

智能芯片实验报告

**实 验 名 称 LWIP以太网数据传输实验**

**学 院 电子与信息工程学院**

**学 号 2150248**

**姓 名 姚天亮**

**任 课 教 师 徐和根**

**实验十二 LWIP以太网数据传输实验**

**一、实验目的**

PS通过AXI-GPIO核通知PL用户自定义ip核循环构造32bit位宽的0-1001的数据，写入FIFO，利用AXI-DMA核将FIFO中的数据读入到DDR中，PL每发完一次 0-1001，AXI-DMA核便会产生一个中断信号到PS，PS 得到中断信号后将 DDR3缓存的数据以乒乓操作的方式通过TCP协议发送至PC机。

**二、实验原理**

以太网MAC是一个基础模块，它使得我们可以实现一个TCP/IP协议栈，协议使得应用可以基于网络来通信。如果想要解析一个TCP/IP协议栈，需要了解以下层次：

• 第一层——物理层：（描述物理连接）主要由外部物理层芯片实现

• 第二层——数据链路层：（描述物理层上数据传输的途径）可以由PS的MAC实现

• 第三层——网络层：（描述不同网络间数据传输的途径）提供网络路由和寻址层，可以使IPv4或者IPv6

• 第四层——传输层：（提供终端用户间的数据传输）TCP

• 第五层——应用层：应用层操作数据



SDK开发环境包含一个轻量级的IP栈(lwIP)，lwIP是个开源IP栈，在Zynq上的PS中有两个千兆以太网控制器，每个控制器都是可配置的，并且可以单独控制，千兆以太网控制器可以实现一个MAC（介质访问控制层）（数据链路层的组成之一），物理层主要是由外部的PHY芯片、电缆或光纤完成

应用和lwIP栈接口的主要方法：（详情参考XAPP1026）

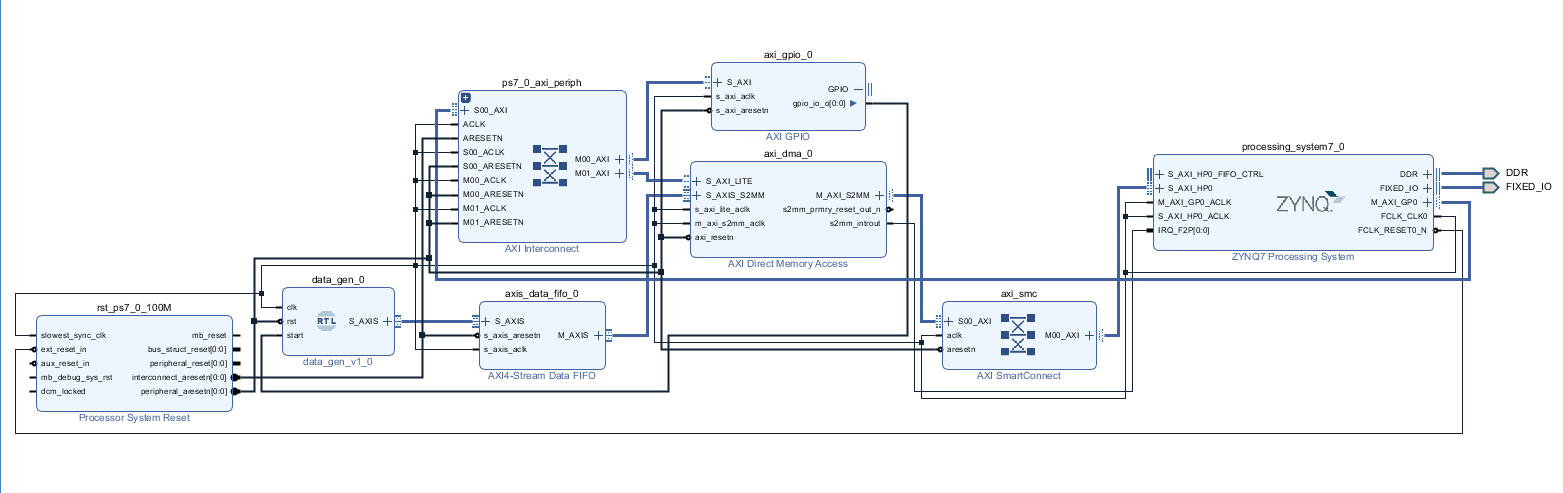
　• Raw——没有操作系统时使用的基本API，基于事件驱动，使用lwIP的应用在该模式下，在触发如接收数据等事件时调用相应的回调函数

　• Sockets——使用在运行操作系统的Zynq上，该方法使用到线程，应用使用它来和lwIP栈通信

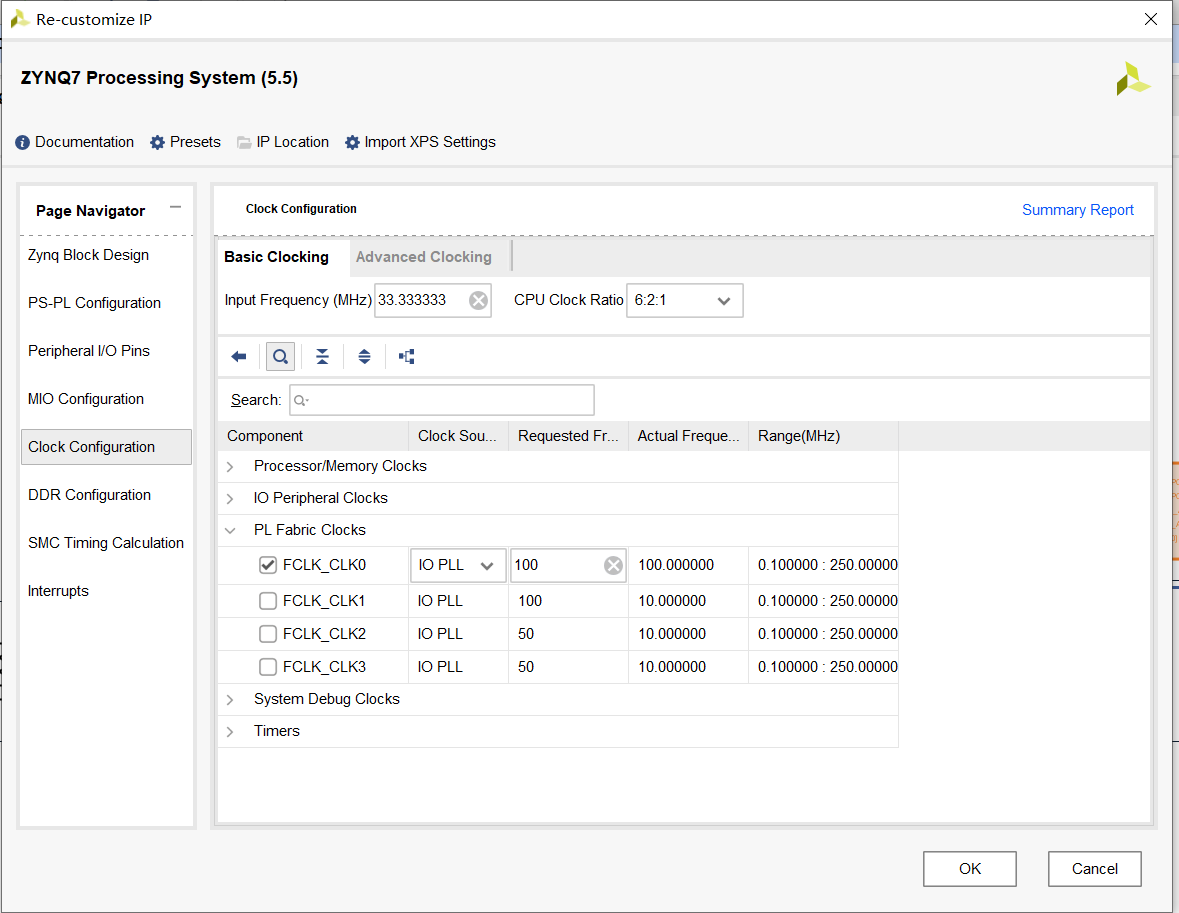
**三、实验步骤**

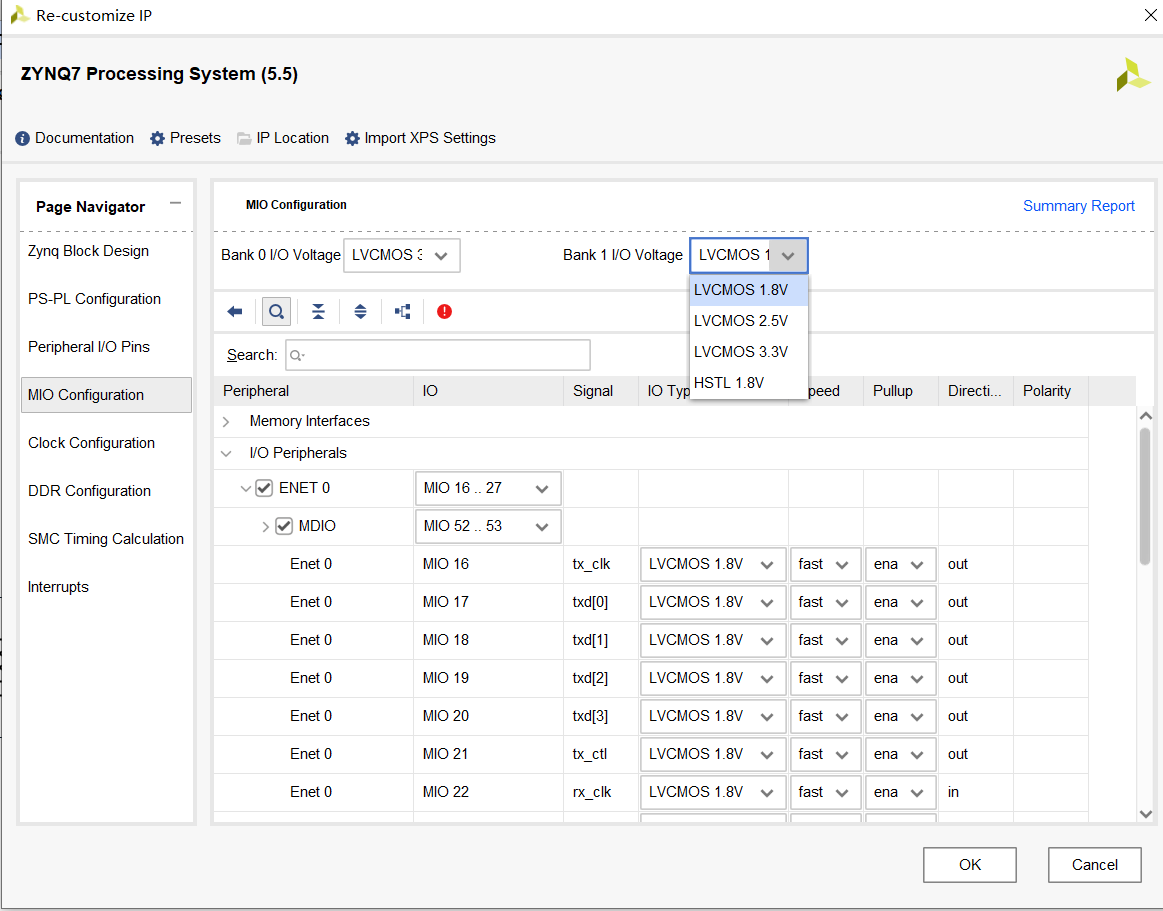
1. 硬件平台搭建

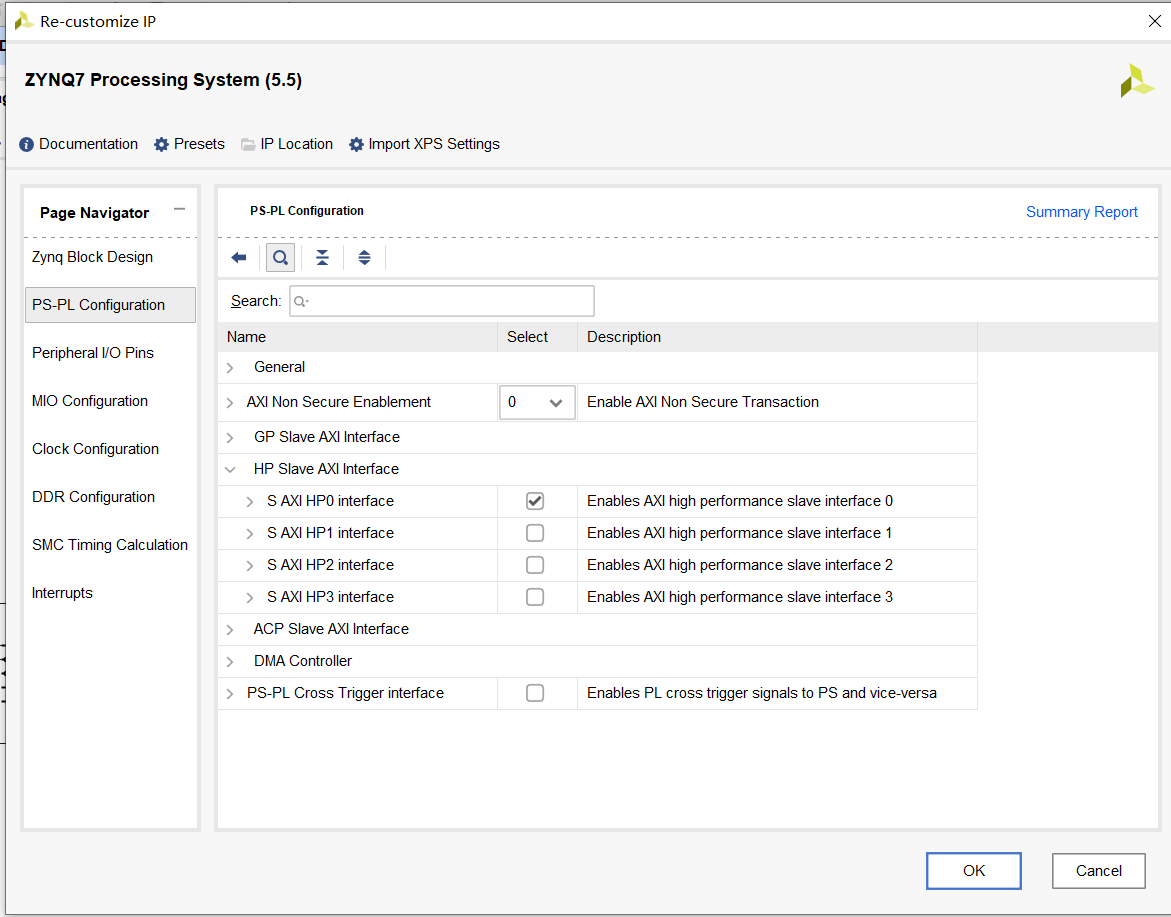
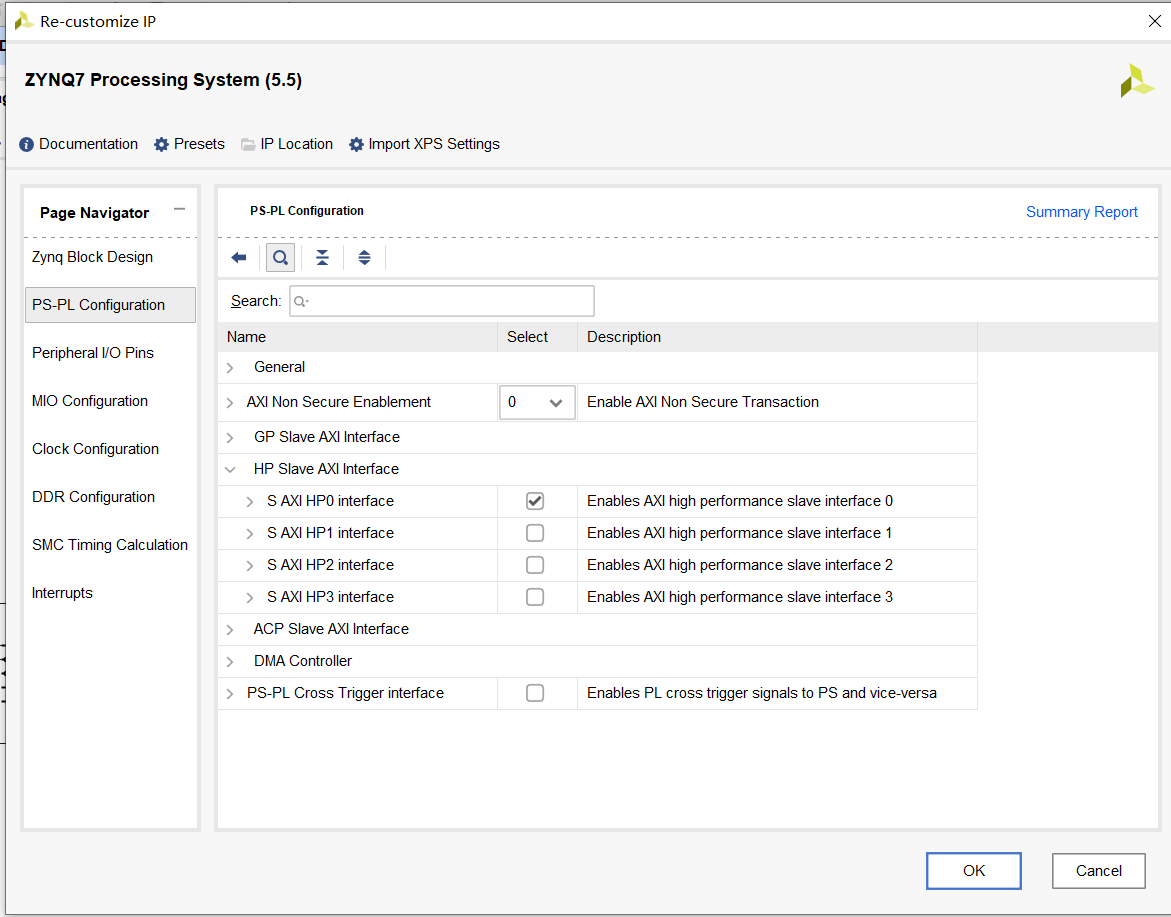
本实验硬件平台是在DMA环路实验基础上搭建。系统总体原理图如下：

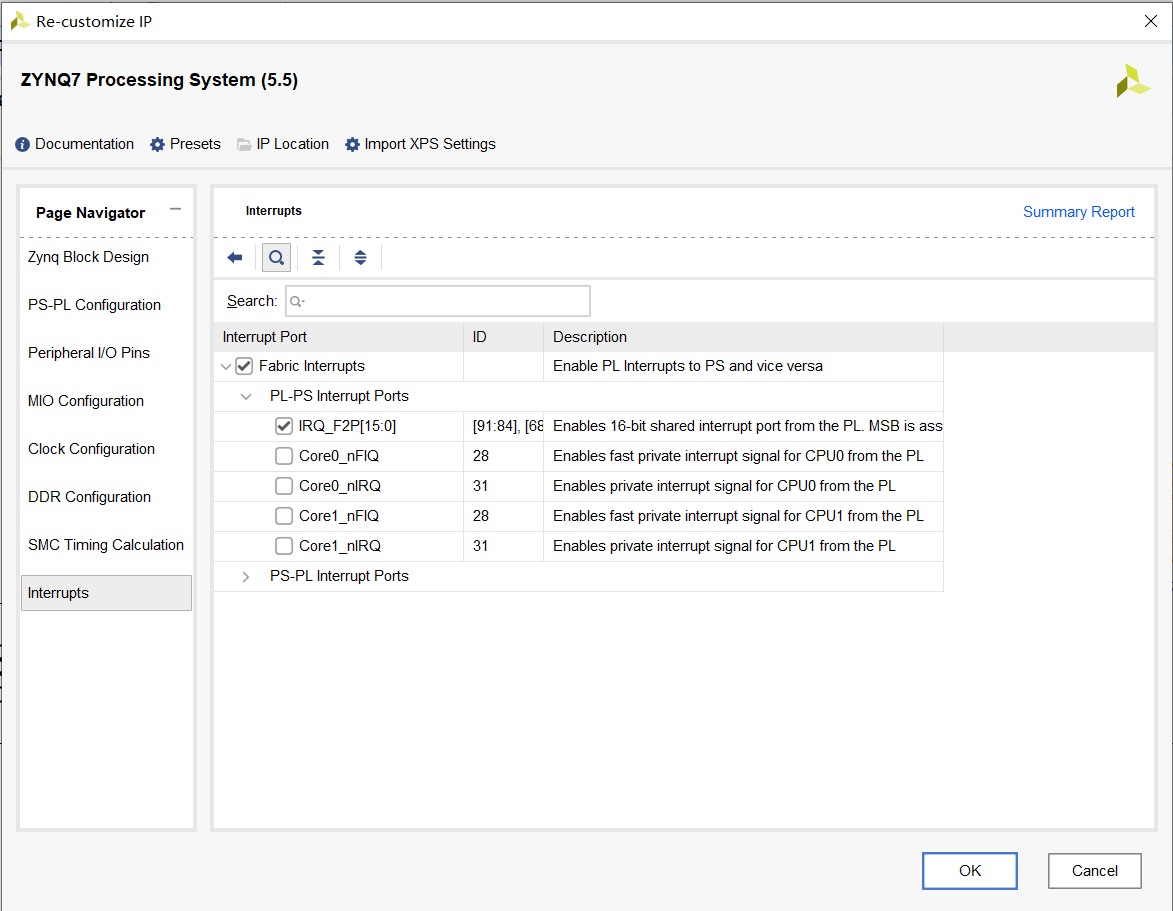


1. PS参数设定，PS-PL的频率100Mhz，以太网口设定，添加一个HP接口，添加PL-PS的中断接口，使能uart1口

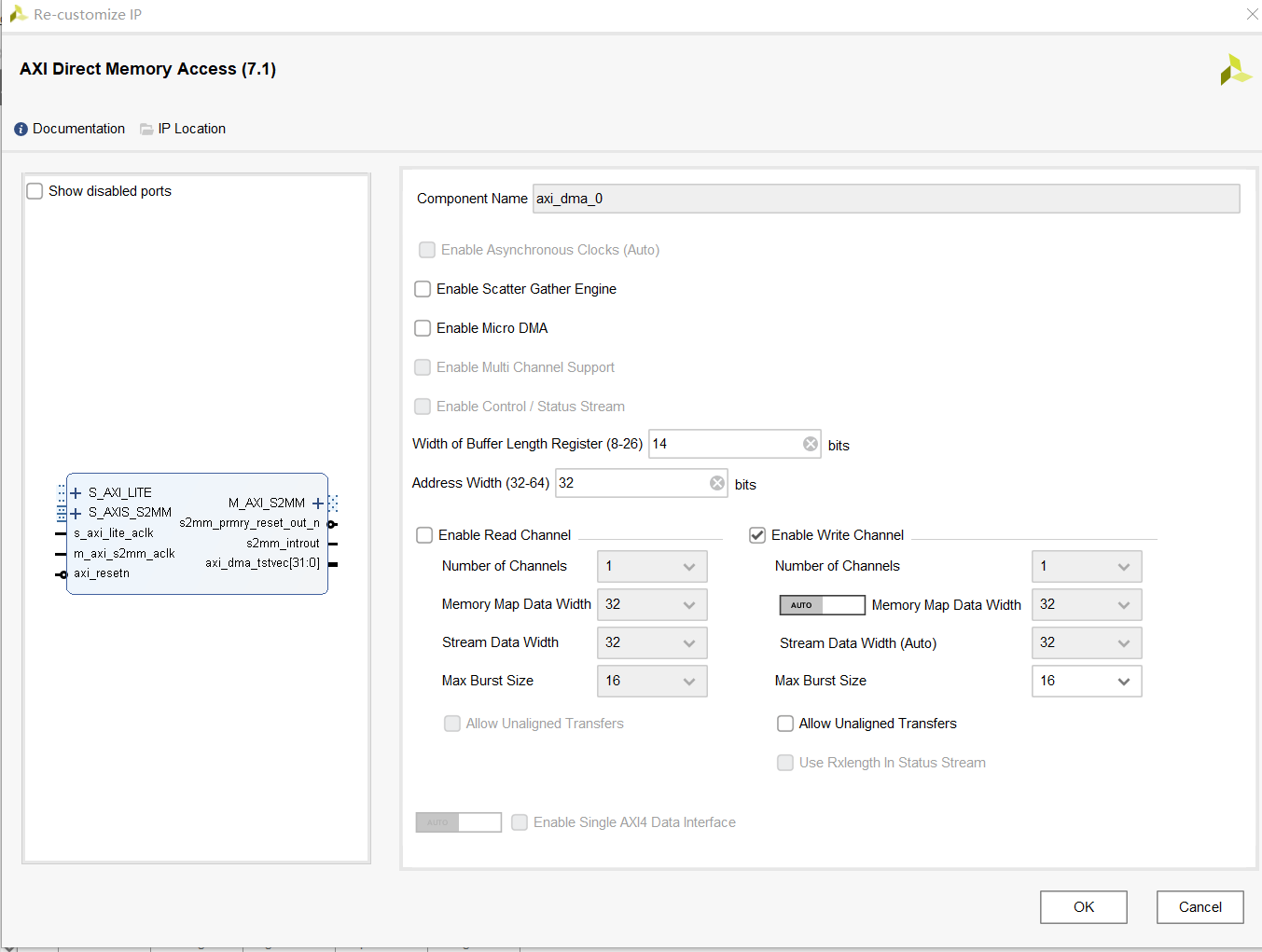




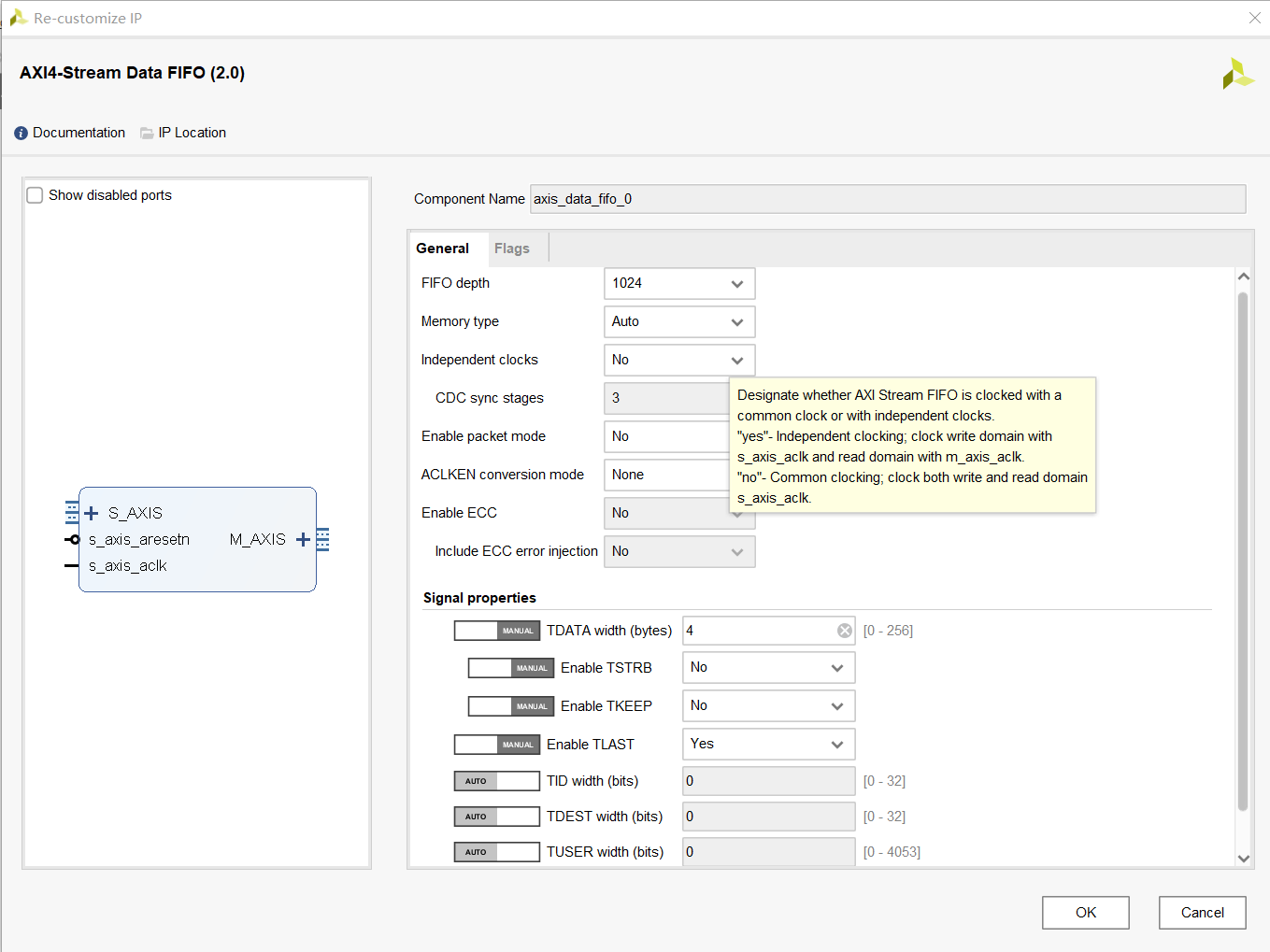




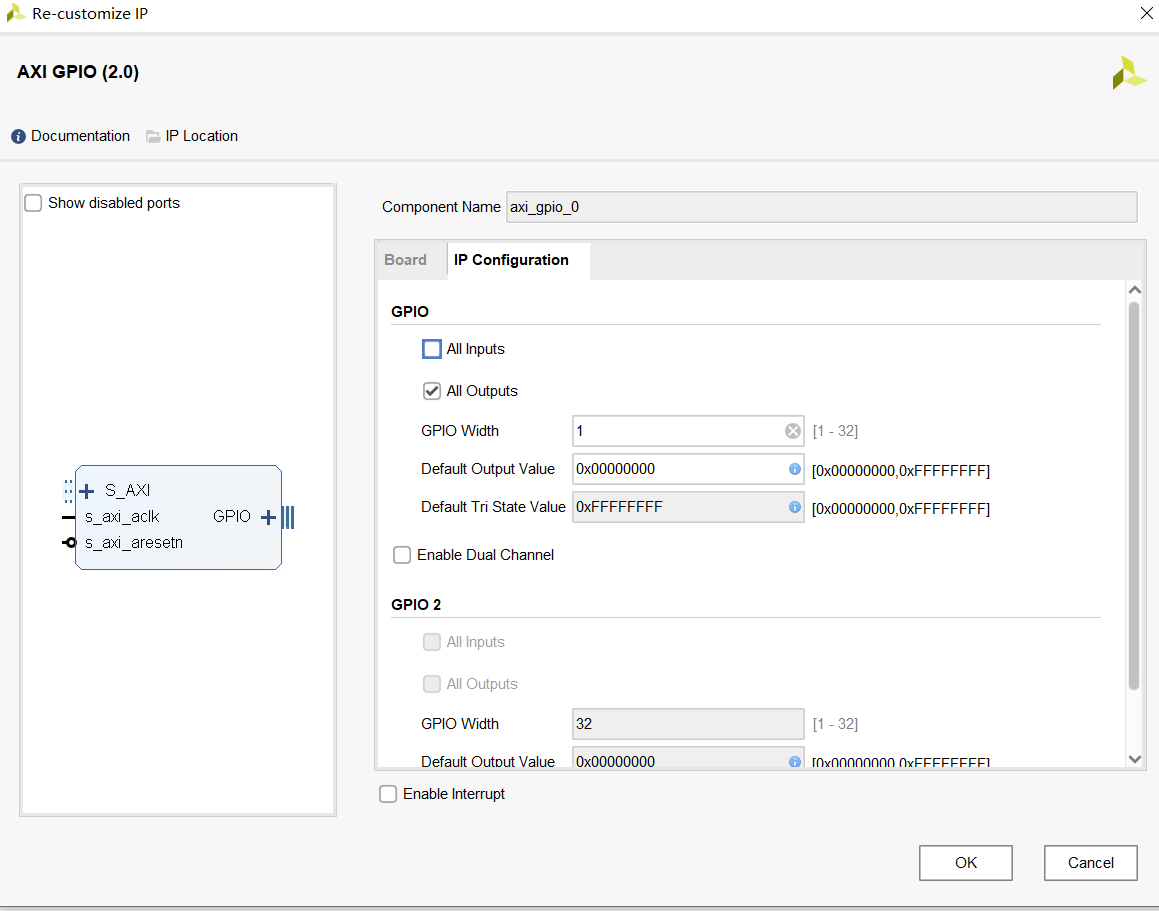
1. 添加AXI-DMA ip core，设置参数，本实验只是让fpga产生数据通过DMA写入DDR，故只需要打开写通道。此处的读/写是以Memory-Map，即DDR为对象的。



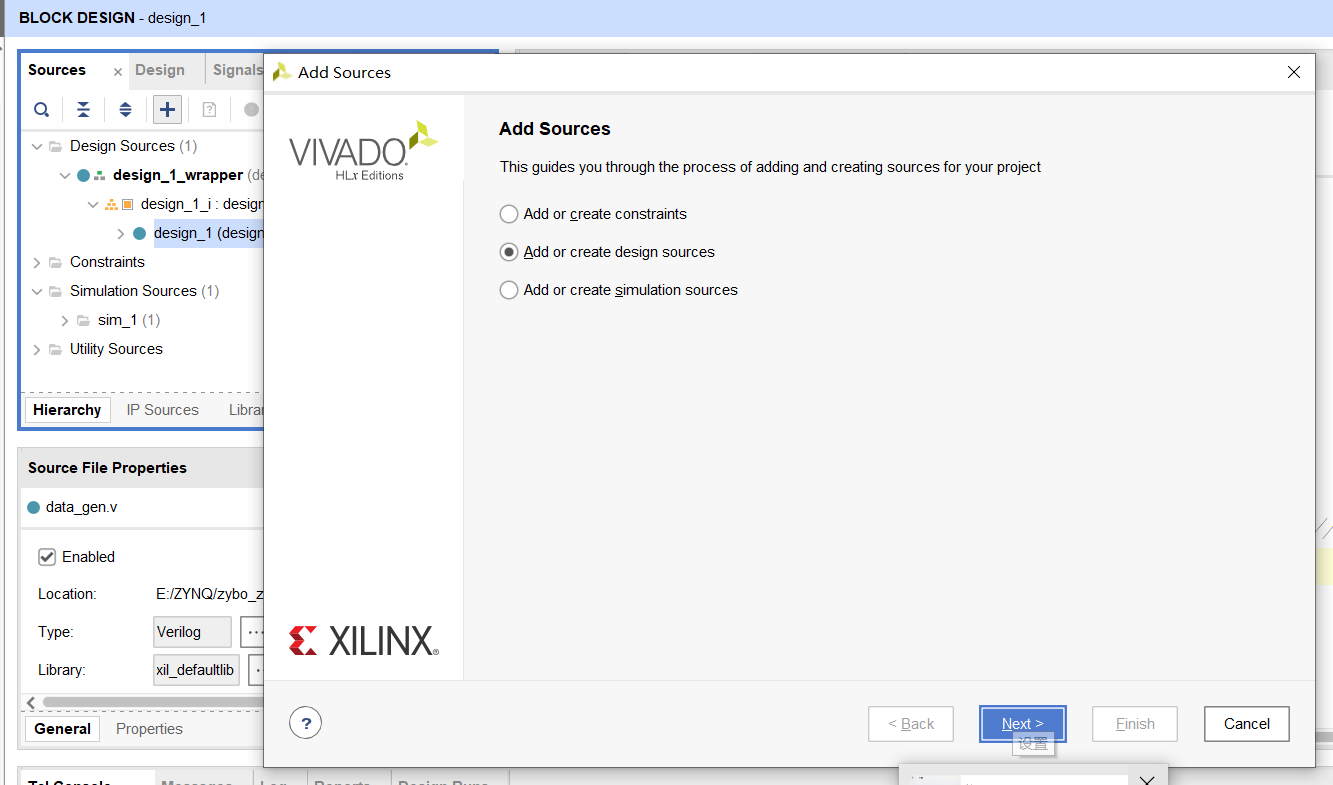
1. 添加Stream Data FIFO 核，设置参数



1. 添加一个AXI-GPIO核，custom配置，1bit输出，用于由ps控制启动用户自定义的数据生成模块。



1. ADD source，添加用户自定的数据生成模块，代码如下：



module data\_gen(

input clk,

input rst,

input start,

input S\_AXIS\_tready,

output S\_AXIS\_tvalid,

output S\_AXIS\_tlast,

output [31:0] S\_AXIS\_tdata

);

reg S\_AXIS\_tvalid;

reg S\_AXIS\_tlast;

reg [31:0] S\_AXIS\_tdata;

reg [1:0] state;

always@(posedge clk) begin

if(!rst) begin

S\_AXIS\_tvalid <= 1'b0;

S\_AXIS\_tlast <= 1'b0;

S\_AXIS\_tdata <= 32'd0;

state <= 0;

end

else begin

case(state) //状态机

0: begin

if(start&&S\_AXIS\_tready) begin //启动信号到来且FIFO可写

S\_AXIS\_tvalid <= 1'b1; //设置写FIFO有效

state <= 1;

end

else begin

S\_AXIS\_tvalid <= 1'b0;

state <= 0;

end

end

1:begin

if(S\_AXIS\_tready) begin //来自fifo模块，表示FIFO准备就绪可写

S\_AXIS\_tdata <= S\_AXIS\_tdata + 32'b1;

if(S\_AXIS\_tdata == 32'd1000) begin //判断是否结束

S\_AXIS\_tlast <= 1'b1;//发送最后一个数据

state <= 2;

end

else begin//等待数据发完

S\_AXIS\_tlast <= 1'b0;

state <= 1;

end

end

else begin//等待FIFO可写

S\_AXIS\_tdata <= S\_AXIS\_tdata;

state <= 1;

end

end

2:begin

if(!S\_AXIS\_tready) begin //FIFO满则等待

S\_AXIS\_tvalid <= 1'b1;

S\_AXIS\_tlast <= 1'b1;

S\_AXIS\_tdata <= S\_AXIS\_tdata;

state <= 2;

end

else begin //写入结束

S\_AXIS\_tvalid <= 1'b0;

S\_AXIS\_tlast <= 1'b0;

S\_AXIS\_tdata <= 16'd0;

state <= 0;

end

end

default: state <=0;

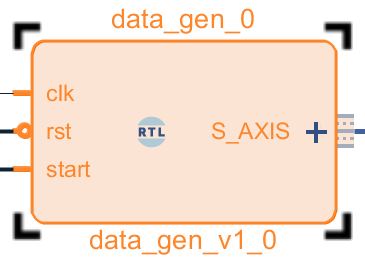
endcase

end

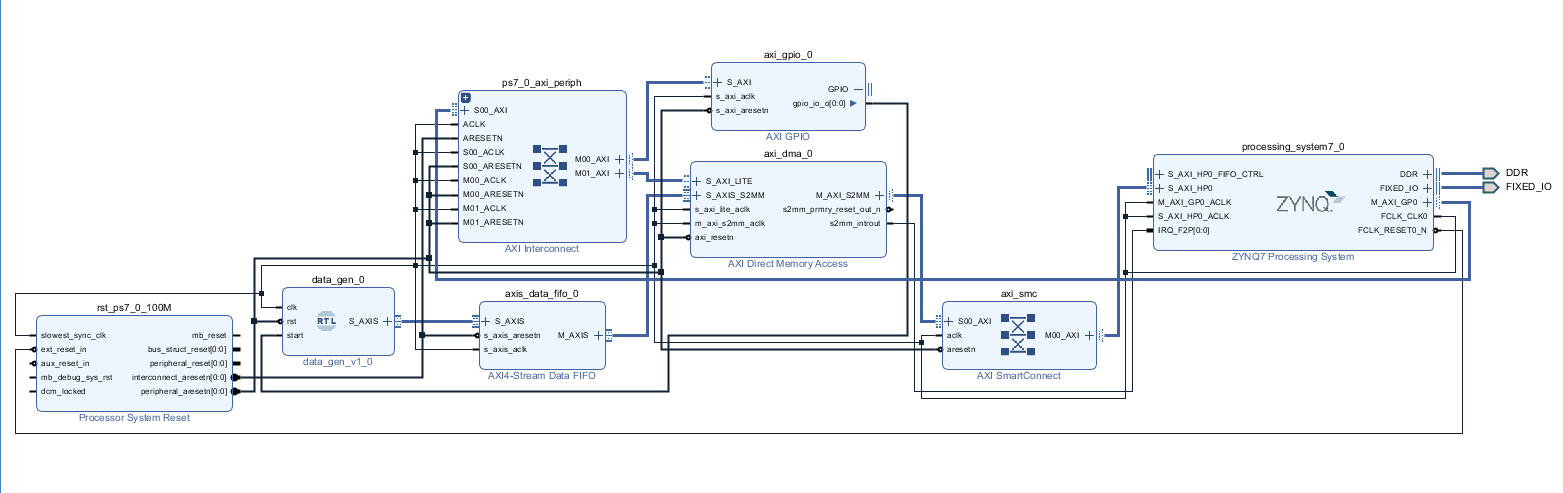
end

endmodule

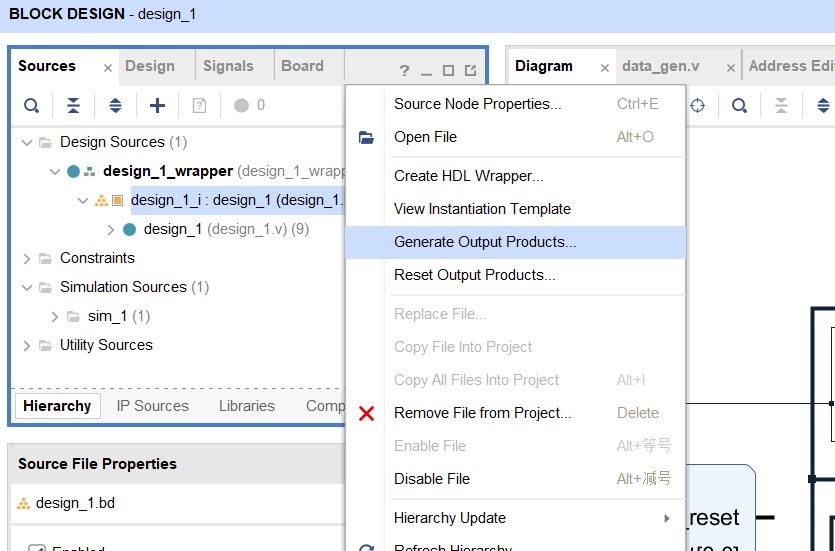
1. 在source栏选中data\_gen.v文件，将它拖入block design窗口中，将自动生成一个RTL模块。



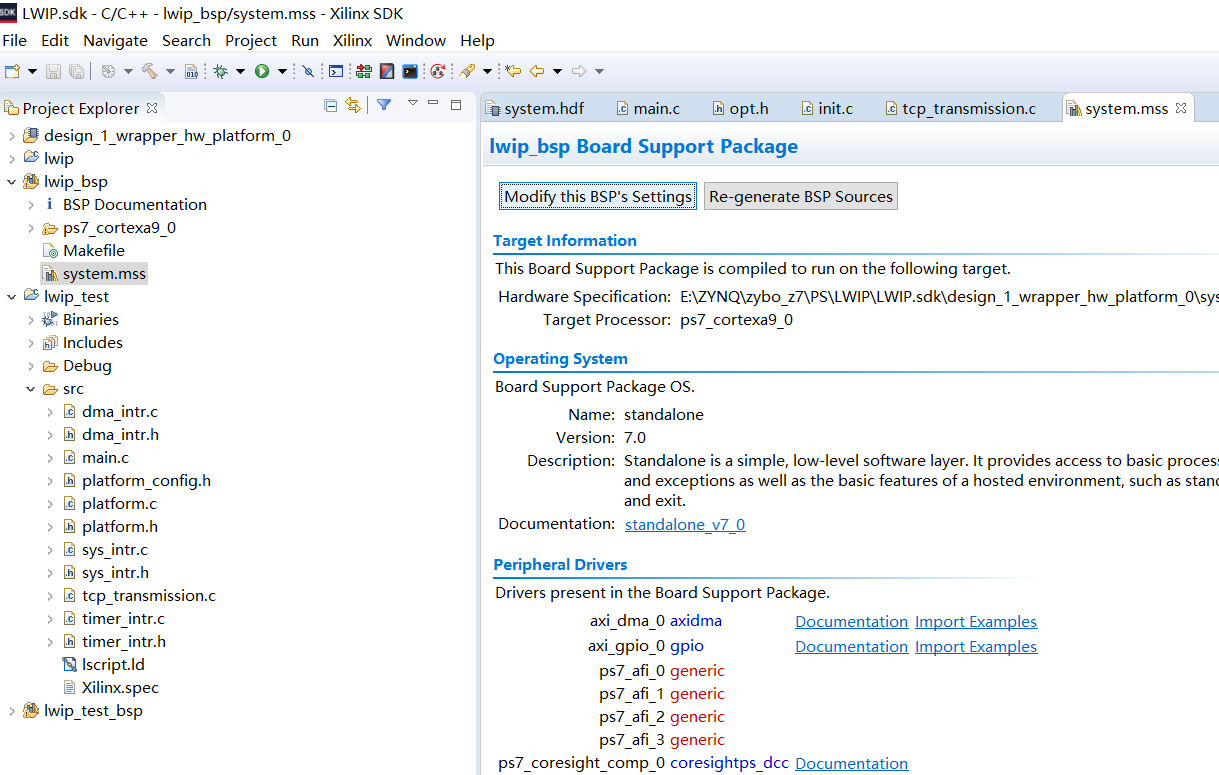
1. 自动连线，必要时手动调整，最后的总体结构图如下：

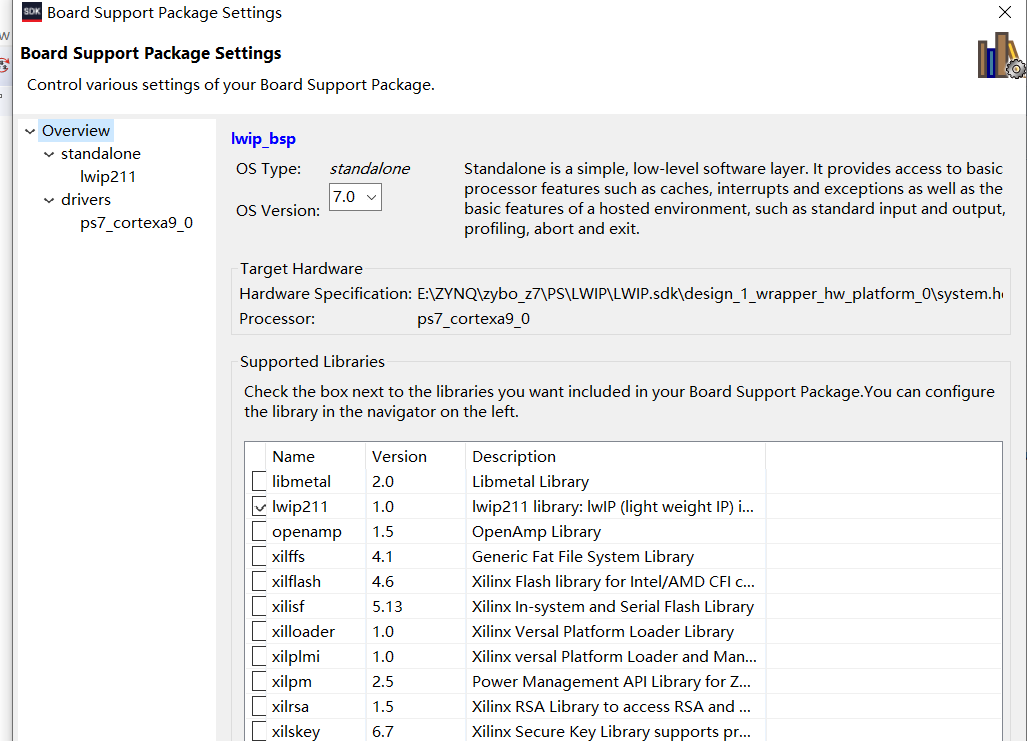


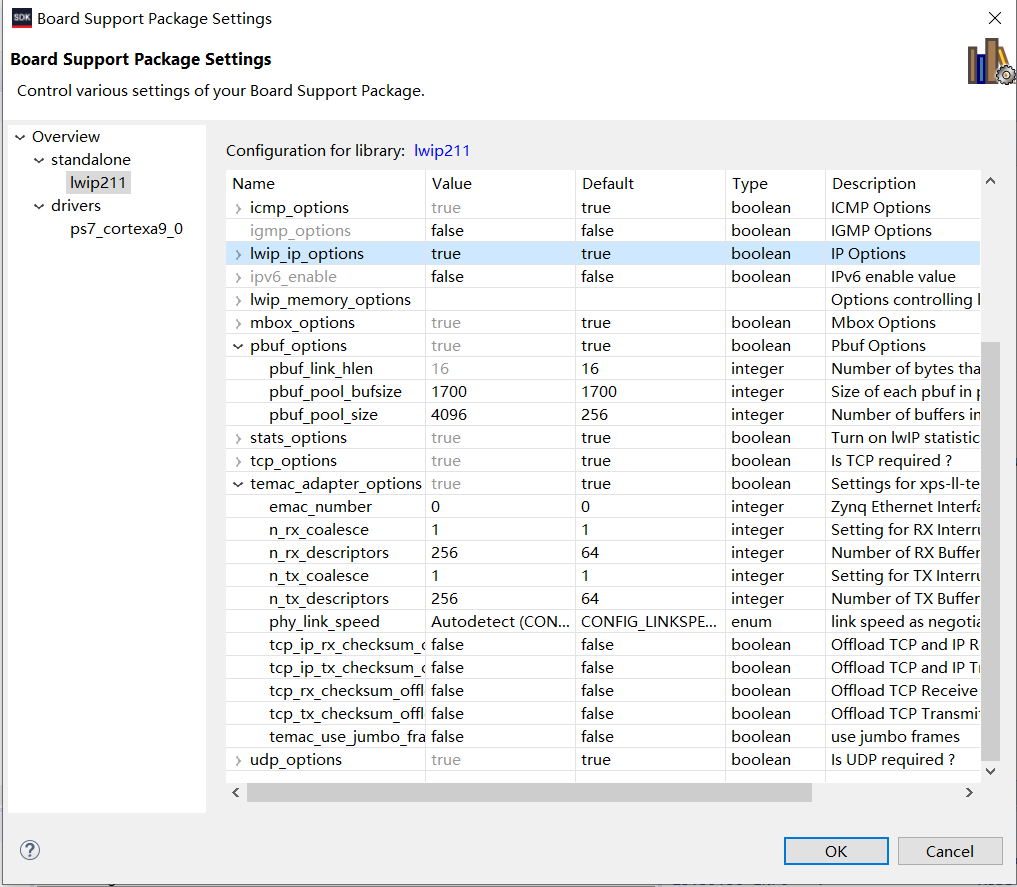
1. Validate Design，没有错误后，generate output product，然后create HDL Wrapper



1. 综合、实施、生成bitstream
2. Export hardware、launch SDK
3. SDK软件设计
4. 新建 application project，名称为LWIP\_TEST，模板选默认的helloworld。
5. 打开system.mss文件，modifiy this bsp’s settings，添加lwip支持包，设置lwip的相关参数，将默认值改大的目的是为了加快tcp传输速度，提高传输效率。

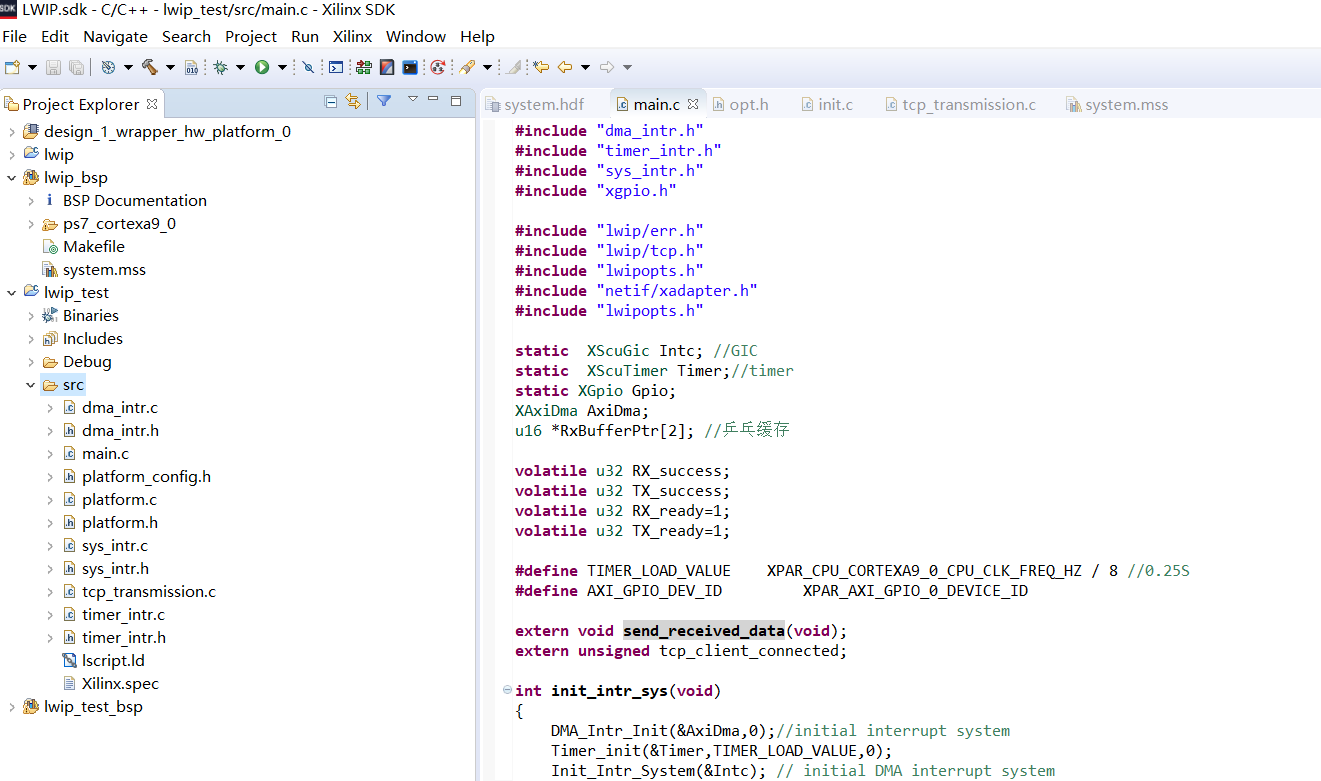






1. 删除helloword.c 文件，将参考代码中的源文件粘贴到src目录





1. 代码中：

main.c：对各设备和IP协议栈初始化，与服务器建立连接，通知启动PL产生数据并写到FIFO，在while循环中将PL产生的数据通过TCP发送给PC机

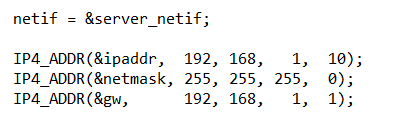
tcp\_transmission.c：包含tcp的初始化函数、连接服务器成功的回调函数、和将DMA数据通过TCP发送给PC的函数，需要注意的是SEND\_SIZE和PAKET\_LENGTH是按照字节数算的，要不小于FIFO的一帧数据的字节数

dma\_intr.c：DMA初始化和接收中断的处理函数

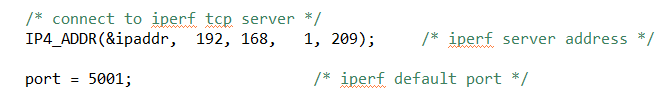
sys\_intr.c：系统中断(GIC和系统异常)初始化函数

timer\_intr.c：定时器初始化

main函数中，将开发板的ip地址、掩码及默认网关设定为：



在tcp\_transmission.c 中，将PC机的ip地址和端口号设定为：



1. AXI DMA 数据传输过程

例程中 axi dma 采用了 simple transfer 方式，通过 XAxiDma\_SimpleTransfer 函数完成。每次 dma 传输都需要 PS 主动发起，PS 通过 AXI 总线配置 PL 侧 axi dma 内部寄存器，发起一次 dma 传输。dma 传输发起后，axi dma 开始 通过 S\_AXIS\_S2MM 接口接收数据，当其中的 tlast 信号被拉高，则代表当次传输所需要的数据发送完毕，当该次 dma 传输结束，axi dma 通过 s2mm\_introut 产生中断信号，触发 PS 中断控制器产生中断，PS 通过中断服务函数 Dma\_RxIsr 清除 axi dma 的中断状态，在 DMA 中断函数中，拉高 dma 完成指示信号 packet\_trans\_done，一次完整的simple transfer 的 dma 传输结束。下面是 dma 中断接收函数，接收来自 PL 的中断信号，并且设置 packet\_trans\_done。

DMA 中断接收函数为：static void DMA\_RxIntrHandler(void \*Callback)

PS 的 dma 数据接收采用了乒乓操作的模式，两个缓冲区交替进行数据接收。 需要注意的是， XAxiDma\_SimpleTransfer 函数中 Length，以字节为单位，每次发起 dma 时，所设置的 Length 的值必须大于或等于 PL 实际传输的数据长度，否则会出现错误。本例程中设置的长度为 2048 字节。 first\_trans\_start 是为了进行第一次先进行一次 DMA 中断传输，这样完成后设置 first\_trans\_start 为 0。以后每次 完成网络传输后，再启动 DMA 接受。 TCP 数据包的发送主要依赖于 tcp\_write 和 tcp\_output 两个函数，tcp\_write 将所需要发送的数据写入 tcp 发送缓 冲区等待发送，tcp\_output 函数则将缓存区内数据包发送出去。在发送 TCP 数据包时，这两个函数往往要同时配合使用。

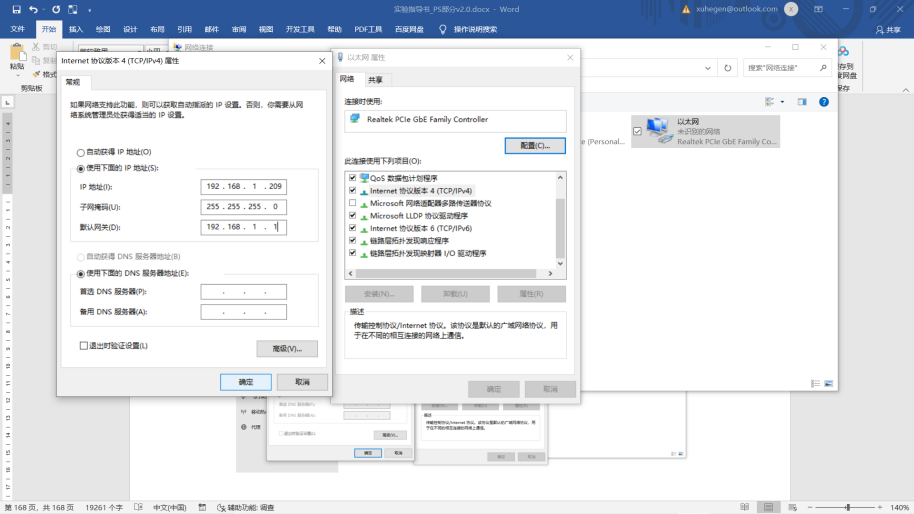
TCP的收发函数为：void send\_received\_data()

1. TCP 发送流程

在本例程中，zynq 作为客户端，PC 作为服务器。由 zynq 向 PC 主动发起 TCP 连接请求，通过 tcp\_connect 函数 便可以完成这个过程。该函数的参数包含了一个回调函数指针 tcp\_connected\_fn，该回调函数将在 TCP 连接请求三 次握手完成后被自动调用。该回调函数被调用时代表客户端和服务器之间的 TCP 连接建立完成。在本例程中，该回 调函数被定义为 tcp\_connected\_callback，在该函数中，拉高连接建立完成信号 tcp\_client\_connected，并通过 tcp\_sent函数配置另一个 TCP 发送完成的回调函数。该回调函数在每个 TCP 包发送完成后会被自动调用，代表 TCP 包发送 完成。该回调函数在本例程中被定义为 tcp\_sent\_callback，仅作发送完成数据包的计数。

1. 连接测试

把开发板网卡通过网线接到 PC 网口上，修改 IP 地址如下图，打开网络调试助手，设置电脑为 TCP Server 本机 IP 为刚才设置的 192.168.1.209 端口号为 5001。



工程编译下载运行后，网络调试助手不停接收数据。

