

MYIR-ZYNQ7000系列-zturn教程(23)：DMA回环测试

原创

虚无缥缈vs威武

最后发布于2019-03-20 18:17:40

阅读数 556

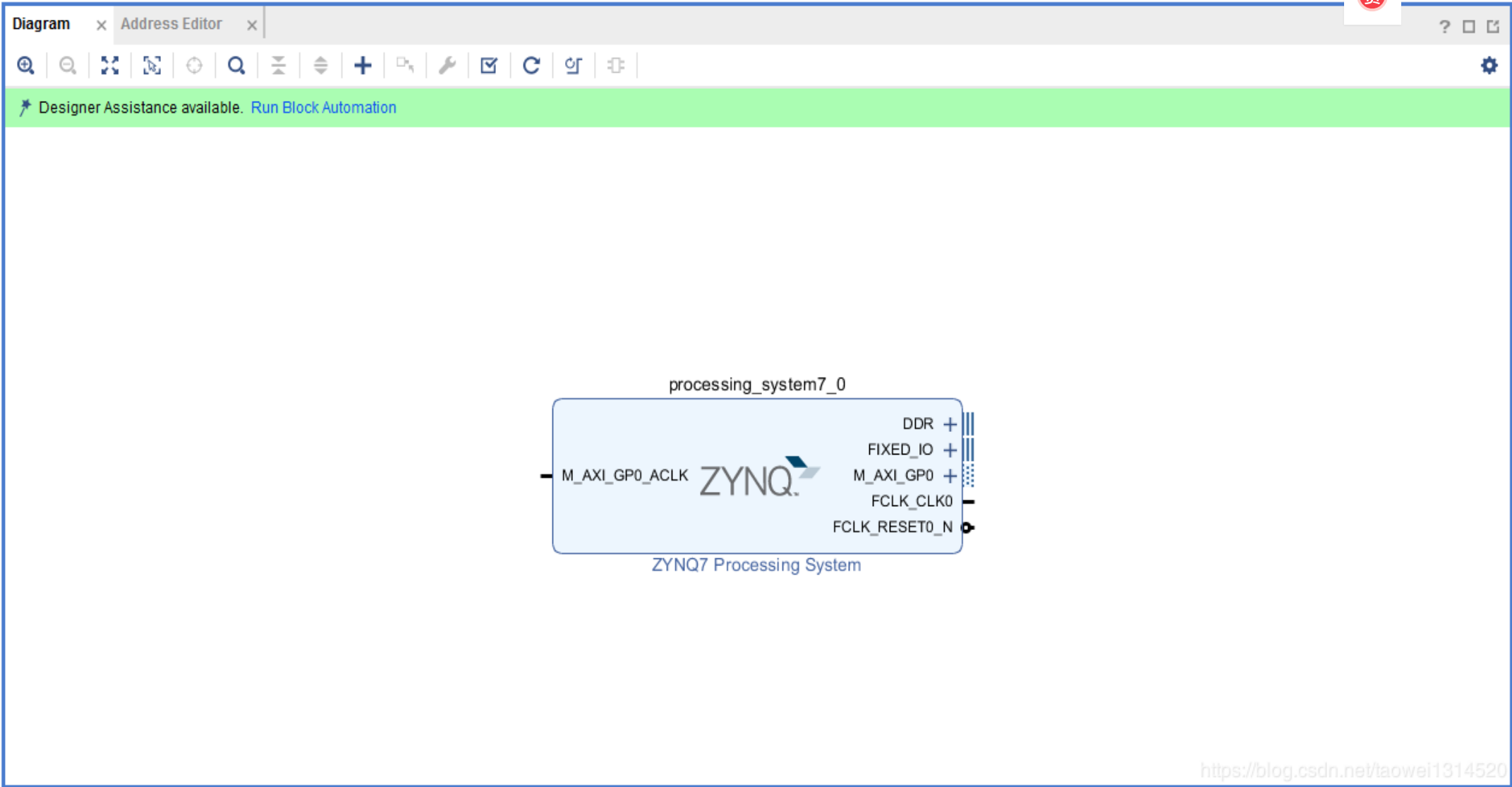
☆ 收藏

开发板环境：vivado 2017.4，开发板型号xc7z020clg400-1，这个工程主要使用DMA进行回环测试

先将DDR内写入数据，然后DMA通过MM2S将数据从DDR读出并写入到fifo中，再通过S2MM将数据从fifo中读出写入到DDR中构成一个回环。

step1 调用一个zynq核并配置

调用zynq核



勾选HP0

👍

🔗

💬

☆

📱

<

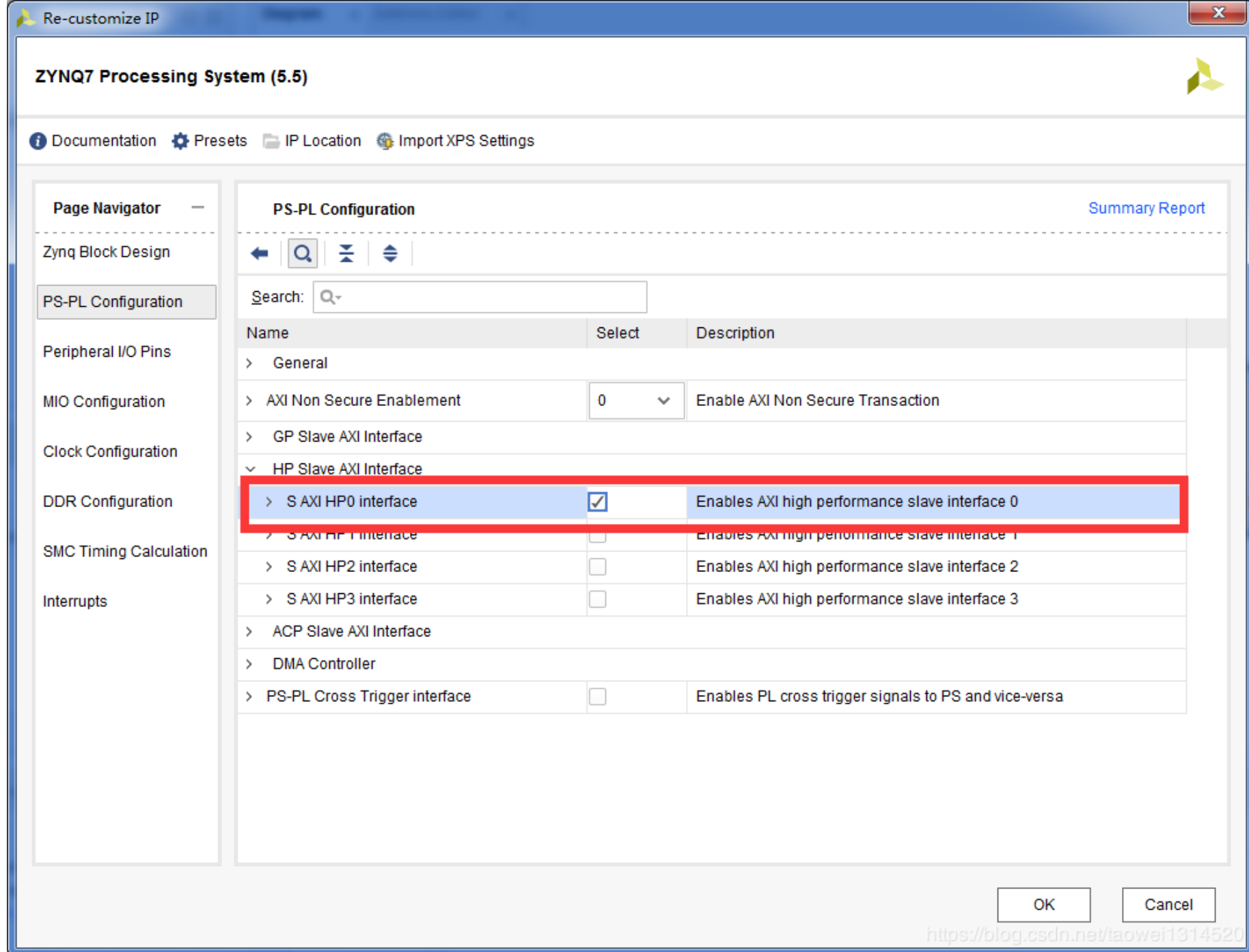
>

赏

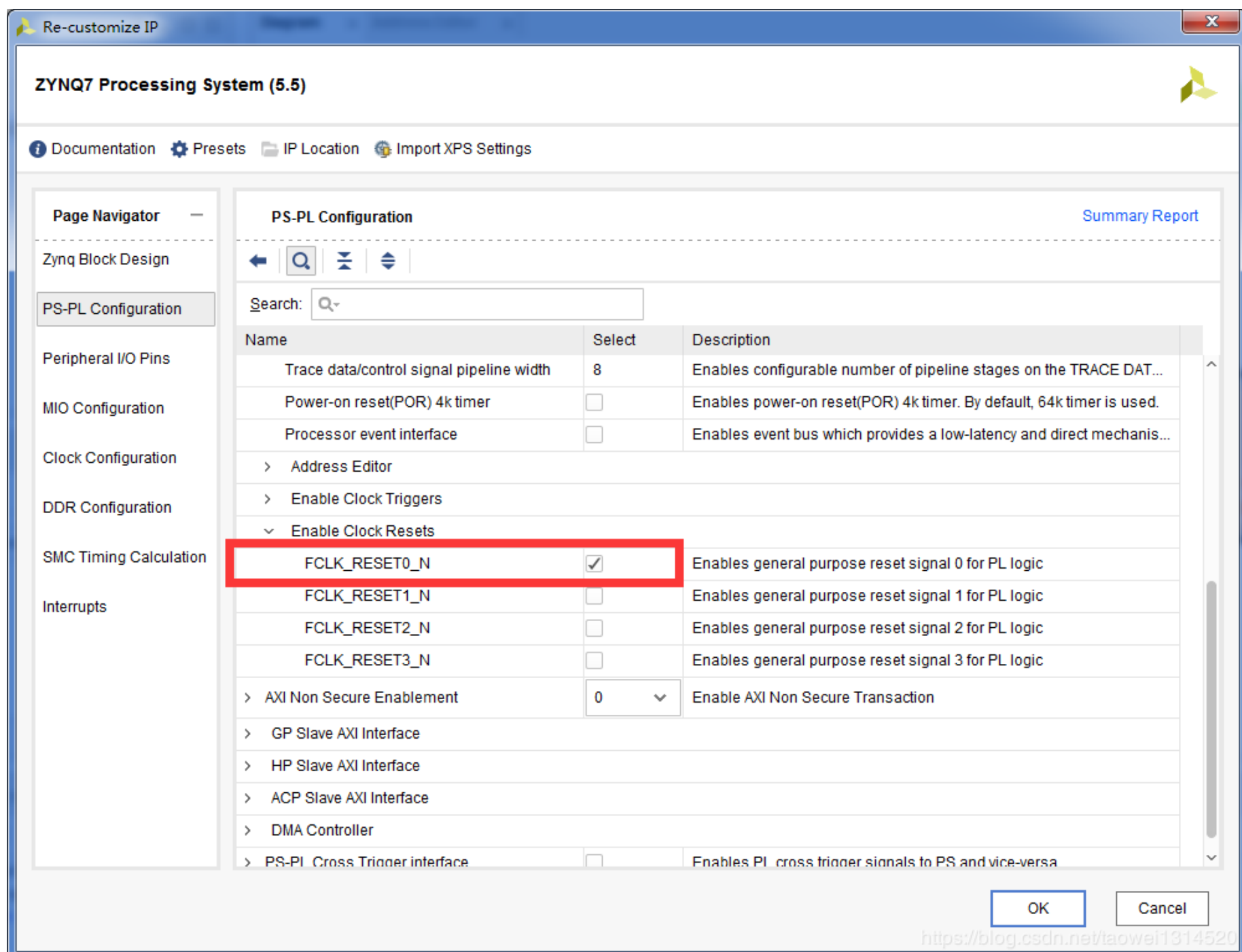
展开

🔊

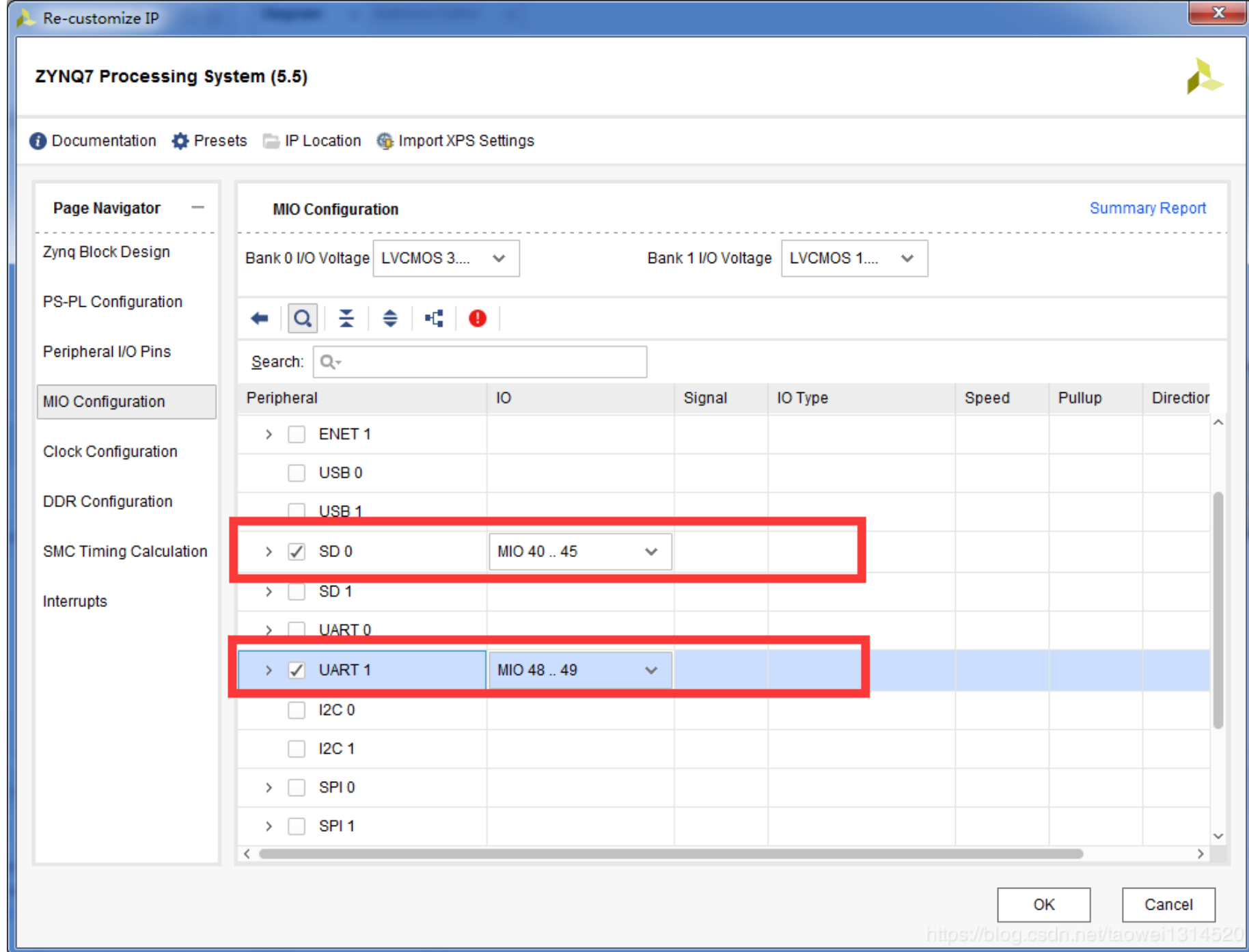
举报



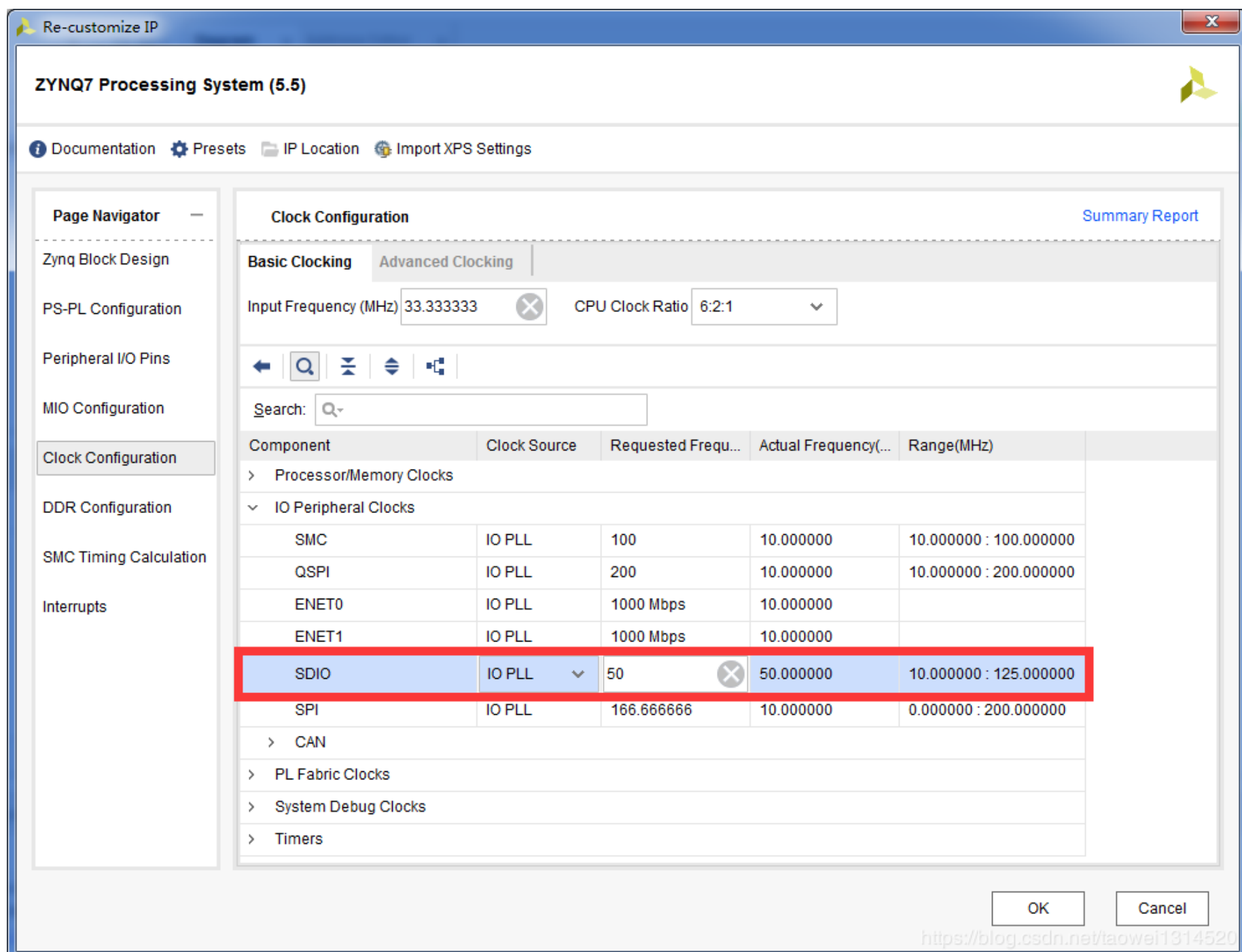
勾选reset管脚



勾选SD卡和uart (不同的开发板会有所差异)



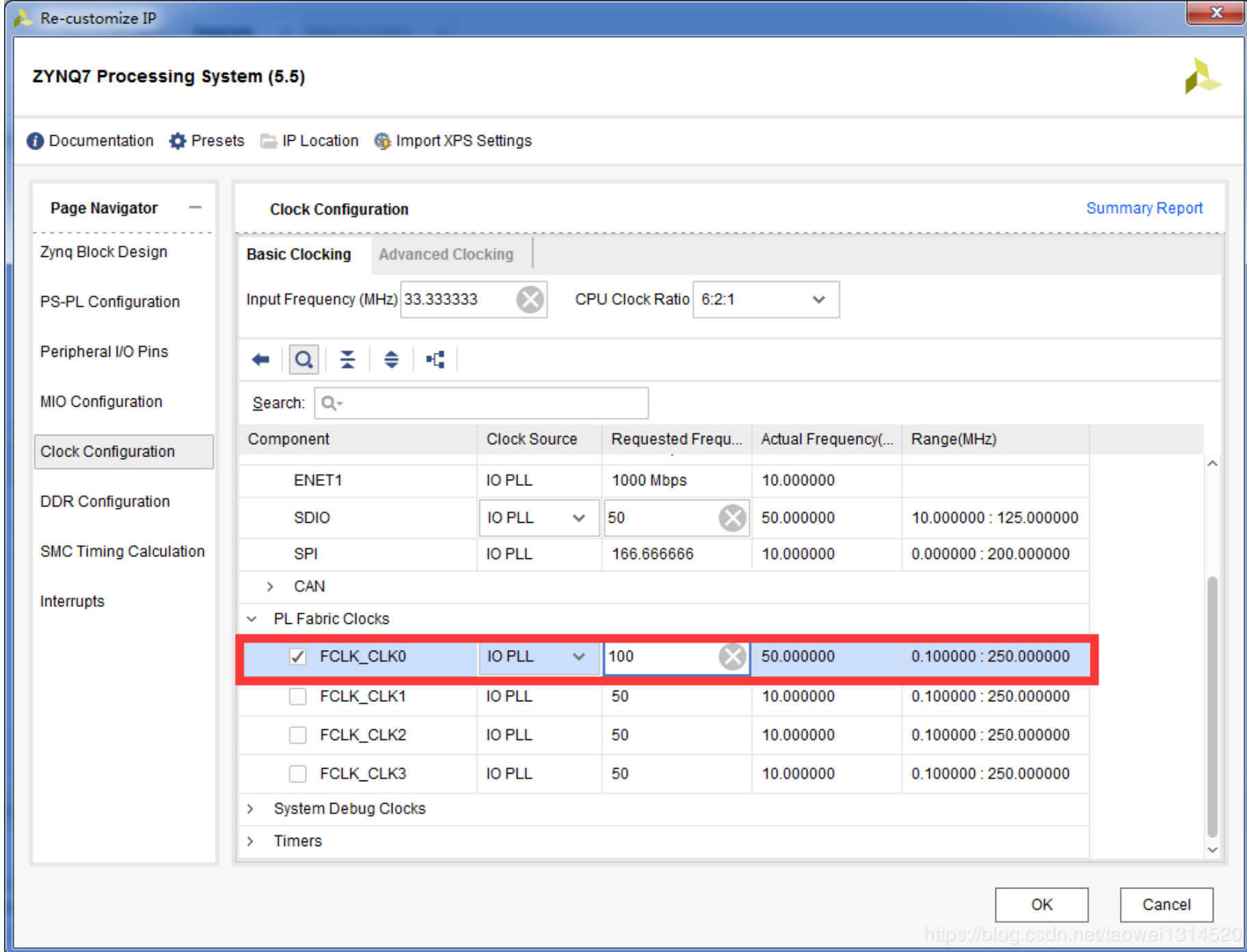
将SDIO设置为50M (不同的开发板会有所差异)



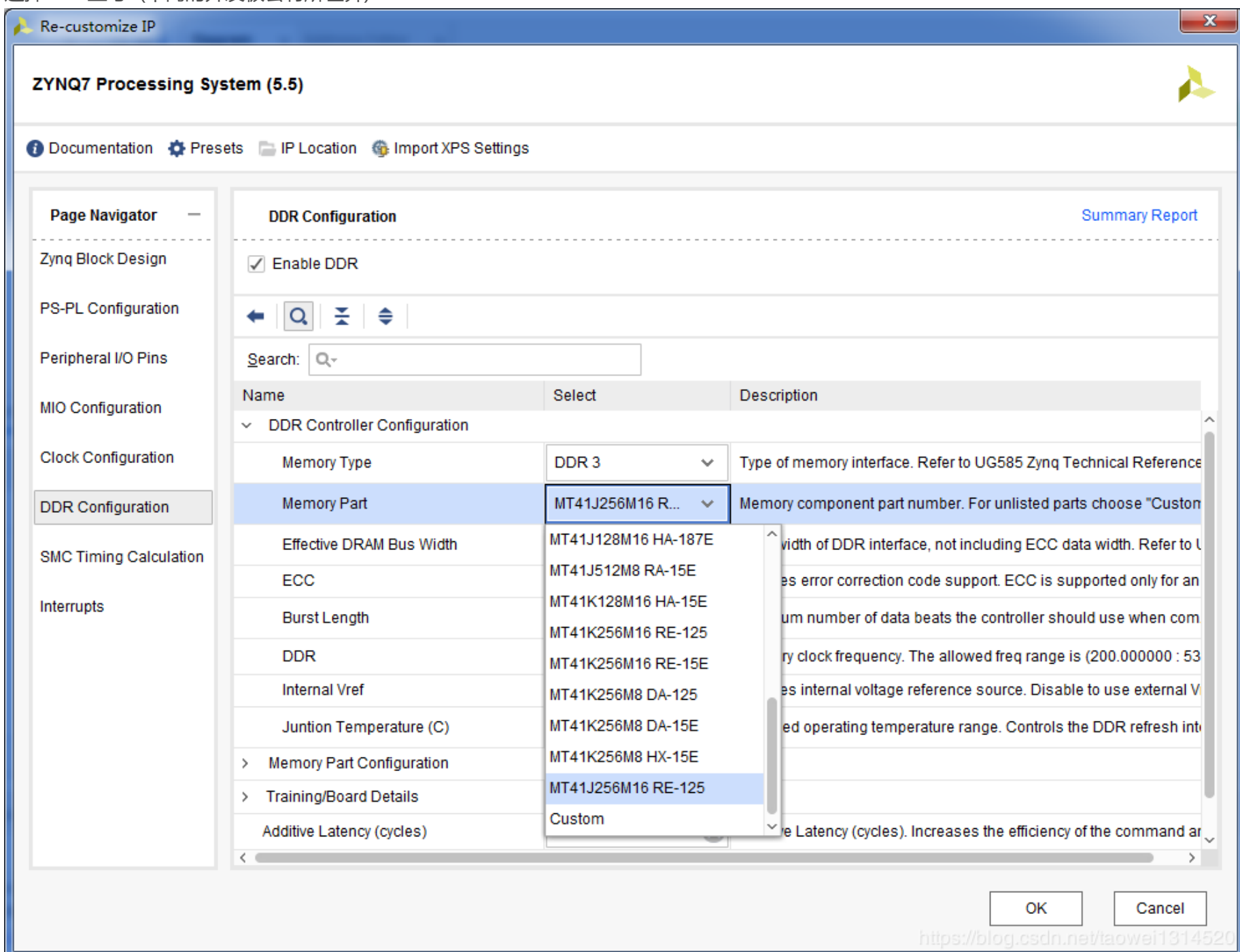
将FCLK0设置为100M (不同的开发板会有所差异)



举报



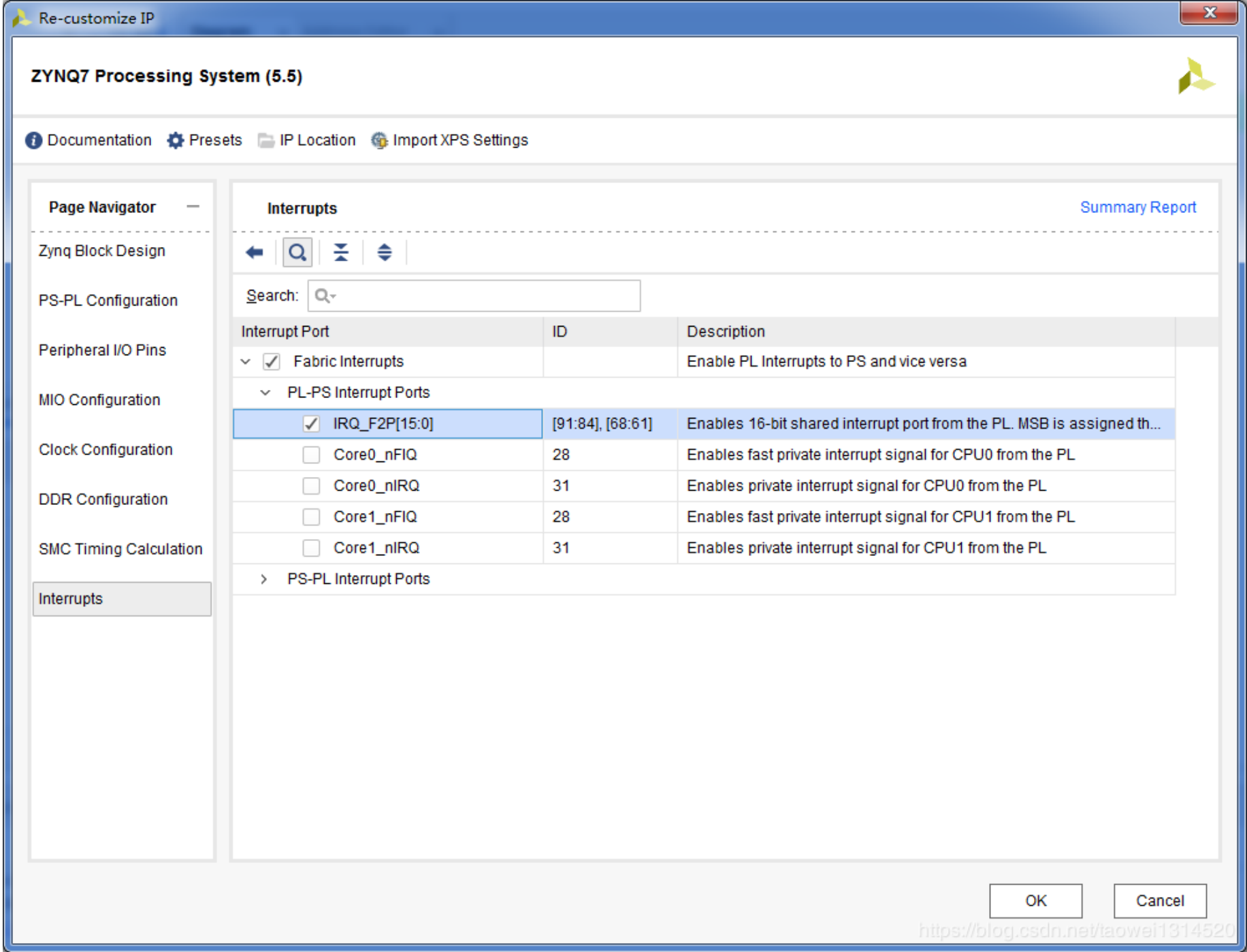
选择DDR型号（不同的开发板会有所差异）



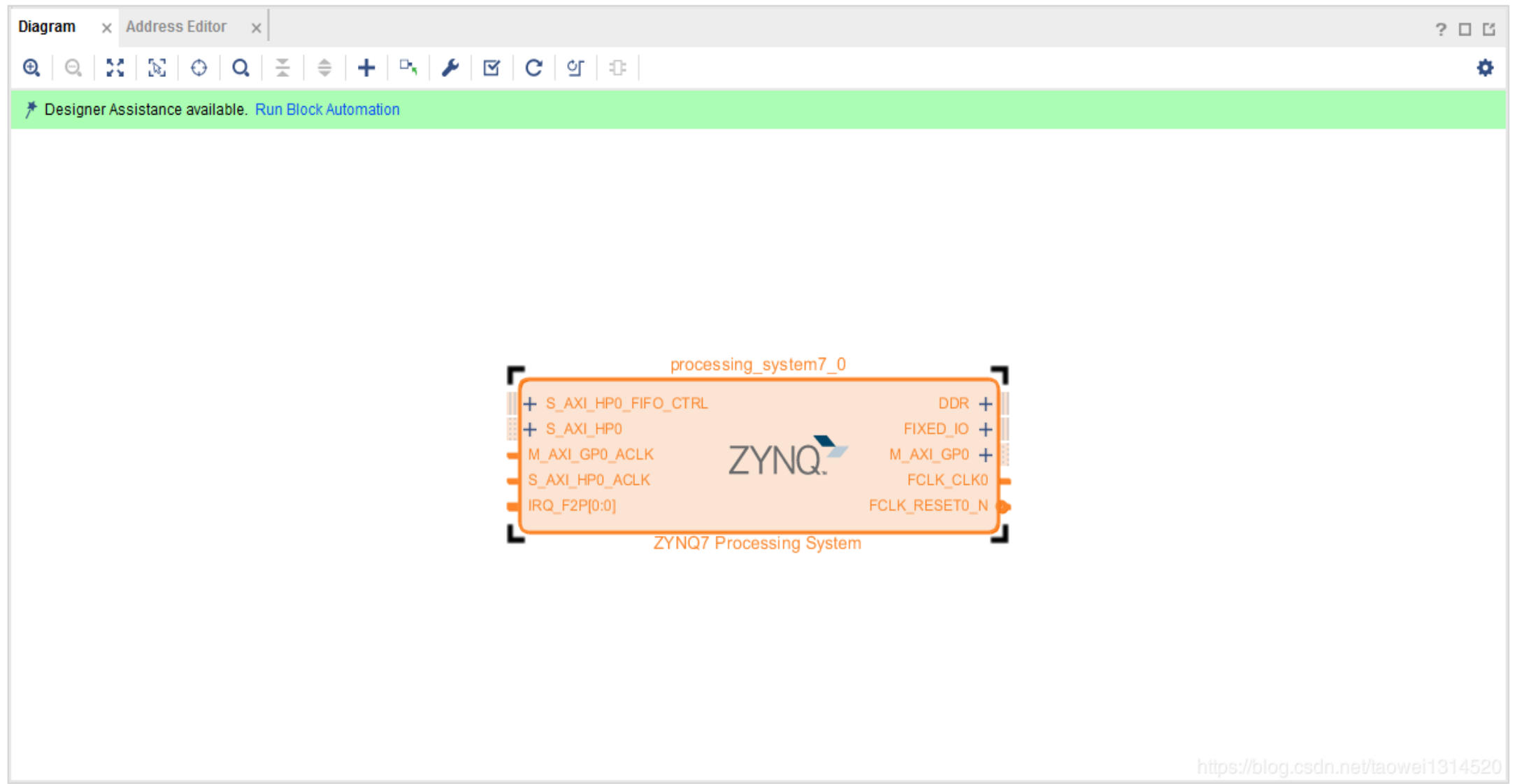
勾选PL中断



举报

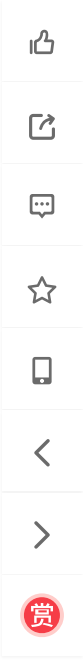


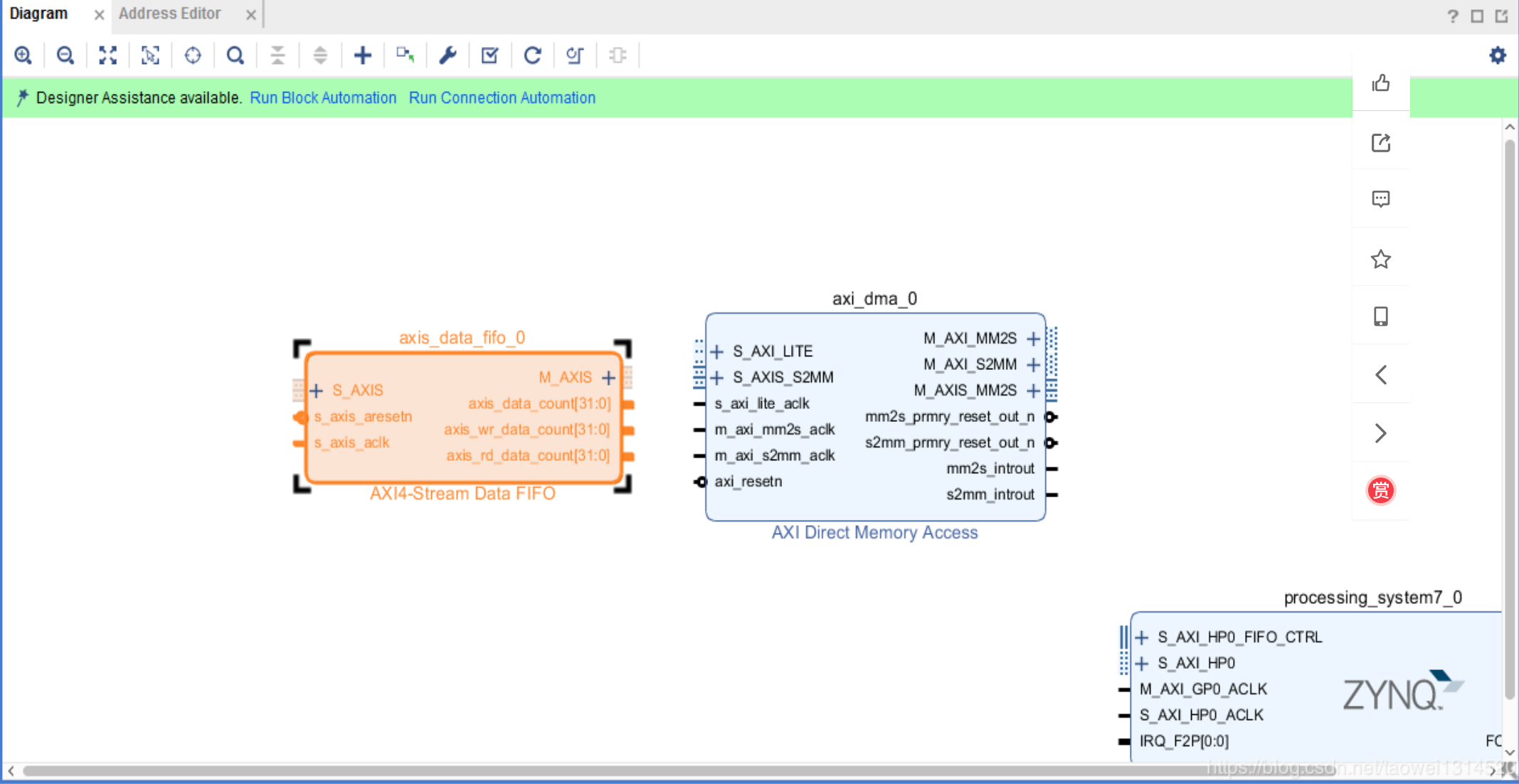
配置完成后，如下图所示



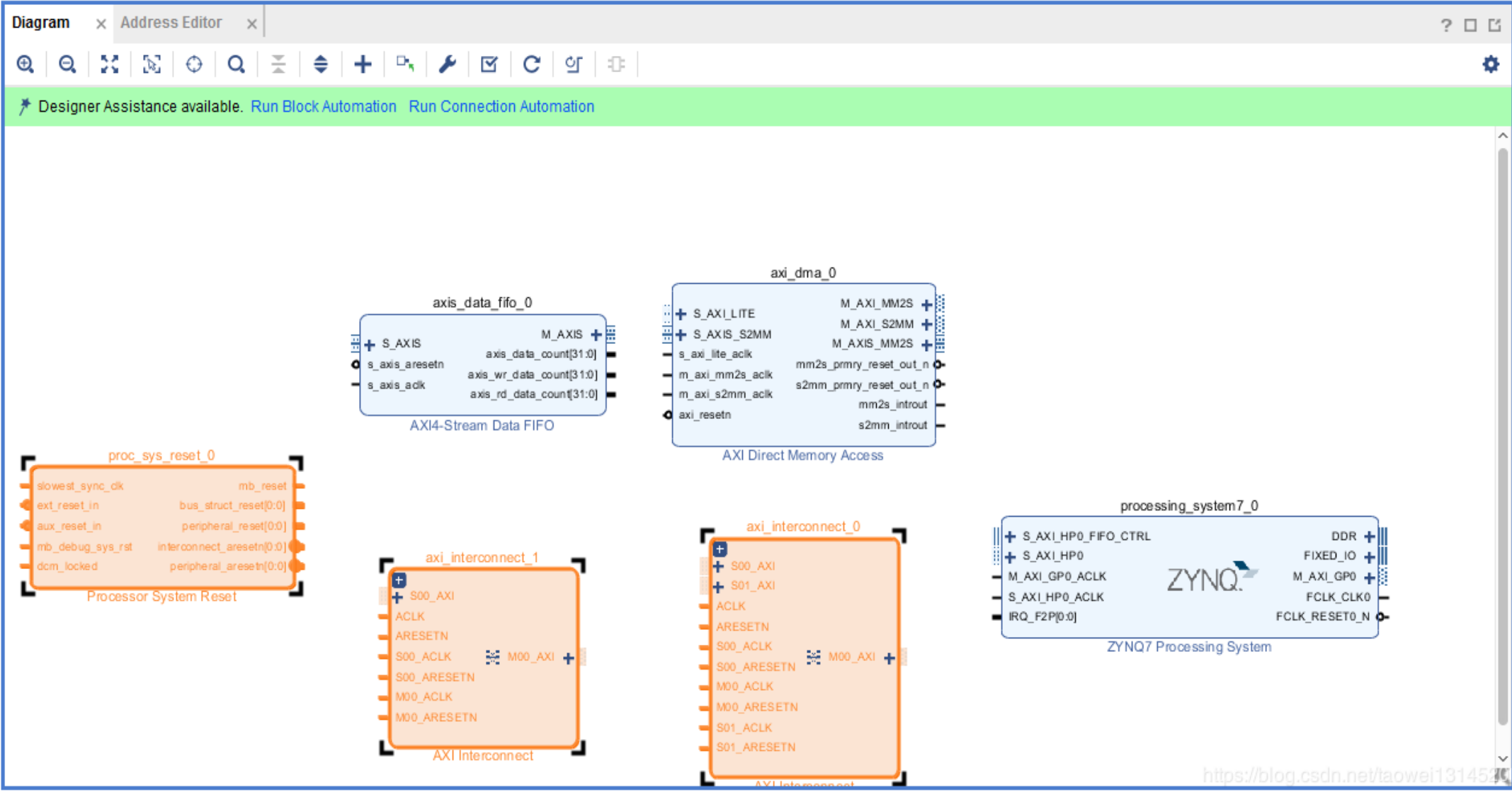
step2 调用dma和fifo进行配置，并将各个模块连接起来

调用一个dma核

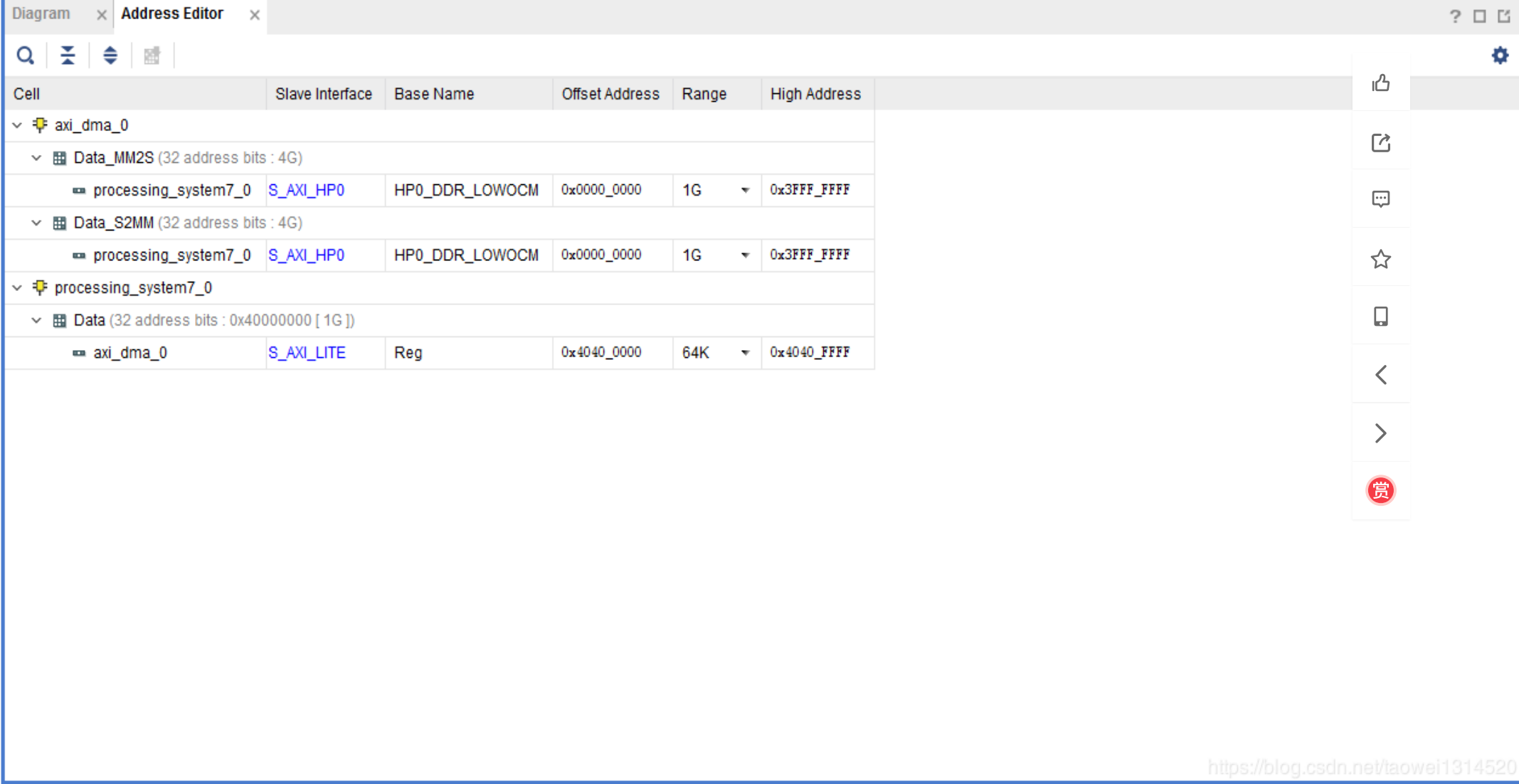




调用System Reset 、 axi_interconnect按照如下配置就可以了

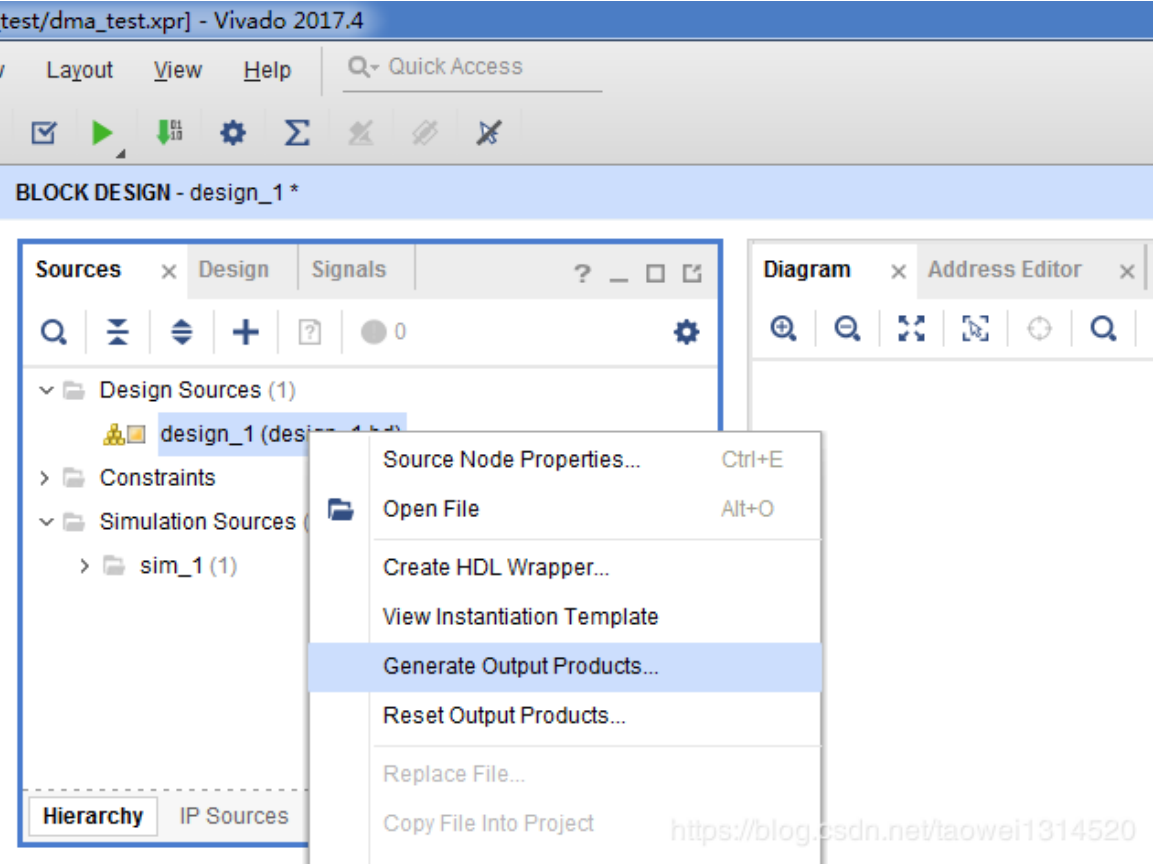


调用一个concat核，这里直接用默认配置

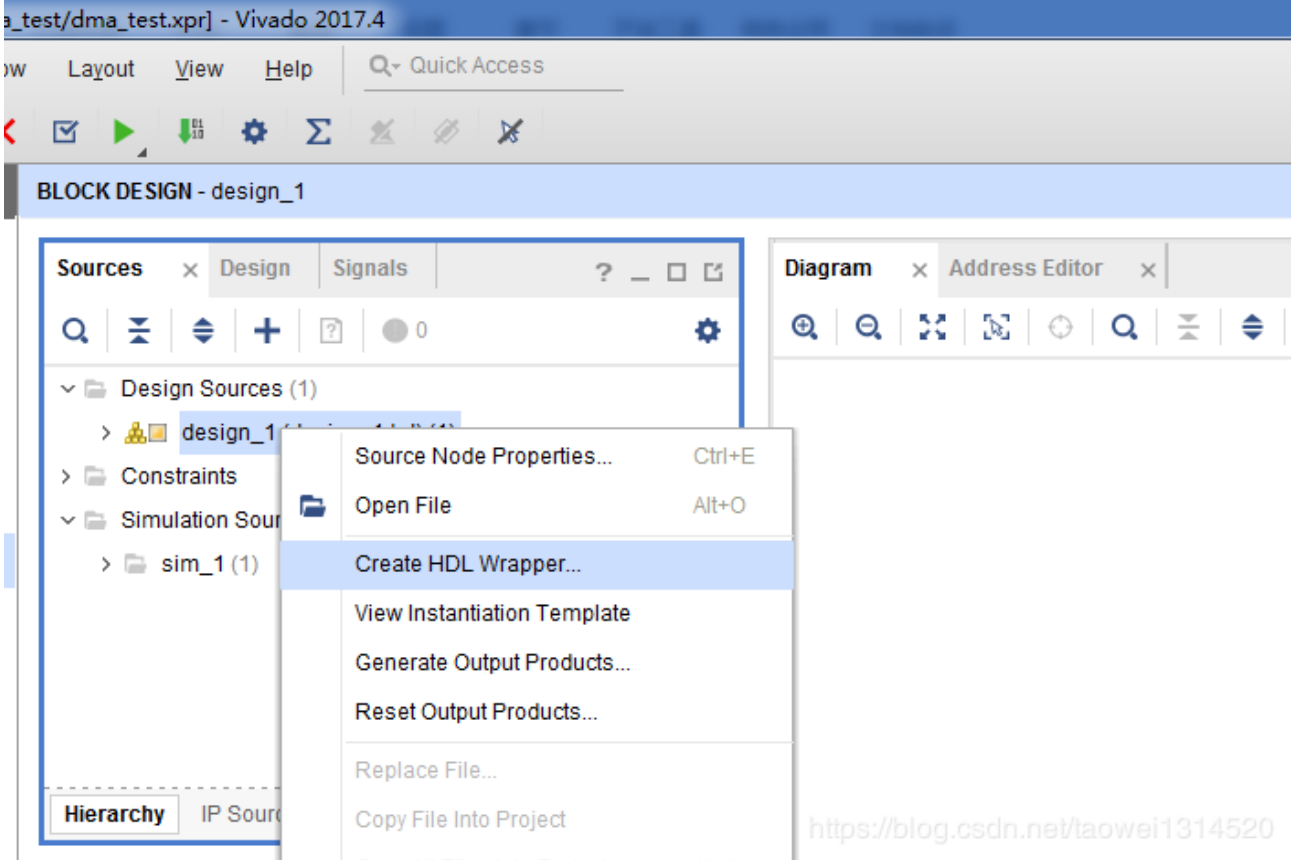


step3 综合、生成顶层文件、生成bit文件

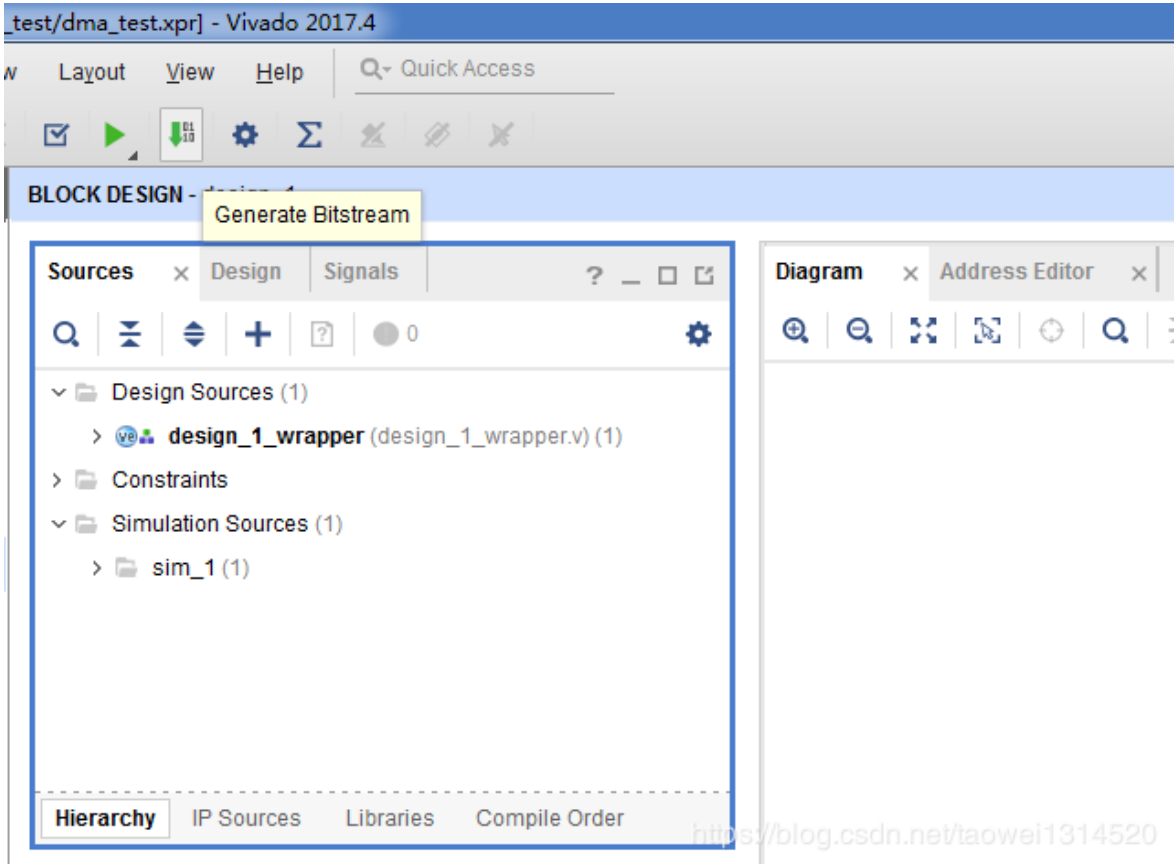
综合



生成顶层文件

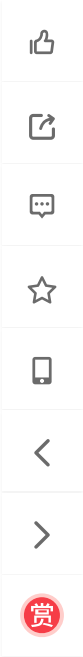


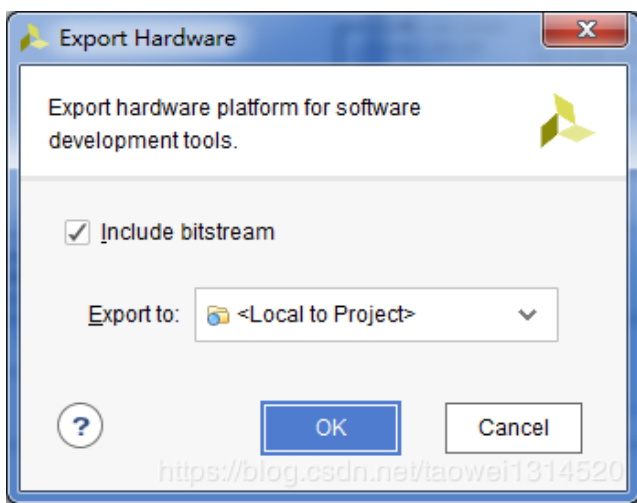
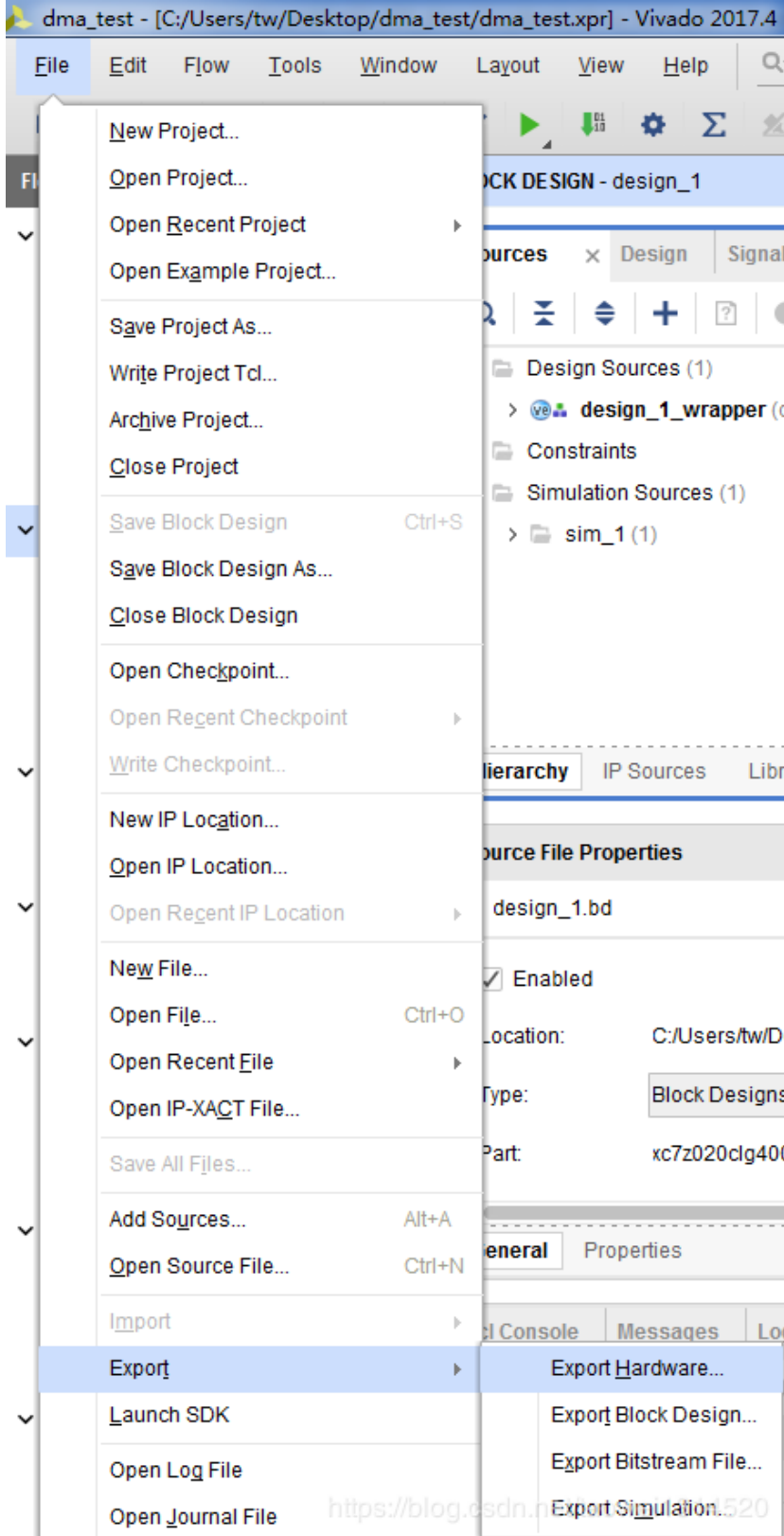
生成bit文件



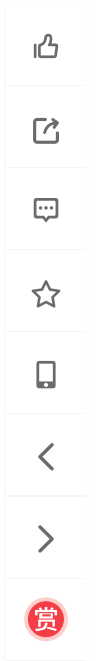
step4 导出硬件配置，打开SDK

导出硬件配置

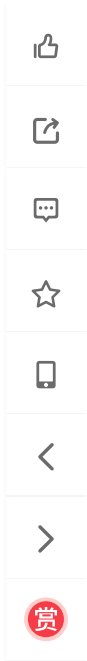
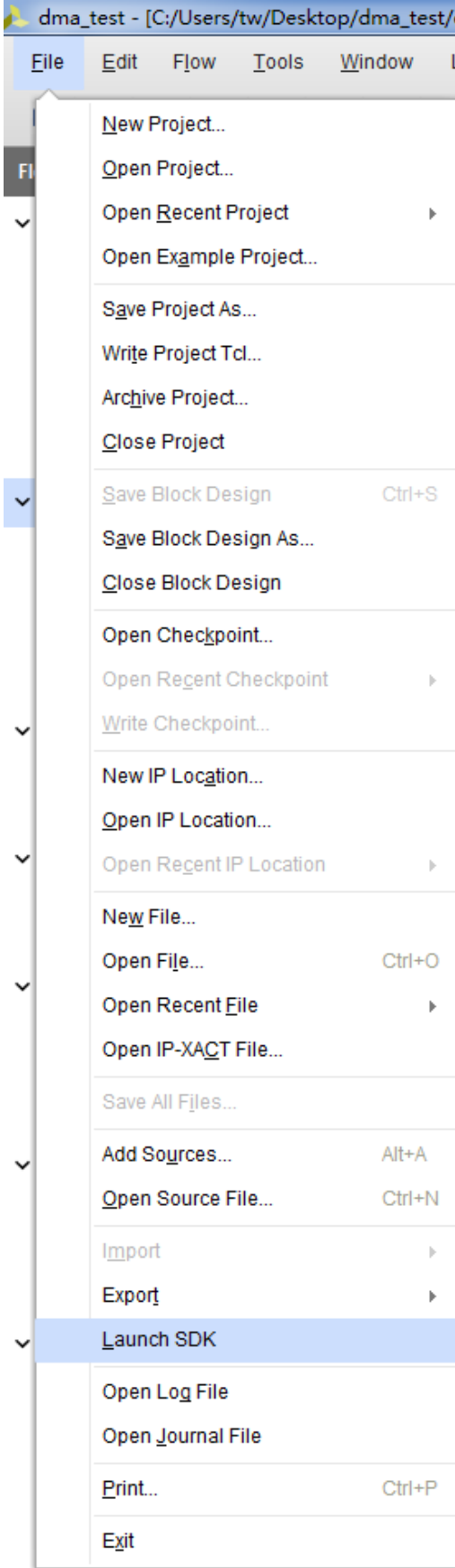




打开SDK

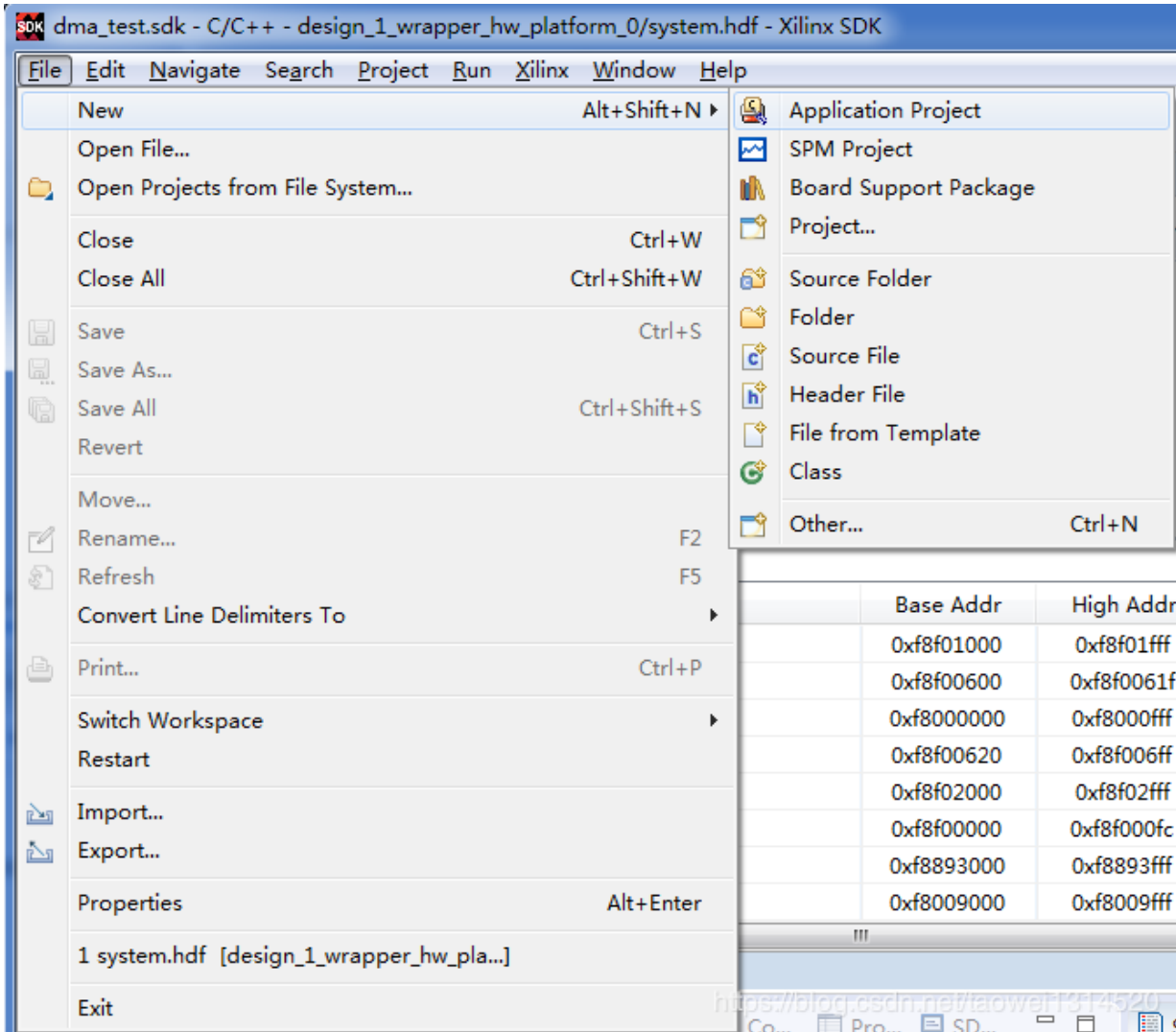


举报



step5 新建fsbl和新建一个dma_test工程

新建fsbl



SDK

New Project

Application Project

Create a managed make application project.

Project name: fsbl

☒ Use default location

Location: C:\Users\tw\Desktop\dma_test\dma_test.sdk\fsbl

Browse...

Choose file system: default

OS Platform: standalone

Target Hardware

Hardware Platform: design_1_wrapper_hw_platform_0

New...

Processor: ps7_cortexa9_0

Target Software

Language: ☒ C ☐ C++

Compiler: 32-bit

Hypervisor Guest: N/A

Board Support Package: ☒ Create New fsbl_bsp ☐ Use existing

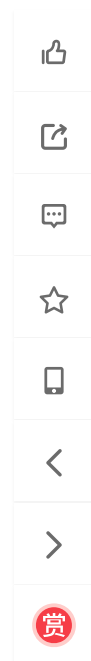
?

< Back

Next >

Finish

Cancel



SDK

New Project

Templates

Create one of the available templates to generate a fully-functioning application project.

Available Templates:

Dhrystone
Empty Application
Hello World
lwIP Echo Server
Memory Tests
OpenAMP echo-test
OpenAMP matrix multiplication Demo
OpenAMP RPC Demo
Peripheral Tests
RSA Authentication App
Zynq DRAM tests
Zynq FSBL

First Stage Bootloader (FSBL) for Zynq. The FSBL configures the FPGA with HW bit stream (if it exists) and loads the Operating System (OS) Image or Standalone (SA) Image or 2nd Stage Boot Loader image from the non-volatile memory (NAND/NOR/QSPI) to RAM (DDR) and starts executing it. It supports multiple partitions, and each partition can be a code image or a bit stream.

?

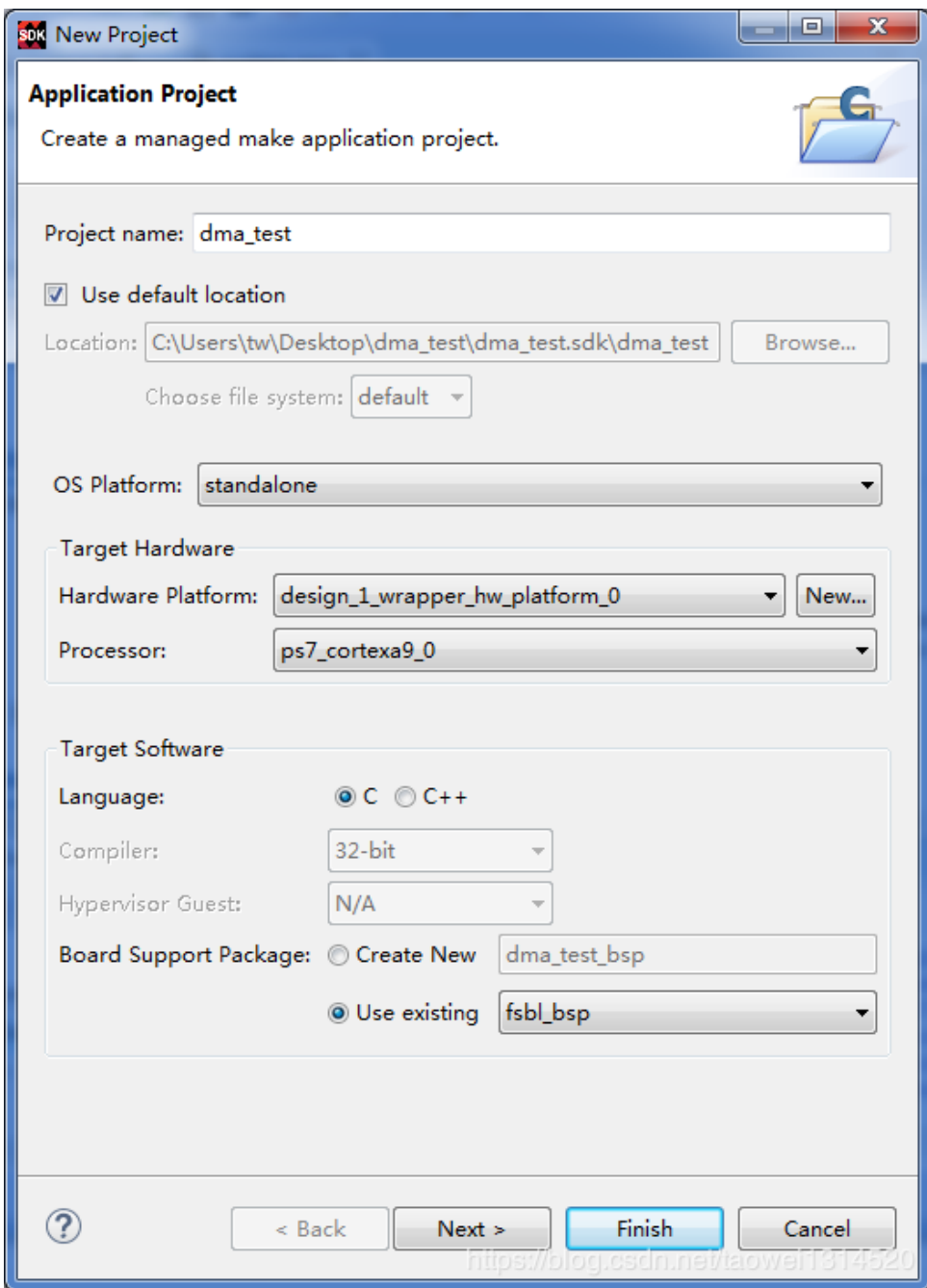
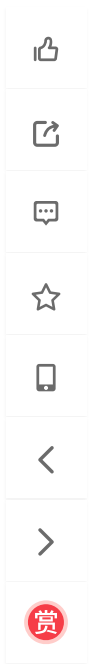
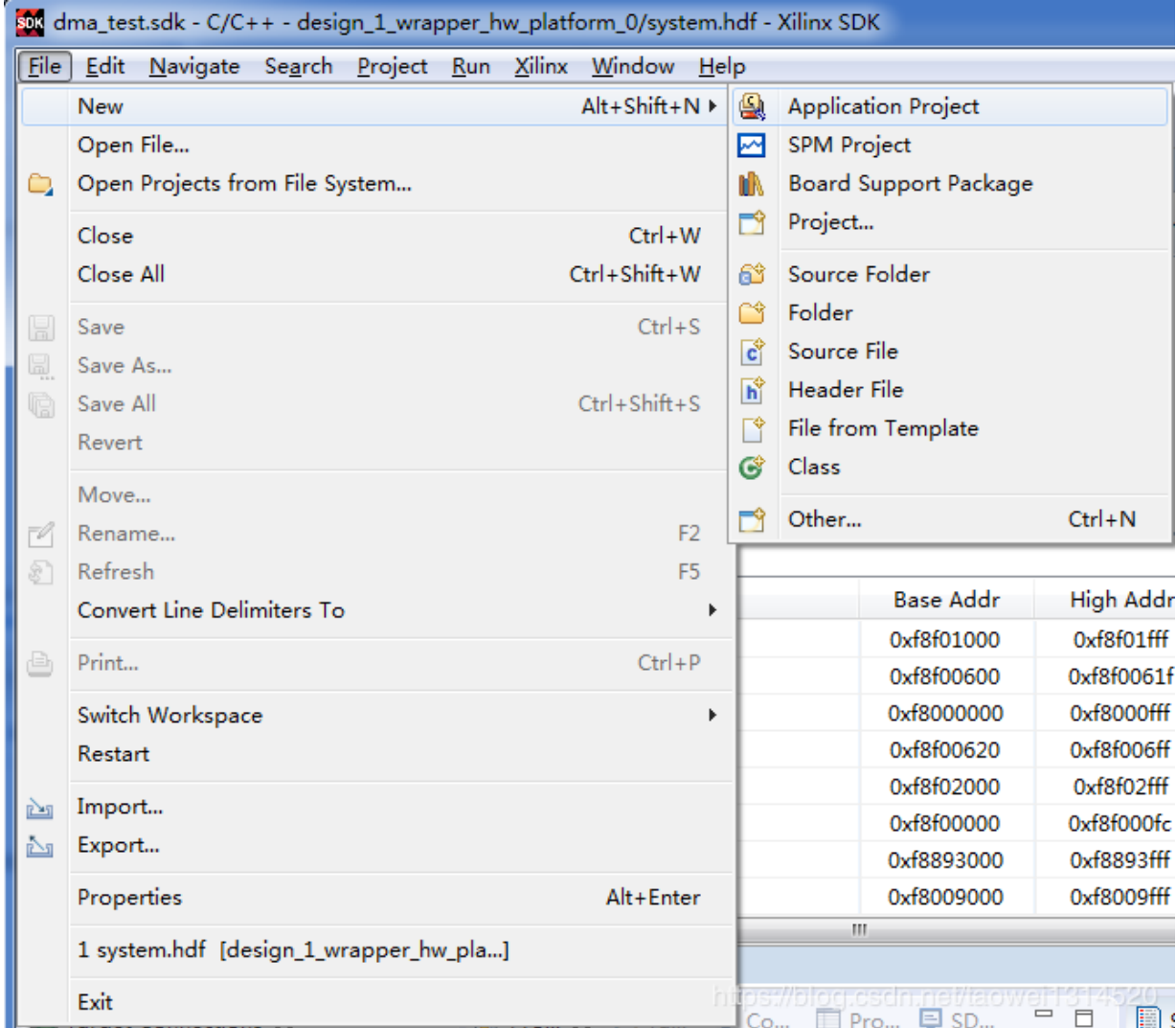
< Back

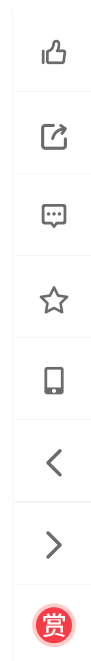
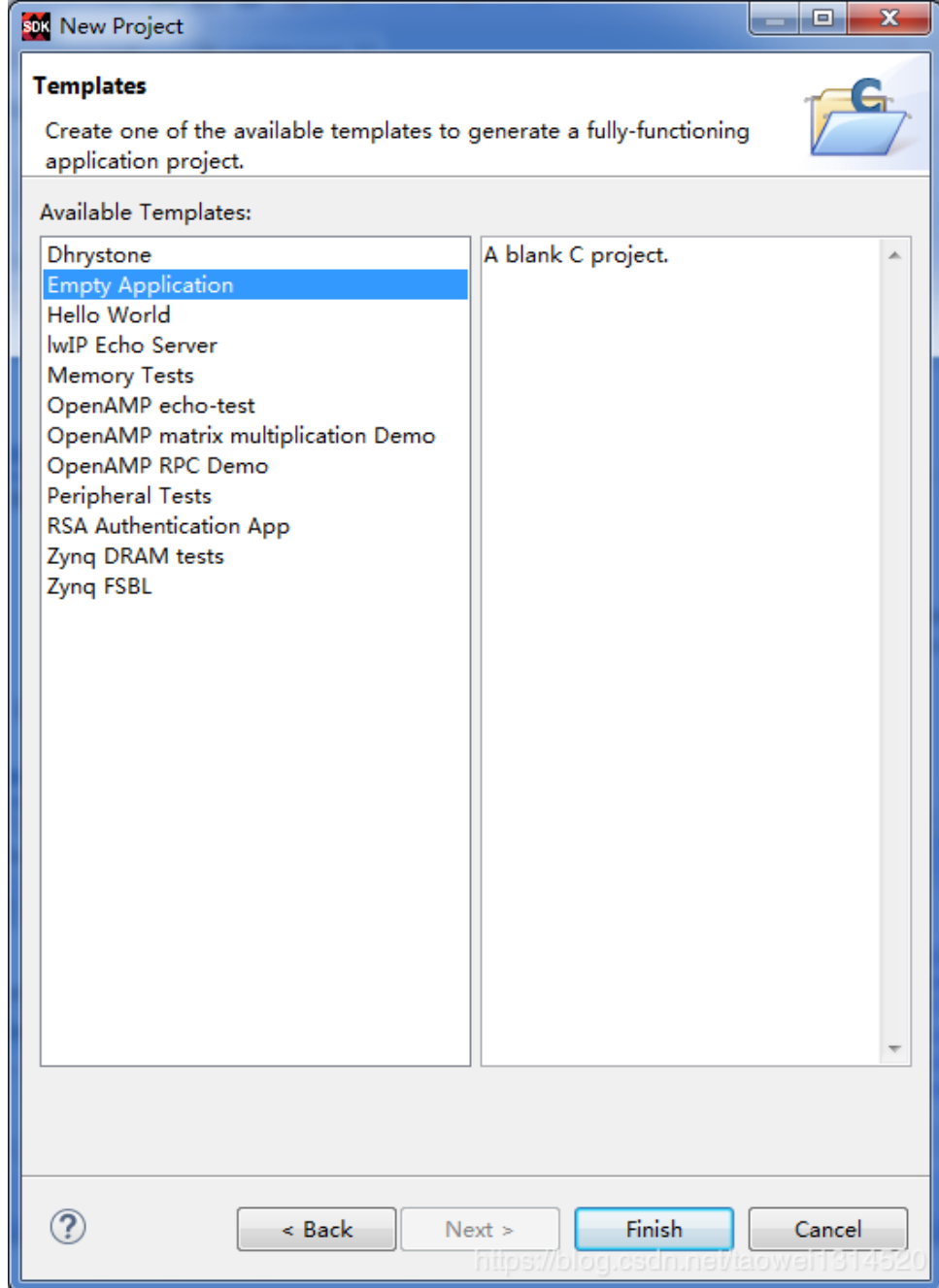
Next >

Finish

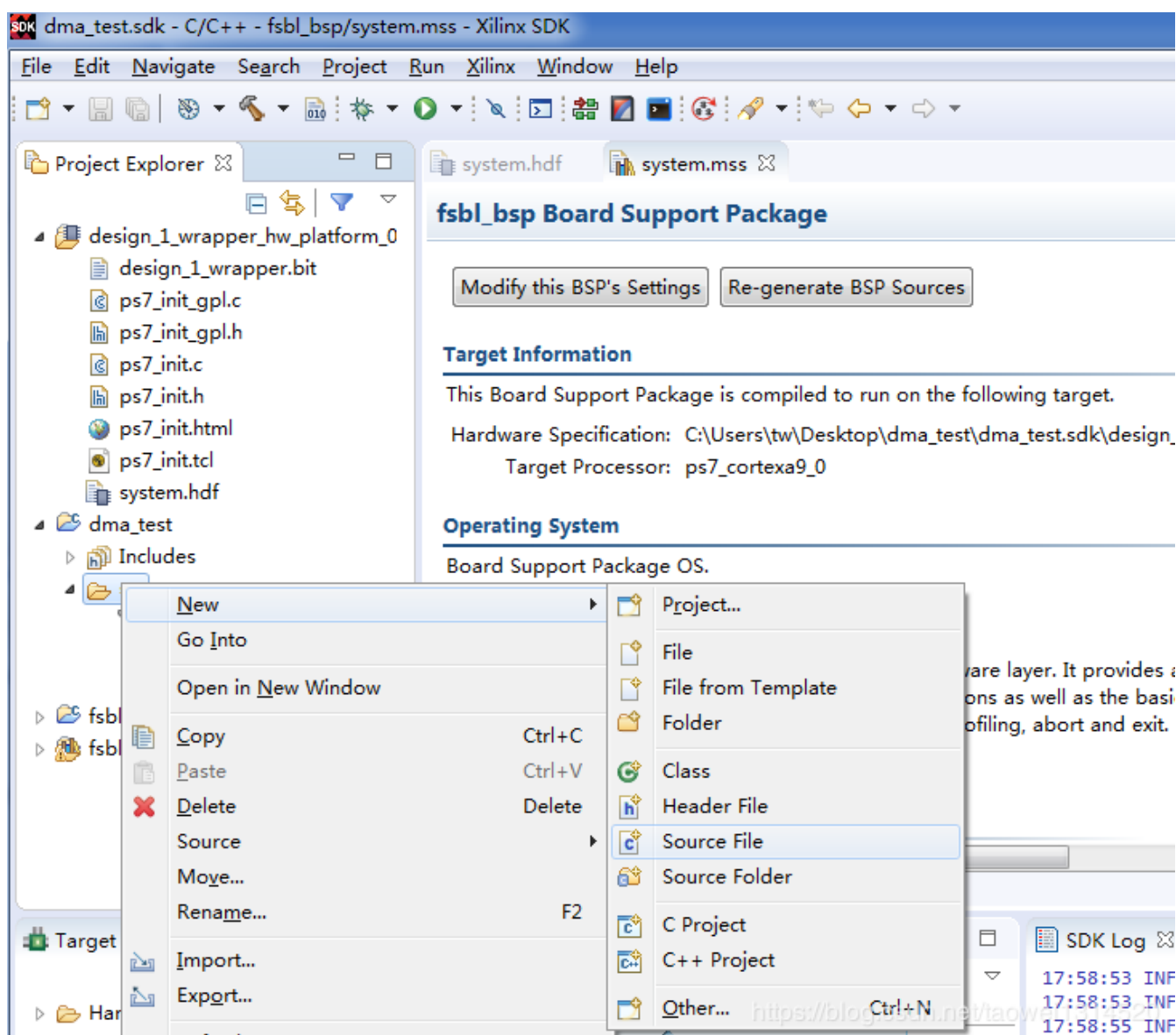
Cancel

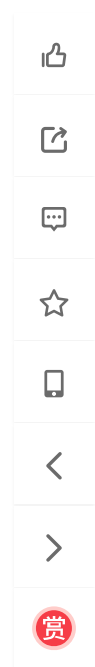
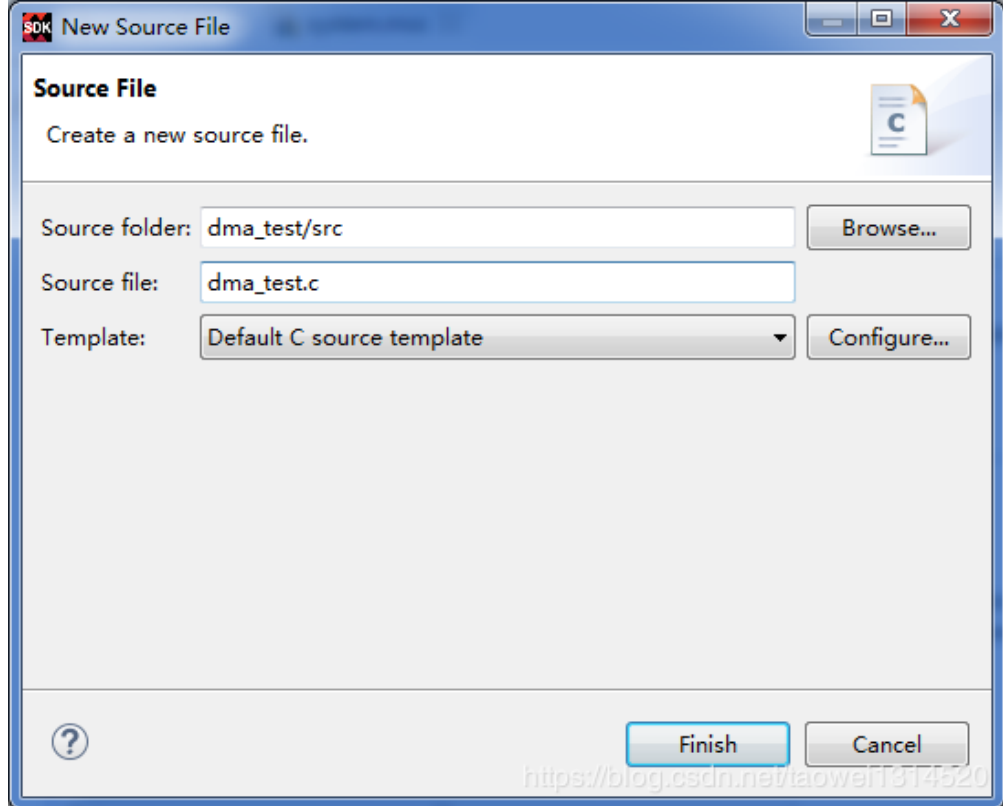






在dma_test工程的src目录下生成一个dma_test.c文件





再将这个主程序复制到这个dma_test.c文件中

```
1
2
3 #include "xaxidma.h"
4 #include "xparameters.h"
5 #include "xil_printf.h"
6 #include "xscugic.h"
7
8
9 #define DMA_DEV_ID          XPAR_AXIDMA_0_DEVICE_ID
10 #define INT_DEVICE_ID      XPAR_SCUGIC_SINGLE_DEVICE_ID
11 #define INTR_ID            XPAR_FABRIC_AXI_DMA_0_S2MM_INTROUT_INTR
12
13 #define MAX_PKT_LEN        32
14
15 #define TEST_START_VALUE    0x0
16
17 #define NUMBER_OF_TRANSFERS 1
18
19 /*
20  * Function declaration
21  */
22 int XAxiDma_Setup(u16 DeviceId);
23 static int CheckData(void);
24 int SetInterruptInit(XScuGic *InstancePtr, u16 IntrID, XAxiDma *XAxiDmaPtr) ;
25
26 XScuGic INST ;
27
28 XAxiDma AxiDma;
29
30 u8 TxBufferPtr[MAX_PKT_LEN] ;
31 u8 RxBufferPtr[MAX_PKT_LEN] ;
32
33
34 int main()
35 {
36     int Status;
37
38     xil_printf("\r\n--- Entering main() --- \r\n");
39
40     Status = XAxiDma_Setup(DMA_DEV_ID);
41
42     if (Status != XST_SUCCESS) {
43         xil_printf("XAxiDma Test Failed\r\n");
44         return XST_FAILURE;
45     }
46
47     xil_printf("Successfully Ran XAxiDma Test\r\n");
48
49     xil_printf("--- Exiting main() --- \r\n");
50
51     return XST_SUCCESS;
52
53 }
54
55
```



```
56
57 | int SetInterruptInit(XScuGic *InstancePtr, u16 IntrID, XAxiDma *XAxiDmaPtr)
58 {
59
60     XScuGic_Config * Config ;
61     int Status ;
62
63     Config = XScuGic_LookupConfig(INT_DEVICE_ID) ;
64
65     Status = XScuGic_CfgInitialize(&INST, Config, Config->CpuBaseAddress) ;
66     if (Status != XST_SUCCESS)
67         return XST_FAILURE ;
68
69     Status = XScuGic_Connect(InstancePtr, IntrID,
70                             (Xil_ExceptionHandler)CheckData,
71                             XAxiDmaPtr) ;
72
73     if (Status != XST_SUCCESS) {
74         return Status;
75     }
76
77     XScuGic_Enable(InstancePtr, IntrID) ;
78
79     Xil_ExceptionInit();
80     Xil_ExceptionRegisterHandler(XIL_EXCEPTION_ID_INT,
81                                 (Xil_ExceptionHandler) XScuGic_InterruptHandler,
82                                 InstancePtr);
83
84     Xil_ExceptionEnable();
85
86
87     return XST_SUCCESS ;
88
89 }
90
91
92 int XAxiDma_Setup(u16 DeviceId)
93 {
94     XAxiDma_Config *CfgPtr;
95     int Status;
96     int Tries = NUMBER_OF_TRANSFERS;
97     int Index;
98     u8 Value;
99
100     /* Initialize the XAxiDma device.
101      */
102     CfgPtr = XAxiDma_LookupConfig(DeviceId);
103     if (!CfgPtr) {
104         xil_printf("No config found for %d\r\n", DeviceId);
105         return XST_FAILURE;
106     }
107
108     Status = XAxiDma_CfgInitialize(&AxiDma, CfgPtr);
109     if (Status != XST_SUCCESS) {
110         xil_printf("Initialization failed %d\r\n", Status);
111         return XST_FAILURE;
112     }
113
114     if(XAxiDma_HasSg(&AxiDma)){
115         xil_printf("Device configured as SG mode \r\n");
116         return XST_FAILURE;
117     }
118
119     Status = SetInterruptInit(&INST,INTR_ID, &AxiDma) ;
120     if (Status != XST_SUCCESS)
121         return XST_FAILURE ;
122
123     /* Disable MM2S interrupt, Enable S2MM interrupt */
124     XAxiDma_IntrEnable(&AxiDma, XAXIDMA_IRQ_IOC_MASK,
125                       XAXIDMA_DEVICE_TO_DMA);
126     XAxiDma_IntrDisable(&AxiDma, XAXIDMA_IRQ_ALL_MASK,
127                        XAXIDMA_DMA_TO_DEVICE);
128
129     Value = TEST_START_VALUE;
130
131     for(Index = 0; Index < MAX_PKT_LEN; Index ++) {
132         TxBufferPtr[Index] = Value;
133
134         Value = (Value + 1) & 0xFF;
```

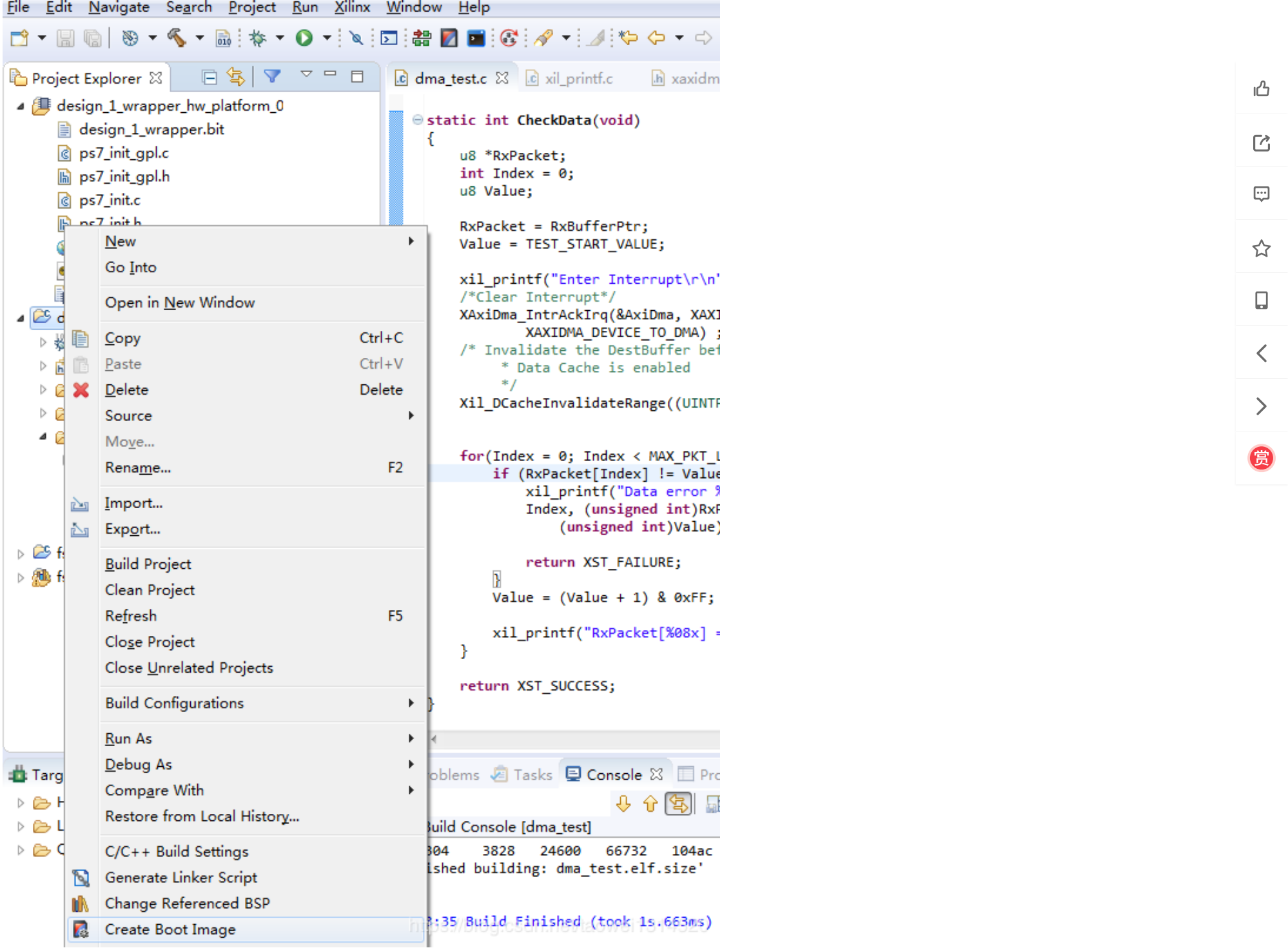


举报

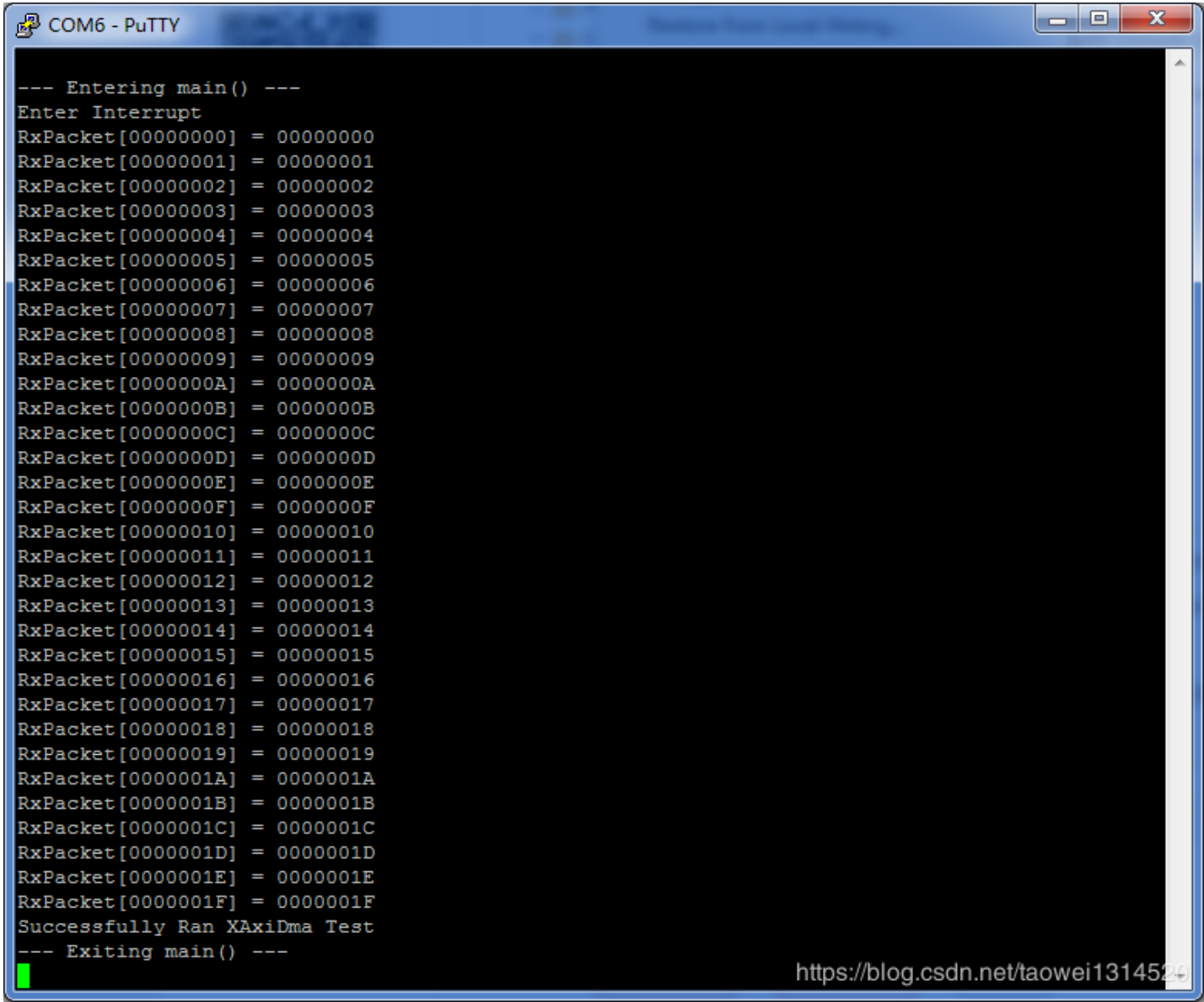
```
135 }
136 |         /* Flush the SrcBuffer before the DMA transfer, in case the Data Cache
137    * is enabled
138    */
139    Xil_DCacheFlushRange((UINTPTR)TxBufferPtr, MAX_PKT_LEN);
140
141    for(Index = 0; Index < Tries; Index++) {
142
143        Status = XAxiDma_SimpleTransfer(&AxiDma,(UINTPTR) TxBufferPtr,
144                                         MAX_PKT_LEN, XAXIDMA_DMA_TO_DEVICE);
145
146        if (Status != XST_SUCCESS) {
147            return XST_FAILURE;
148        }
149
150        Status = XAxiDma_SimpleTransfer(&AxiDma,(UINTPTR) RxBufferPtr,
151                                         MAX_PKT_LEN, XAXIDMA_DEVICE_TO_DMA);
152
153
154        if (Status != XST_SUCCESS) {
155            return XST_FAILURE;
156        }
157
158
159        while ((XAxiDma_Busy(&AxiDma,XAXIDMA_DEVICE_TO_DMA)) ||
160              (XAxiDma_Busy(&AxiDma,XAXIDMA_DMA_TO_DEVICE)))
161        {
162            /* Wait */
163        }
164
165
166    }
167
168    /* Test finishes successfully
169    */
170    return XST_SUCCESS;
171 }
172
173
174 static int CheckData(void)
175 {
176     u8 *RxPacket;
177     int Index = 0;
178     u8 Value;
179
180     RxPacket = RxBufferPtr;
181     Value = TEST_START_VALUE;
182
183     xil_printf("Enter Interrupt\r\n");
184     /*Clear Interrupt*/
185     XAxiDma_IntrAckIrq(&AxiDma, XAXIDMA_IRQ_IOC_MASK,
186                      XAXIDMA_DEVICE_TO_DMA) ;
187     /* Invalidate the DestBuffer before receiving the data, in case the
188        * Data Cache is enabled
189        */
190     Xil_DCacheInvalidateRange((UINTPTR)RxPacket, MAX_PKT_LEN);
191
192
193     for(Index = 0; Index < MAX_PKT_LEN; Index++) {
194         if (RxPacket[Index] != Value) {
195             xil_printf("Data error %d: %x/%x\r\n",
196                      Index, (unsigned int)RxPacket[Index],
197                      (unsigned int)Value);
198
199             return XST_FAILURE;
200         }
201         Value = (Value + 1) & 0xFF;
202
203         xil_printf("RxPacket[%08x] = %08x\n\r",Index,RxPacket[Index]);
204     }
205
206     return XST_SUCCESS;
207 }
```



举报



将BOOT.bin文件拷贝到开发板运行，可以看到串口打印了32个8位数据，这个是接收buffer里的数据



下面是对程序的部分分析

1.

```
1 #define MAX_PKT_LEN 32 //这里是值发送多少个8位数据，这里是发送32个8位数据
2
3 #define TEST_START_VALUE 0x0 //往ddr里写数据，写入的第一个起始数据
4
```



举报

2.

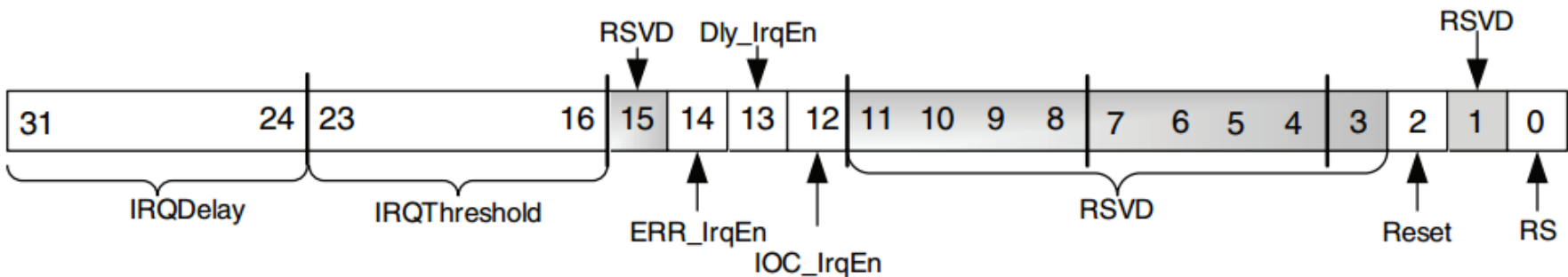
```
1 XAxiDma_IntrEnable(&AxiDma, XAXIDMA_IRQ_IOC_MASK,
2 XAXIDMA_DEVICE_TO_DMA);
3
4 设置S2MM使能中断，这里操作的是0x30这个寄存器，当然这里也不只是设置了中断也配置了其它的控制位
5 具体请参考这个DMA手册
```

具体的设置请参考DMA手册的0x30寄存器设置

Stream to Memory Map Register Detail

S2MM_DMACR (S2MM DMA Control Register - Offset 30h) (C_INCLUDE_SG = 1/0)

This register provides control for the Stream to Memory Map DMA Channel.



DS781_08

Figure 2-8: S2MM DMACR Register

<https://blog.csdn.net/taowei1314520>

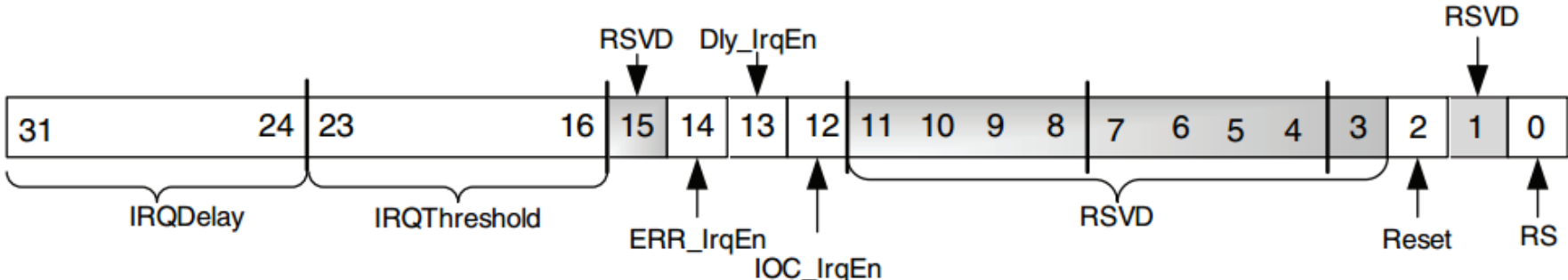
3

```
1 XAxiDma_IntrDisable(&AxiDma, XAXIDMA_IRQ_ALL_MASK,
2 XAXIDMA_DMA_TO_DEVICE);
3
4 关闭MM2S使能中断，这里操作的是0x00这个寄存器，这里不仅是关闭MM2S中断也配置了其它的控制位
5 具体请参考这个DMA手册
```

Memory Map to Stream Register Detail

MM2S_DMACR (MM2S DMA Control Register - Offset 00h) (C_INCLUDE_SG = 1/0)

This register provides control for the Memory Map to Stream DMA Channel.



DS781_04

Figure 2-2: MM2S DMACR Register

<https://blog.csdn.net/taowei1314520>

4

```
1 Value = TEST_START_VALUE;
2
3
4 for(Index = 0; Index < MAX_PKT_LEN; Index ++) {
```



举报

这里往TxBuffer里填充32个8位数据

```
Status = XAxiDma_SimpleTransfer(&AxiDma, (UINTPTR) TxBufferPtr,
                                MAX_PKT_LEN, XAXIDMA_DMA_TO_DEVICE);
```

启动dma发送，将发送buffer里的数据通过MM2S读出并写入到fifo里



```


1         Status = XAxiDma_SimpleTransfer(&AxiDma, (UINTPTR) RxBufferPtr,
2                                         MAX_PKT_LEN, XAXIDMA_DEVICE_TO_DMA);
3
4  启动dma接收，将fifo里数据读出并且通过S2MM接口写入到ddr里

```



```
1 while ((XAxiDma_Busy(&AxiDma,XAXIDMA_DEVICE_TO_DMA)) ||
2         (XAxiDma_Busy(&AxiDma,XAXIDMA_DMA_TO_DEVICE)))
3     {
4
5 检测发送和接收通道是否处于空闲状态，这个类似于while(tx || rx),只有当发送和接收都处于空闲为0时
6 才会跳出while语句，不然就会一直在这里等待
```






3 年

发布了37 篇原创文章 · 获赞 59 · 访问量 15万+

私信


关注






CorelDRAW Graphics Suite 2020

為您帶來市面上最快速、流暢、精準的向量圖形設計軟體。

广告 coreldraw.com



想对作者说点什么

ZYNQ-7000 SoC几种DMA的区别与对比	阅读数 565
一、AXI总线与DMA对于ZYNQ，掌握PS与PL的高速接口；掌握几种DMA的区别与用法；能够编写基于AXI-4总线的...	博文 来自： dylll321的博客
zynq DMA分析	阅读数 3534
Zynq Linux pl330 DMAEdit 0 10 ...3 TagsDMAlinuxZynq Notify RSS Backlinks Source Print Export (PDF)IMP...	博文 来自： zhoudengqing的...
ZYNQ AXI DMA	阅读数 1万+
此文是转载自 http://www.fpgadeveloper.com/2014/08/using-the-axi-dma-in-vivado.html我在测试AXI DMA...	博文 来自： weilxuext的博客
MYIR-ZYNQ7000系列-zturn教程(6)：uart_cycle	阅读数 3194
开发板环境：vivado 2017.1，开发板型号xc7z020clg400-1，这个工程主要实现的功能是能在uart上进行数据的回...	博文 来自： taowei1314520的...
MYIR-ZYNQ7000系列-zturn教程(2)：Hello_World	阅读数 3875
开发板环境：vivado 2017.1，开发板型号xc7z020clg400-1（工程末尾提供了工程源代码大家可以去网盘下载）ste...	博文 来自： taowei1314520的...
和菜鸟一起学linux总线驱动之DMA传输	阅读数 281
最早接触DMA的时候是大三的微机原理，当时不是很理解，什么DMA模式啊，只知道是传输速度快，不经过CPU，...	博文 来自： weixin_30765505...
MYIR-ZYNQ7000系列-zturn教程(10)：debug调试	阅读数 1574
开发板环境：vivado 2017.1，开发板型号xc7z020clg400-1，这个工程主要是用hello_world这个工程进行debug...	博文 来自： taowei1314520的...
ZYNQ基础系列（六）DMA基本用法	阅读数 1万+
DMA环路测试涉及到高速数据传输时，DMA就显得非常重要了，本文的DMA主要是对PL侧的AXIDMA核进行介绍...	博文 来自： long_fly的博客
MYIR-ZYNQ7000系列-zturn教程(14)：在PL中使用ILA进行调试	阅读数 974
开发板环境：vivado 2017.1，开发板型号xc7z020clg400-1，这个工程主要是用ILA观测FPGA输出管脚的波形链接...	博文 来自： taowei1314520的...
MYIR-ZYNQ7000系列-zturn教程(17)：用axi_uart发送数据	阅读数 4161
开发板环境：vivado 2017.1，开发板型号xc7z020clg400-1，这个工程主要用axi_uart发送数据，IP核设置的波特...	博文 来自： taowei1314520的...
第十章 ZYNQ-MIZ701 DDR3 PS读写操作方案	阅读数 53
本编文章的目的主要用简明的方法在纯PS里对DDR3进行读写。 本文所使用的开发板是Miz701 PC 开发环境版本：V...	博文 来自： weixin_30243533...
<div><div></div><div>刘小狼 13篇文章 <div>关注</div>排名:千里之外</div></div>	
<div><div></div><div>骆驼日记 331篇文章 <div>关注</div>排名:千里之外</div></div>	
<div><div></div><div>weilxuext 3篇文章 <div>关注</div>排名:千里之外</div></div>	
FPGA片上PS在SDK编译环境下调用DMA	阅读数 469
背景：我们之前通过linux编译模式下调用DMA，testBench中运用的指令为fd = open("/dev/axi-dma1", O_RDWR...	博文 来自： 邢翔瑞的技术博客
Zynq PS DMA控制器应用笔记	阅读数 8410
Zynq PS DMA应用笔记Hello,PandaZynq-7000系列器件PS端的DMA控制器采用ARM的IP核DMA-330（PL-330）...	博文 来自： haoxingheng的专栏
MYIR-ZYNQ7000系列-zturn教程(9)：将bit文件固化到QSPI_Flash	阅读数 4060
开发板环境：vivado 2017.1，开发板型号xc7z020clg400-1。我们用FPGA最后生成的是二进制bit文件，bit文件下...	博文 来自： taowei1314520的...
Zynq PS_PL间通信学习（一）AXI_DMA_LOOP测试	阅读数 5011
参考资料：Xilinx官方参考文档：PG021_axi_dma、UG585_zynq_7000_TRM等AXIDMA开发http://www.fpgadev...	博文 来自： weisili2000_2000...
Xilinx的Zynq系列，ARM和PL通过DMA通信时如何保证DDR数据的正确性。	阅读数 547
使用ZYNQ或者MPSoC的好处是可以通过PL逻辑设计硬件加速器，对功能进行硬件加速。加速器和ARM之间的交互...	博文 来自： 拾贝壳的大男孩

ZYNQ7000 pl330DMA vs CPU读DDR速率分析 阅读数 158

ZYNQ7000 pl330DMA vs CPU读DDR速率分析；解决DMA在AMP环境下无法正常运行的问题；得出core0对内存... 博文 来自： Doriswang84的博客

MYIR-ZYNQ7000系列-zturn教程(8)-PS给PL时钟点亮LED 阅读数 3614

开发板环境：vivado 2017.1，开发板型号xc7z020clg400-1，这里用的这个工程是“从新建工程到下载bit”这个工... 博文 来自： taowei1314520的...

dma 测试例子 阅读数 245

#include <linux/module.h>#include <linux/slab.h>#include <linux/s... 博文 来自： weixin_34174322...

MYIR-ZYNQ7000系列-zturn教程(1)-从新建工程到下载bit文件 阅读数 1417

开发板环境：vivado 2017.1，开发板型号xc7z020clg400-1（工程末尾提供了工程源代码大家可以去网盘下载）St... 博文 来自： taowei1314520的...

Zynq-linux PL与PS通过DMA数据交互 阅读数 1076

一、目标在米尔科技的z-turn板上，采用AXI DMA 实现zynq的PS与PL数据交互。二、分析①PS数据传PL驱动中的... 博文 来自： 天使之猜的博客

MYIR-ZYNQ7000系列-zturn教程(24)：用vdma搭建hdmi显示通路 阅读数 920

开发板环境：vivado 2017.4，开发板型号xc7z010clg400-1，这个工程主要用vdma核搭建hdmi显示工程链接：ht... 博文 来自： taowei1314520的...

ZYNQ DMA 回环 32位数据不完整的问题 阅读数 70

这里写自定义目录标题欢迎使用Markdown编辑器新的改变功能快捷键合理的创建标题，有助于目录的生成如何改变... 博文 来自： qq_39214002的博客

ZYNQ7000 #4 - Linux环境下使用AXI-DMA读取PL外接ADC 阅读数 1389

该篇文章是上一篇博客（https://blog.csdn.net/sements/article/details/90230188）的实际应用版本。在上篇中... 博文 来自： 里先森

【ZYNQ-7000开发之四】在PS端使用AXI DMA传输的步骤 阅读数 1万+

本篇文章简要总结下AXI DMA在ZYNQ PS端的初始化方法。本文摘抄自xilinx SDK的API文档，更加详细的内容请参... 博文 来自： RZJMPB的博客

CC2642研究之控制SPI Nand FLASH和EEPROM 阅读数 672

接上篇博客，测试蓝牙例程通过后，接下来实现利用CC2642控制板上的SPI接口NAND Flash和EEPROM。软件：si... 博文 来自： yyw_0429的博客

FPGA设计——CMOS摄像与HDMI显示(MIPI版) 阅读数 177

1. 概述本设计采用FPGA技术，将CMOS摄像头(MIPI接口)的视频数据经过采集、存储、帧率转换及格式转换，最终... 博文 来自： weixin_34413065...

ZYNQ7000 #3 - Linux环境下在用户空间使用AXI-DMA进行传输 阅读数 1693

本文使用Petalinux搭建相关linux环境，在vivado中搭建了一个简单的PS->AXI-DMA->AXI-FIFO->AXI-DMA->PS... 博文 来自： 里先森

ZYNQ AXI DMA调试细节 阅读数 4011

本文介绍ZYNQAXIDMA的简单模式使用方法，查询模式（poll），不使用中断，32bit。1.有关DMA的函数调用， ... 博文 来自： q774318039a的博客

ZYNQ 7020 PL以AXI_DMA访问DDR或OCM 阅读数 3073

本章主要介绍ZYNQ 7020的PL端在PS的控制下实现对DDR的访问，通过debug的方式抓取DDR S_AXI_HP接口的... 博文 来自： mexican007的专栏

AXI DMA数据对齐 阅读数 4398

查看PG-021知道，AXI DMA采用小端对齐，即高字节放在高地址，低字节放在低地址。在做AD7989数据采集时， ... 博文 来自： yang2011079080...

ZYNQ7000 #SP1 - Linux emmc boot with AXI-DMA loop test 阅读数 231

本篇是综合工程，将较为详细的讲解如何利用vivado搭建一个AXI-DMA环通测试环境，并使用petalinux进行linux系... 博文 来自： 里先森

ZYNQ跑系统 系列（四） AXI-DMA的linux下运行 阅读数 1万+

AXI-DMA的linux驱动一、搭建硬件环境vivado版本2017.4，芯片为7010，不过不管什么版本和芯片大致步骤是一... 博文 来自： long_fly的博客

使用Z702构建摄像头+HLS图像处理模块+HDMI显示数据流工程 阅读数 584

首先拜大神http://blog.csdn.net/rzjmpb/article/details/50212875然后记录一下最近的心路：1、构建上述数据流... 博文 来自： beny270的博客

zynq-7000学习笔记(四)——Zedboard HDMI核的构建和输出显示测试（2017/6/9补充修改） 阅读数 1万+

参考：1、【ZYNQ-7000开发之三】ZYNQ平台的HDMI驱动测试2、 ADV7511 Xilinx Evaluation Boards Refere... 博文 来自： luotong86的专栏

zynq PS侧DMA驱动 阅读数 1万+

linux中，驱动必然会有驱动对应的设备类型。在linux4.4版本中，其设备是以设备树的形式展现的。PS端设备树的de... 博文 来自： shichaog的专栏

Zynq 7000从零开始之一 -- HelloWord 阅读数 2万+

使用myir的z-turn开发板，做一个从uart打印hello world的实验，只用PS，不用PL部分,程序从SD卡启动,跑在PS的... 博文 来自： 青蛙@嘎嘎

Linux with HDMI video output on the ZED, ZC702 and ZC706 boards 阅读数 2807

https://wiki.analog.com/resources/tools-software/linux-drivers/platforms/zynqOverviewPreparing the S... 博文 来自： hengxe的专栏



举报

zedboard (zynqXC7Z020) 入门实验之PS_GPIO的使用 (MIO)

阅读数 4993

开发环境：WIN7-64bit ； ISE 14.4；Zedboard开发板； zynq里含有双核cortex-a9，那么如何使用arm自带的...

博文 来自： 小墨

Xilinx Vivado zynq7000 入门笔记

阅读数 482

http://www.wodefanwen.com/lhd_3076u8kilp4i6jp0x0cx_1.htmlIP Integrator flow 1. 创建RTL工程2. 创建IP In...

博文 来自： Linux Android Web

python json java mysql pycharm android linux json格式



3

TA的个人主页 >

原创

粉丝

获赞

评论

访问

37

195

59

164

15万+

等级:

博客 4

周排名: 3万+

积分: 1255

总排名: 6万+

勋章:



关注

私信

天山汽車借款，不限車種

讓您的汽機車變為活用的週轉

汽車借款立即可知額度，當日立車齡，不限車種，原車可用，可037593333.tw

開啟

最新文章

quartus II 12.1 使用教程（7） vga显示测试

MYIR-ZYNQ7000系列-zturn教程(27): lwip测试

quartus II 12.1 使用教程（6） ROM 测试

quartus II 12.1 使用教程（5） eeprom 读写测试


quartus II 12.1 使用教程（4） uart 测试


分类专栏


	VIVADO 安装教程	1篇
	quartus II	5篇
	三态门详解	
	quartus II 12.1 使用...	1篇
	ZYNQ7000	27篇


归档


2019年12月	1篇
2019年9月	1篇
2019年8月	5篇
2019年7月	2篇
2019年4月	1篇
2019年3月	2篇






















举报

展开

热门文章

VIVADO 安装教程

阅读数 84216

三态门详解

阅读数 15398

quartus II 12.1 使用教程（1） 怎样调用PLL 核

阅读数 7556

MYIR-ZYNQ7000系列-zturn教程(17)：用axi_uart发送数据

阅读数 4156

MYIR-ZYNQ7000系列-zturn教程(9)：将bit文件固化到QSPI_Flash

阅读数 4055

最新评论

VIVADO 安装教程

rq8866：缺License的小伙伴 链接：https://pan.baidu.com/s/11mjkpyERdUH3q5C_TpfQxQ ...

FT232H如何使用jtag接口

taowei1314520：[reply]qq_42662835[/reply]我是直接对eeprom里写数据进去的，数据我已经 ...

FT232H如何使用jtag接口

taowei1314520：[reply]sssshhhhhhhh[/reply]这个vivado有这个usb驱动也需要安装一下， ...

FT232H如何使用jtag接口

sssshhhhhhhh：你好，插上电脑以后显示 USB Serial Conventor （仅配置了USB和EEPROM这 ...

MYIR-ZYNQ7000系列-z...

kuyunge：SPI一次是通信一个字节码？



QQ客服 kefu@csdn.net

客服论坛 400-660-0108

工作时间 8:30-22:00

关于我们 招聘 广告服务 网站地图

京ICP备19004658号 经营性网站备案信息

公安备案号 11010502030143

©1999-2020 北京创新乐知网络技术有限公司 网络110报警服务

北京互联网违法和不良信息举报中心

中国互联网举报中心 家长监护 版权申诉

👍

🔗

💬

☆

📱

<

>

赏

🔊

举报