、创建工程

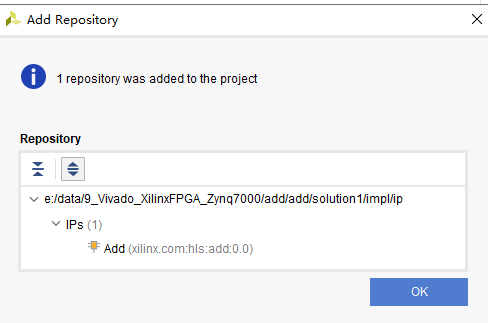


## 2、硬件电路连接

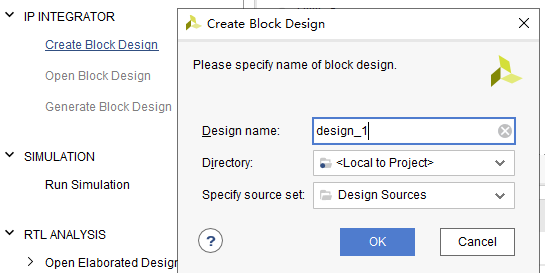
第一步，把 自定义IP核 添加到 IP目录 中，如下图点击。



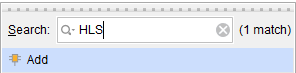
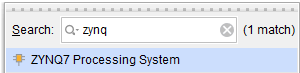


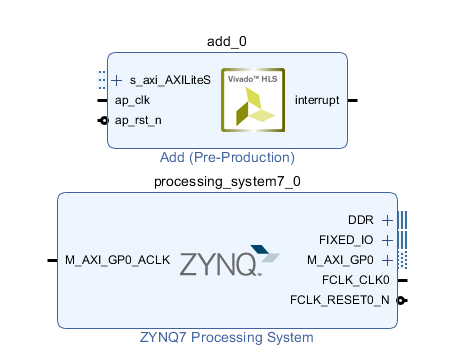


第二步，创建可视化的顶层文件，并连接电路。如下图所示点击 Create Block Design ，然后弹出对话框，不用改名字，直接点击 OK ，完成 Block Design 的创建。

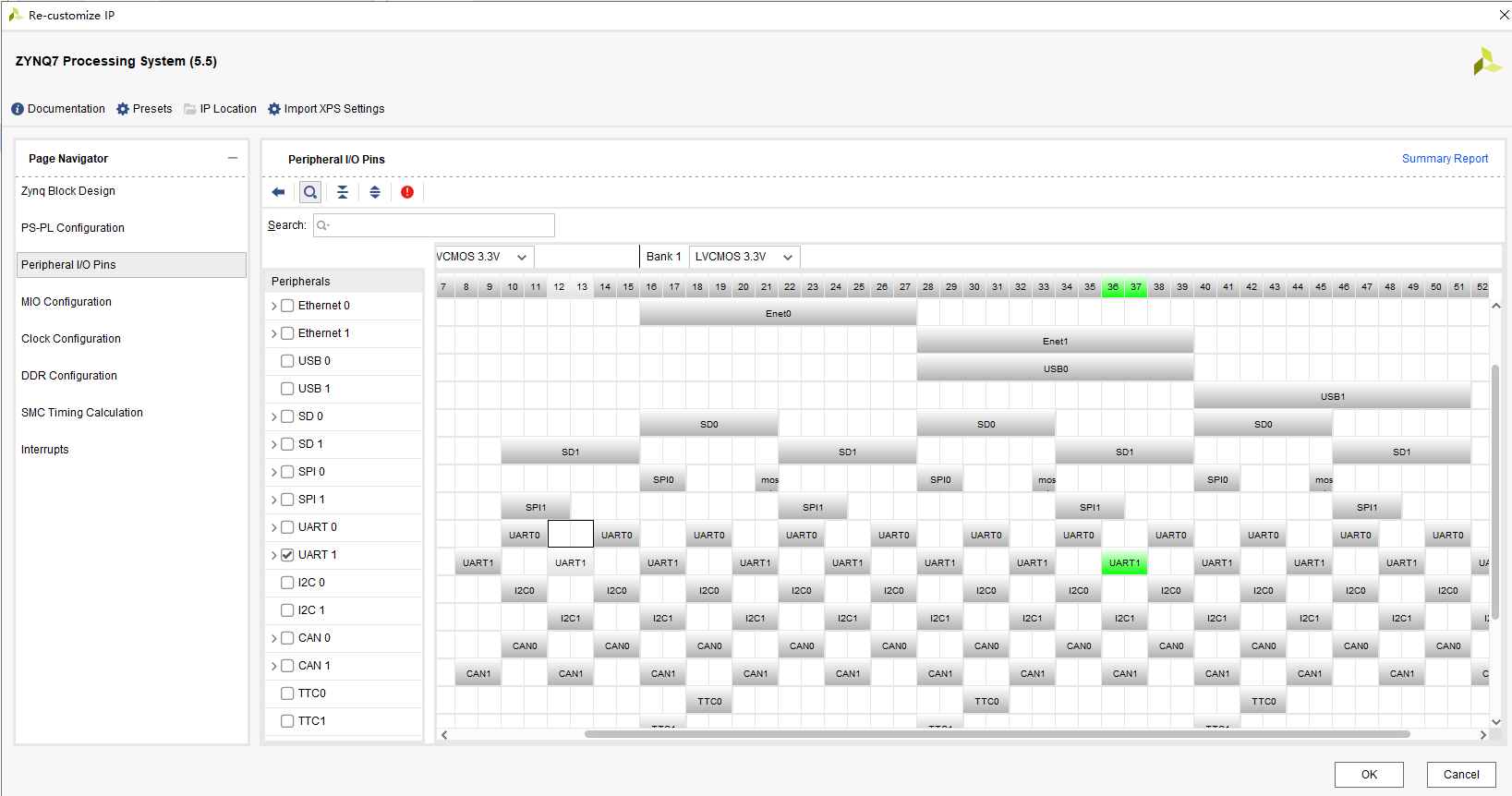


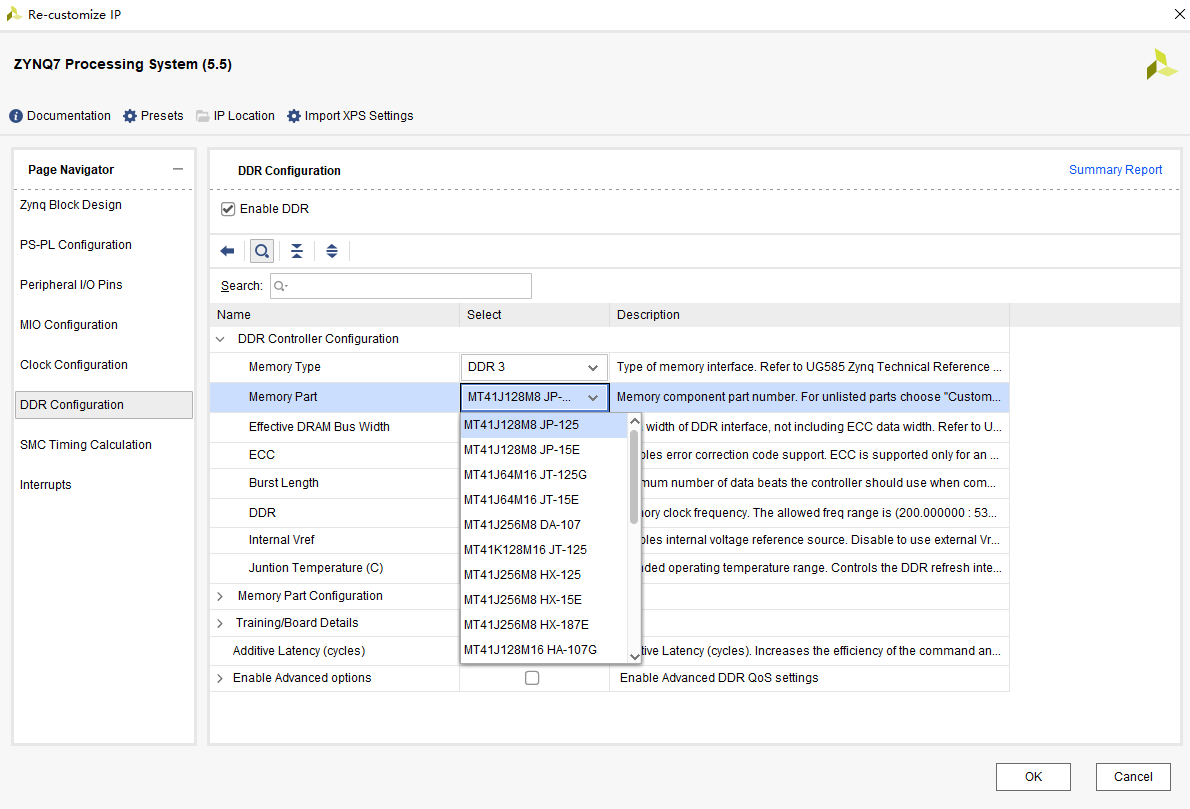
添加IP核：



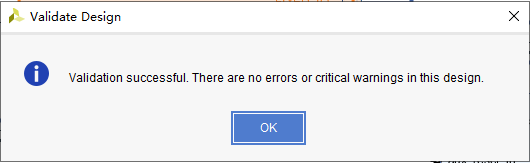
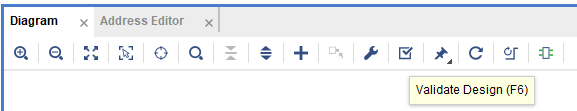


双击 ZYNQ7 Processing System ，配置 串口通信UART ，用于在PC上与硬件通信；配置 DDR3 ，用于数据的存储，根据硬件上 DDR 型号选择最相近的即可。如下图所示。最后点击 OK 完成配置。



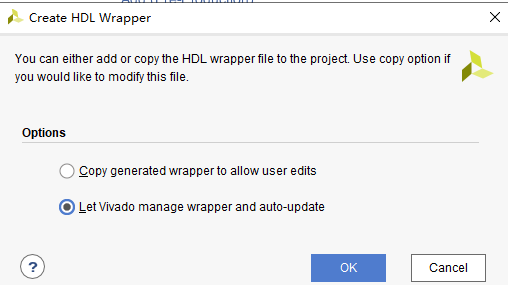
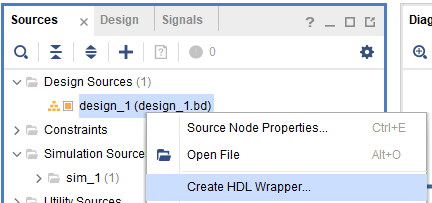


回到 Diagram 界面，分别点击 **Run Connection Automation** 和 **Run Block Automation** ，完成自动连线和引出必要端口。得到如下电路图，然后点击 Validate Design 验证 Block Design 是否有错误。出现以下信息就是电路验证通过了，接下来的操作就是走走流程就可以了。

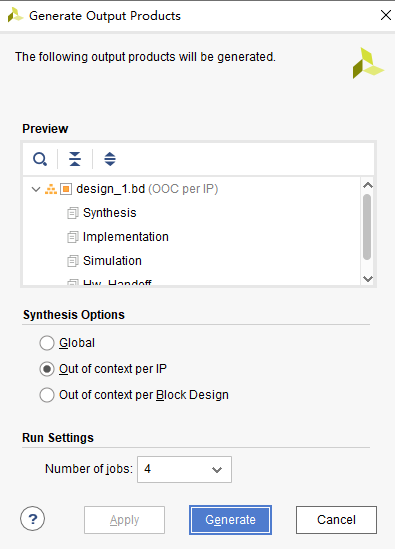
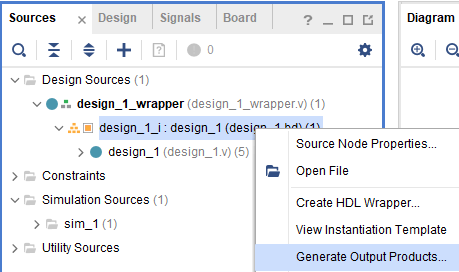


## 3、硬件信息输出

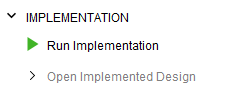
第一步，创建HDL封装（Create HDL Wrapper），按照下图从上到下依次点击，然后在出现的对话框点击 OK 即可。



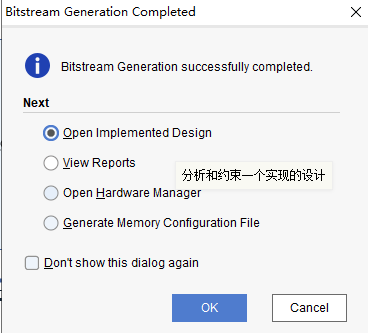
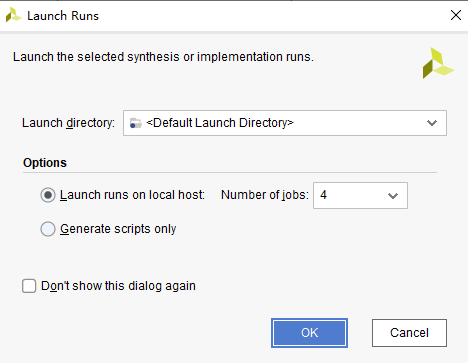
第二步，生成硬件输出（Generate Output Products），如下图所示。弹出的 Generate Output Products 直接点击 Generate 即可，对话框中的 Number of jobs 表示想用几个CPU线程去完成这个输出，线程越多速度生成硬件就越快。



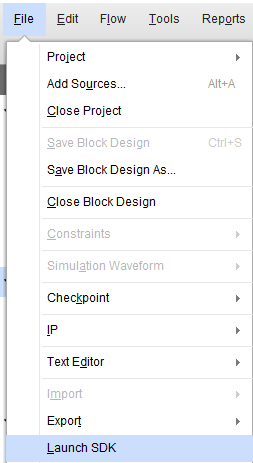
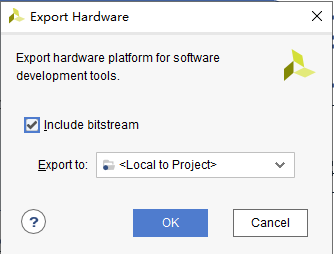
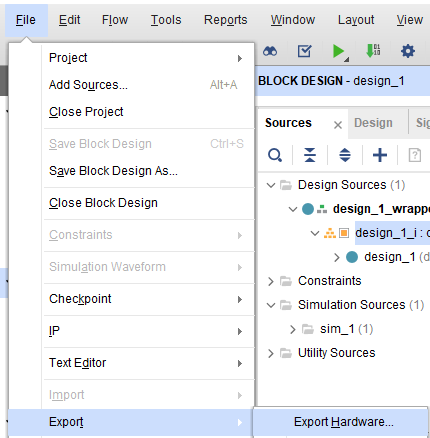
第三步，点击 Run Implementation 运行项目，对话框一直点击 OK 即可。



第四步，第三步完成后会出现 Implementation Completed 对话框，可以看到设计的硬件实现的可视化结果图（点击 Open Implemented Design 再点击 OK ），也可以不看，直接生成含有硬件信息的 bit流（Generate Bitstream），如下图所示。可以直接在对话框中生成，也可以到左侧 Flow Navigator 导航栏中去操作生成，弹出的 Bitstream Generation Completed 对话框可以不看，直接点击 Cancel ，读者有兴趣的话也可以自行去点开看一下结果。



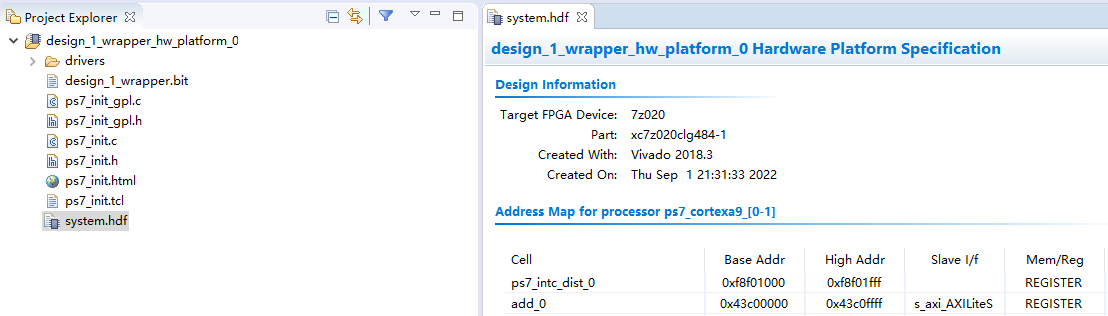
第五步，导出硬件信息（Export Hardware），按照下图点击，弹出的 Export Hardware 对话框把 Include Bitstream 选上，然后点 OK 。



搞定，Vivado软件上的操作到此为止，接下来就是SDK的操作了。直接从Vivado软件中运行SDK，如上图，弹出的Launch SDK对话框直接点击OK。

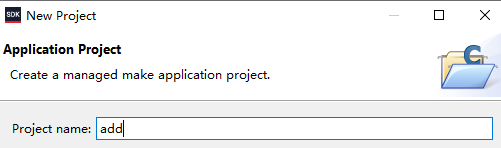
# 四、SDK设计

从 Vivado 软件中运行 SDK 后会出现下面的界面（读者会发现这款 SDK 是用 Eclipse 开发的）。上面两个框是我们的硬件信息，下面的框框内的 add\_0 就是加法器IP核导出的硬件信息了。



## 1、创建工程

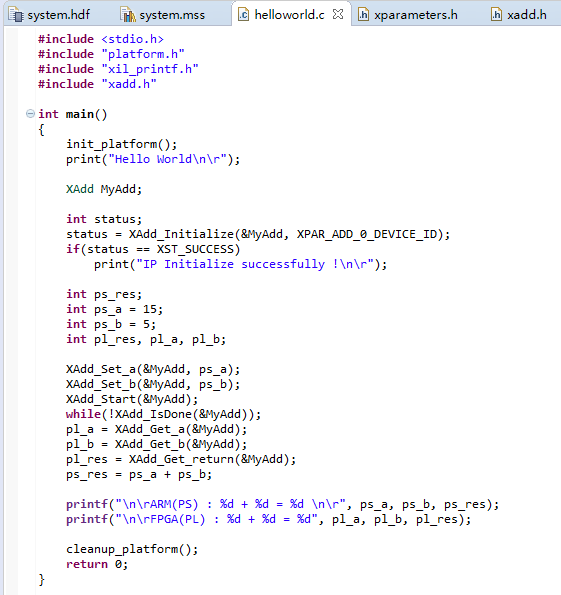
在菜单栏点击 File —— New —— Application Project ，弹出创建工程界面，输入项目名为 add （英文的无空格就可以），如下图所示，然后点击 Next。然后选择工程模板，选择 Hello World 工程模板。



## 2、程序设计

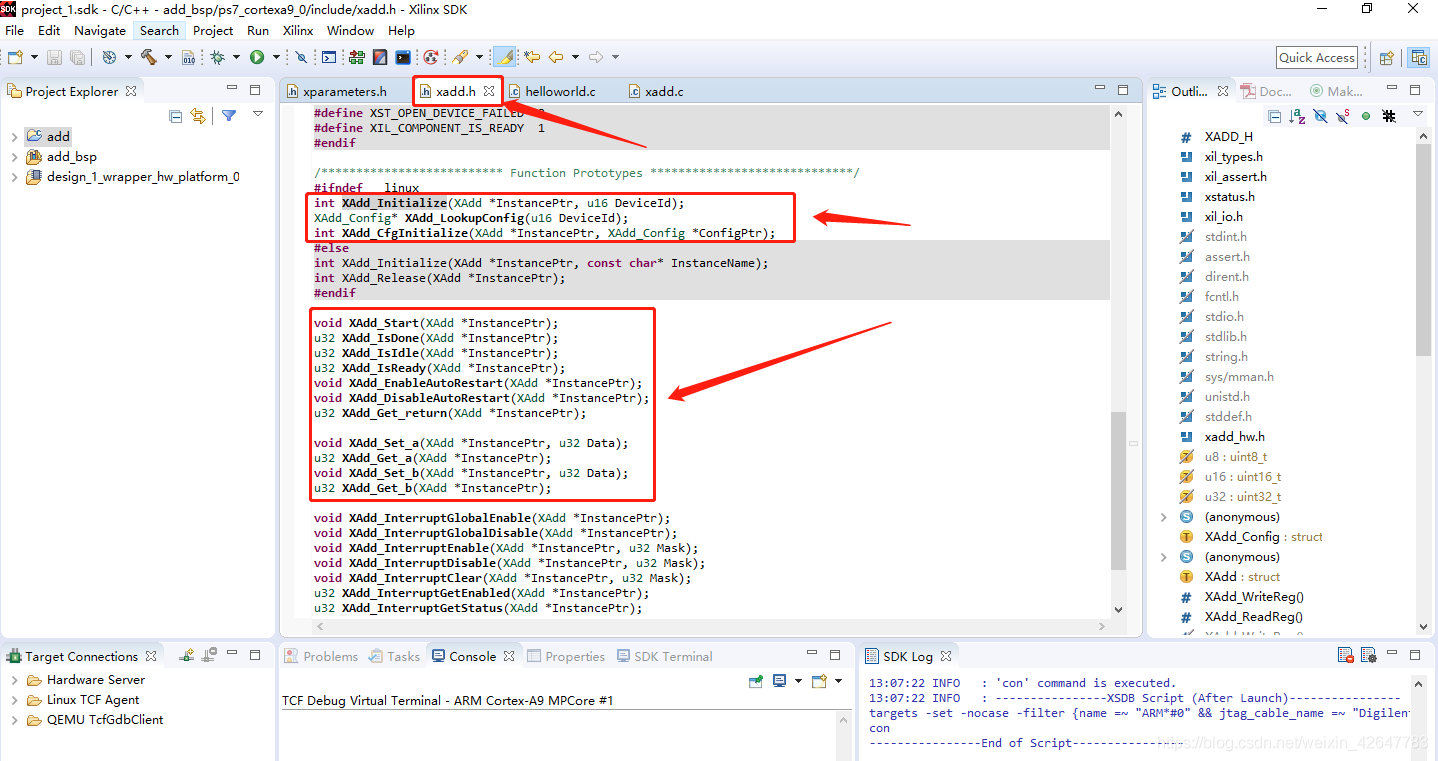
点开导航栏左侧的 Project Explorer 会发现里面有很多的文件，其中最最重要的就三个：第一个是 add/src/Helloworld.c ，这个c文件里面有ZYNQ的CPU程序入口 main函数 ；**第二个是 add\_bsp/ps7\_cortexa9\_0/include/xparameters.h ，这个头文件里面有我们所用到的加法器IP核映射到FPGA的DDR中的内存地址和相关驱动参数（这些是自动生成的）；**第三个是add\_bsp/ps7\_cortexa9\_0/include/xadd.h，其命名方式是与读者设计的IP核命名 相关，在其前面加上一个 ‘x’ 而已，比如我们的加法器是add，因此它的命名方式就是 xadd.h，这里面存放着我们调用自定义加法器IP核的库函数（或者叫做驱动）。有了以上三个文件，就可以开始搞事情了。

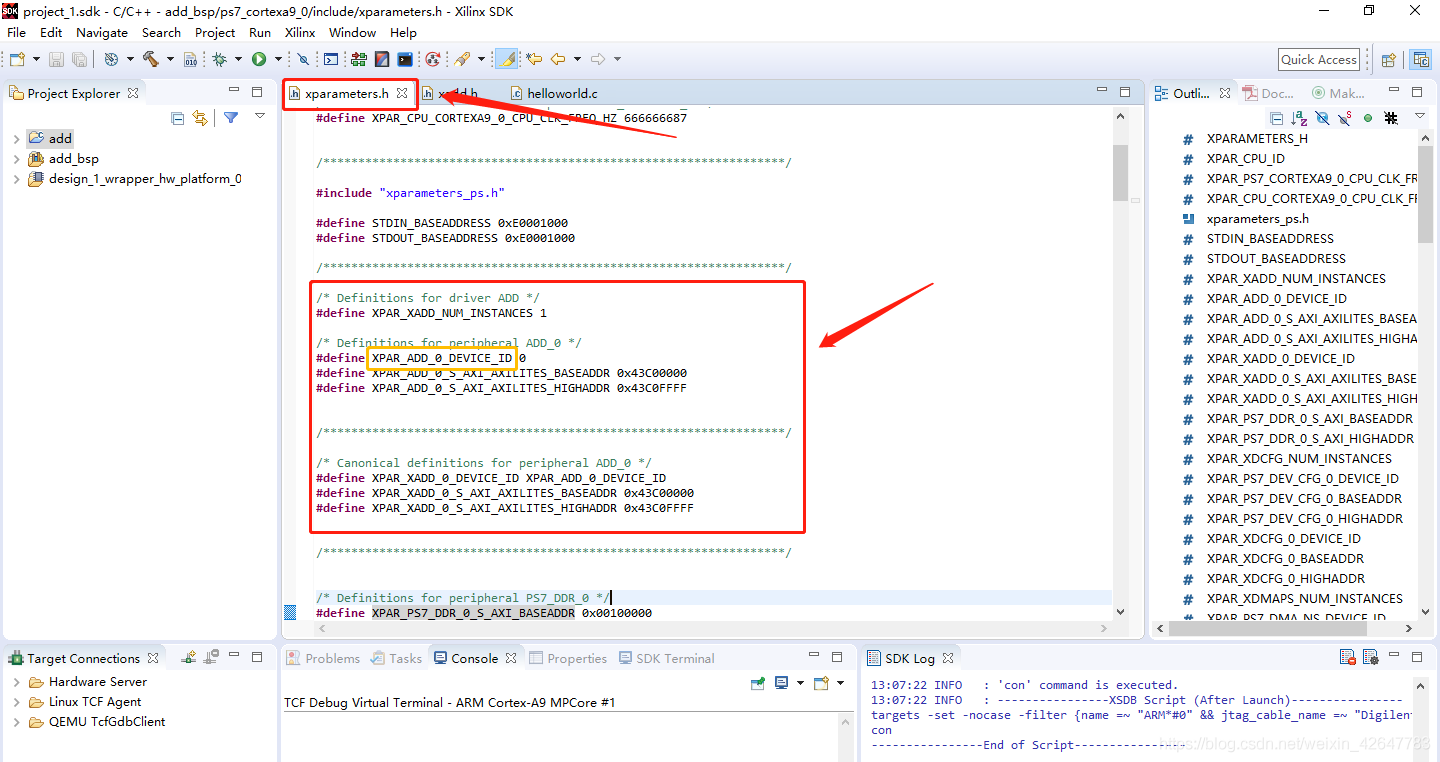
第一步，点开 helloworld.c ，输入以下代码。



首先需要先定义XAdd类型的结构体MyAdd，再调用 XAdd\_Initialize() 初始化IP核，如果返回值为XST\_SUCCESS即表示初始化成功。然后定义PS端（ZYNQ，ARM）和PL（FPGA）端的变量，之后利用库函数XAdd\_Set\_a()和XAdd\_Set\_b()把变量送到PL端，接着调用 XAdd\_Start()启动PL端的IP核开始计算，然后利用while(!XAdd\_IsDone(&MyAdd)); 阻塞方式等待IP核运算完毕，最后用 XAdd\_Get\_return() 取得 IP核的返回值。最终用ARM和FPGA做计算，对比结果，验证是否有误。

上述的XAdd结构体类型，以及对IP核的调用的库函数（驱动）全部来自于头文件xadd.h中。其中XAdd\_Initialize()的第二个参数是IP核的ID，这个ID号到xparameters.h头文件中找。所有用到的东西都在下面的截图中。注意红色方框。之后有用到我会再补充介绍其他函数，本篇博文节省篇幅先把用到的都介绍了，其他的放到之后的博文介绍。

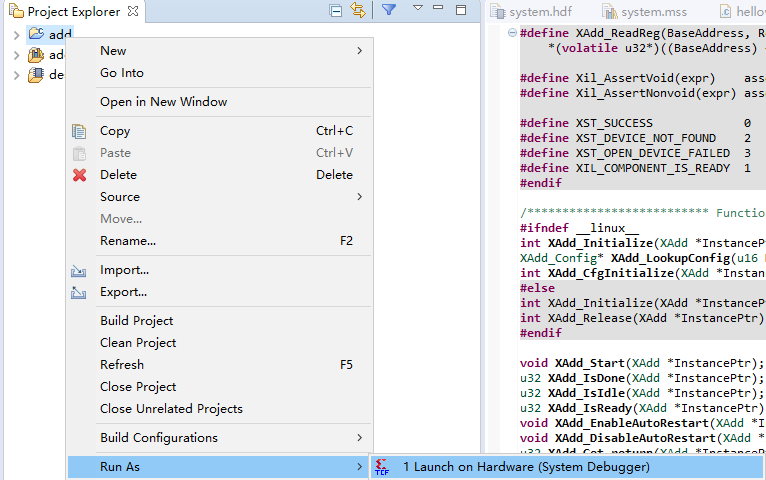




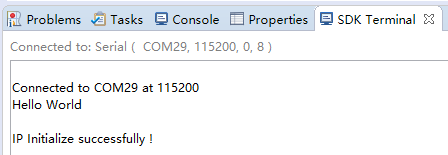
## 3、烧录程序

程序写好之后就可以烧录到板子上了。

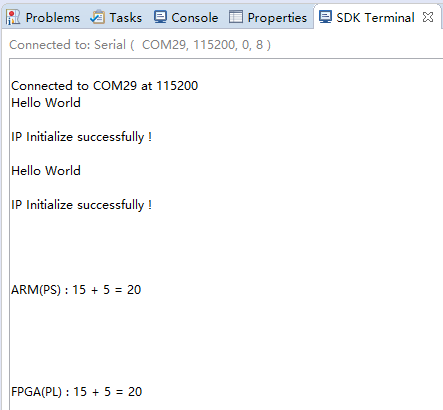
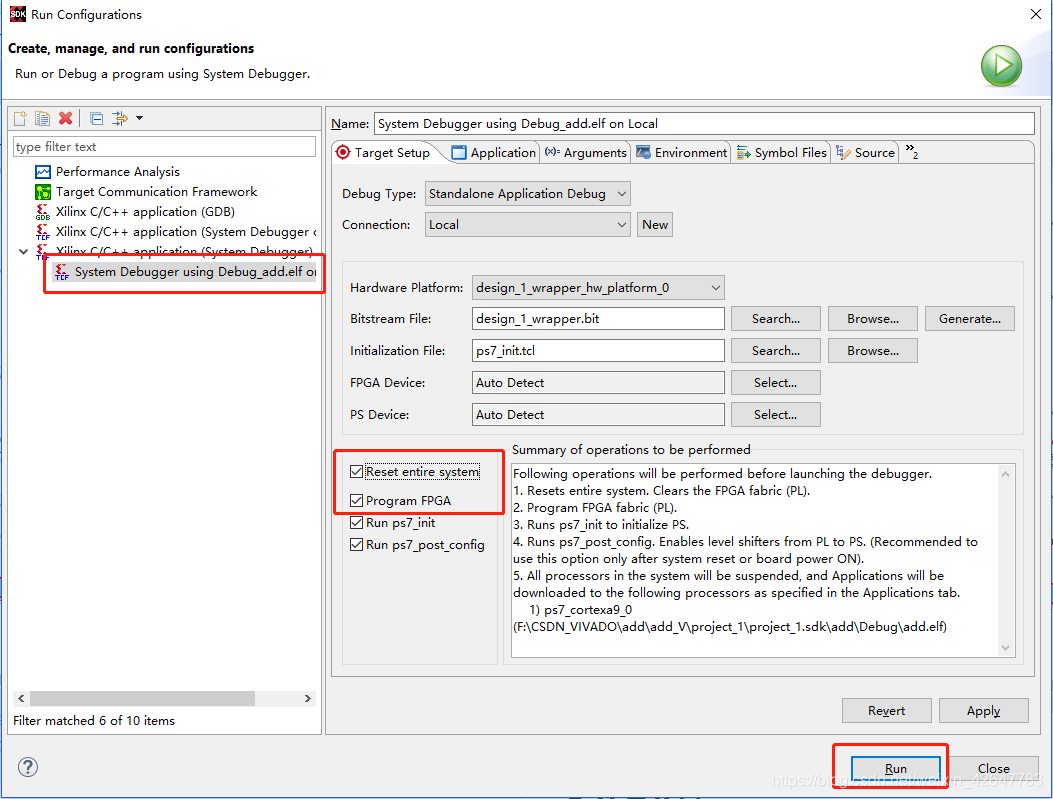
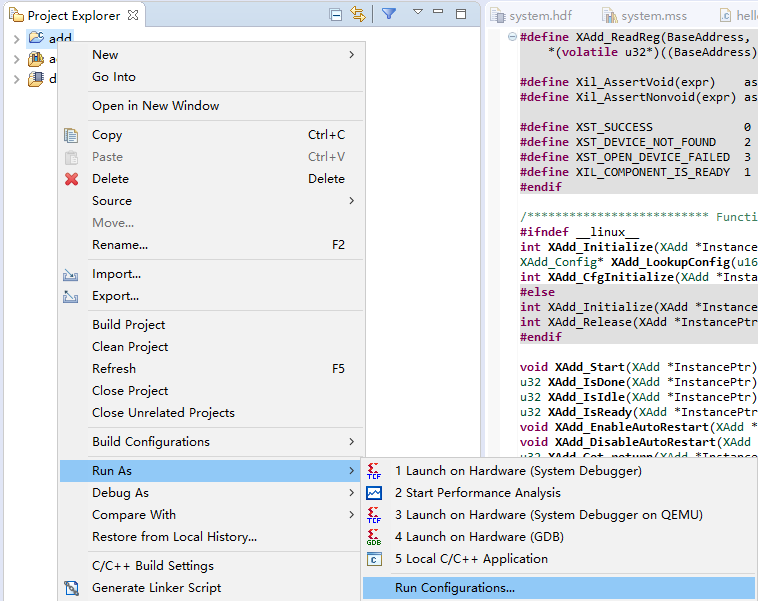
第一步先把程序烧录到 ZYNQ 上，如下图所示操作。



用串口连接PC和开发板，（这里假设读者都已经安装好串口驱动之类的，如果不知道怎么做的，提供开发板的产商应该会给相应的资料，照做即可），打开串口调试助手，看运行结果，如下图所示。可以看到这里程序只运行到初始化这一步，没有之后我们所预想的有打印数据出来，这主要是因为硬件信息未烧录到FPGA中，因此在运行while (!XAdd\_IsDone(&MyAdd)); 会阻塞住不正常运行，到下一步。



第二步，把程序同时烧录到 ZYNQ 和 FPGA 上，操作如下图所示。



在串口助手上看到这样的结果，跟我们的预期一样，大功告成。

**【补充】：为什么要先烧录到ARM再烧录到FPGA，因为一个新的工程如果没有进行烧录到ARM的操作的话，之后烧录到FPGA的操作界面会缺少一个生成的System Debugger。**

# 五、总结

这篇博文笔者详细的介绍了从HLS上IP核的制作，到Vivado上顶层电路的连接，最后到SDK上应用程序的开发，完成了用ARM核控制FPGA完成特定功能的任务。

我们把流程重新理一遍。

**HLS制作IP核**

C仿真（C Simulation）

C综合（C Synthesis）

C/RTL 联合仿真（C/RTL Cosimulation）

导出RTL（Export RTL）

**Vivado构建完整电路**

把自定义IP核添加到IP Catalog（Add Repository）

创建Block Design并添加自定义IP核以及ZYNQ处理系统，然后配置ZYNQ（Create Block Design）

在Block Design中自动连接ZYNQ和自定义IP核并导出必要端口（Open Block Design）

封装HDL并生成硬件输出（Create HDL Wrapper & Generate Output Products）

运行项目（Run Implementation）

生成bitstream并导出硬件（Generate Bitstream & Export Hardware）

**SDK开发应用程序**

包含自定义IP核对应的驱动头文件（例如 xadd.h）

声明自定义IP核对应的结构体并初始化（例如 XAdd MyAdd）

利用库函数把数据输入到IP核，然后等待IP核运算完毕，再取输出数据

把程序烧录到ARM上

把程序同时烧录到ARM和FPGA上

利用串口调试助手检验设计

以上便是全部操作流程，之后在自行设计的时候都要走上面这一串流程，唯一可以跳过的只有C/RTL Cosimulation，因为对一些比较大型的IP核，这个过程特别慢而且非必要，除此之外每一个步骤都不能缺少。

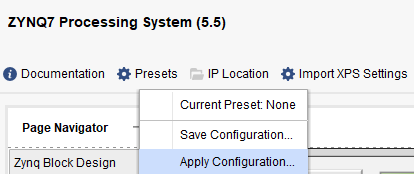
虽然笔者只用了最简单的加法器作为例子给读者讲解这一套开发技巧，但是加法是所有运算的基础，只要加法调通了，之后再做其他的操作都简单很多了，这就像程序员在学习每一门编程语言的“Hello World”一样。希望没有相关开发经验的读者能完整地把这一流程走一遍，之后自己便能举一反三，慢慢摸出自己的一条硬件开发之路。

# 六、IP配置小问题

**HLS创建IP核时要选择XC7Z020CLG484-1**

**创建Vivado工程要选择支持包Zynq-7000 Embedded Development Plantfrom（Z7-EDP-1）**

**自己创建的IP先导入配置文件z7edp.tcl再修改**

****

**重要资料都在E:\data\9\_Vivado\_XilinxFPGA\_Zynq7000\00\_important**