

## FMQL15T485 芯片的开发流程 V01

Date	Version	Revision
2024/9/8	V01	Initial release

## 目录

FMQL15T485芯片的开发流程V01.....	1
1 必须要打两个补丁.....	3
2 支持的开发环境 .....	3
3 VIVADO嵌入IP补丁 .....	3
3.1 存放IP补丁 .....	3
3.2 设置IP补丁环境变量.....	3
3.3 VIVADO添加IP补丁 .....	4
3.3.1 添加IP补丁.....	4
3.3.2 生成补丁按钮.....	4
4 更新VIVADO库文件 .....	5
4.1 添加器件库 .....	5
5 安装PROCISE软件 .....	7
5.1 安装PROCISE软件 .....	7
5.2 设置PROCISE环境变量 .....	7
5.3 设置IAR安装路径 .....	7
6 开发PL端工程 .....	8
6.1 建立VIVADO工程.....	8
6.1.1 建立新工程.....	8
6.1.2 更改老工程.....	9
6.2 一键式补丁 .....	11
6.2.1 给工程加IP补丁.....	11
6.2.2 IP Cache设置.....	13
6.2.3 IP综合的要求.....	13
6.2.4 位流bit补丁的设置.....	15
6.2.5 工程编译及位流生成.....	17
6.3 工程下载及调试 .....	18
7 开发PS端工程 .....	19
7.1 构造IAR工程 .....	19
7.1.1 建立Procise工程.....	19
7.1.2 导入Vivado工程.....	20
7.1.3 启动IAR工程.....	21
7.2 工程下载及调试 .....	23
8 联合调试PL+PS .....	25
9 烧写QSPI .....	25

## 1 必须要打两个补丁

我司PL端开发必须要打的两个补丁分别是IP补丁和Bit补丁。

IP补丁的详细情况可以参考我司《IP\_PATCH工具使用说明...》。

Bit补丁的详细情况可以参考我司Procise安装目录下document文档，或者我司下载站中《FMSH补丁全流程操作指南》。

## 2 支持的开发环境

PL端的开发环境：Vivado2018.3版本或者Vivado2022.1版本，7系列器件推荐使用Vivado2018.3版本，9系列器件推荐使用Vivado2022.1版本。

PS端的开发环境：IAR 8.11.2版本或者IAR 9.20.4版本，内核A7器件推荐使用IAR 8.11.2版本，内核A53器件推荐使用IAR 9.20.4版本。

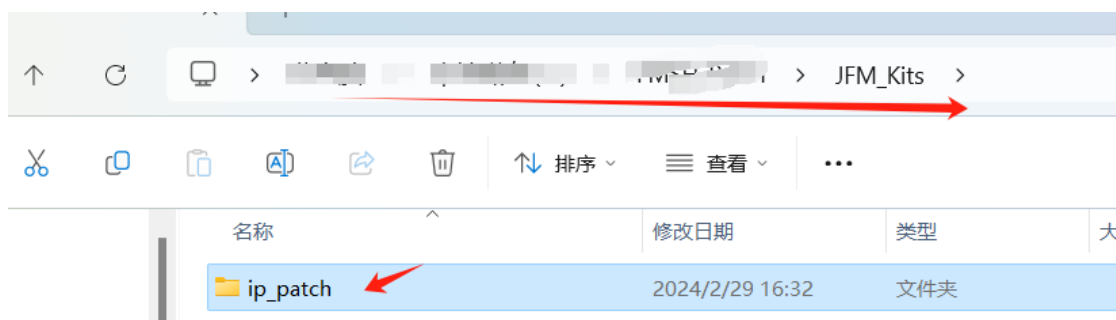
Procise版本：安装20240301之后的版本。

## 3 Vivado嵌入IP补丁

我司IP补丁是嵌入在Vivado开发环境中运行的。

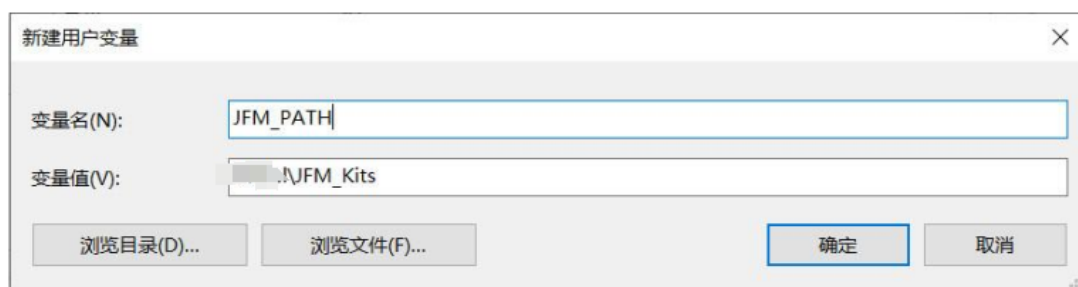
### 3.1 存放IP补丁

从复旦微公司获取IP补丁压缩文件JFM\_Kits.rar文件，FMQL15T485芯片，必须要使用5.3.0以上的IP补丁。选择需要存放的位置，解压缩JFM\_Kits.rar，得到如下图所示，IP补丁存放完成。



### 3.2 设置IP补丁环境变量

新建用户变量，建立变量名为JFM\_PATH，变量值为刚刚IP补丁存放的位置。环境变量后，若已开启Vivado，需要关闭重启Vivado，否则环境变量设置无法生效。此操作，单台电脑只需要操作一次就够了。

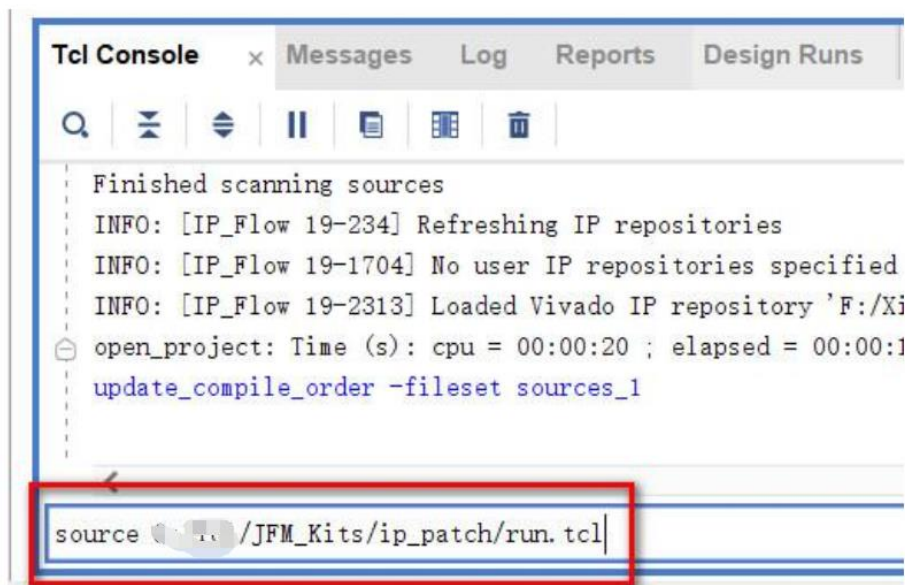


### 3.3 Vivado添加IP补丁

#### 3.3.1 添加IP补丁

在环境变量设置完成后，关闭重启Vivado。在Vivado开发软件的Tcl窗口，输入一下指令source 到run.tcl文件，run.tcl路径就是IP补丁存放的位置，若下图所示。

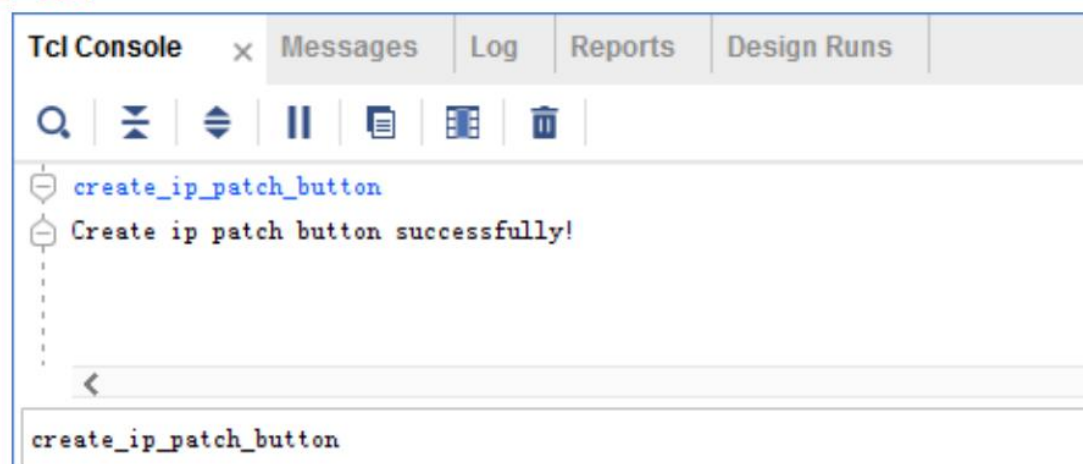
source F:/JFM\_Kits/ip\_patch/run.tcl



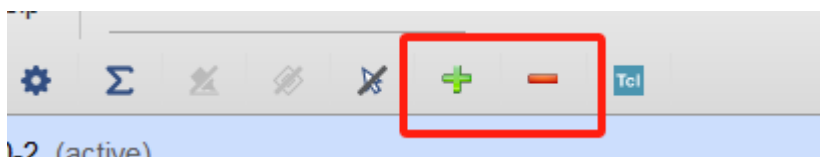
此时补丁已添加到Vivado开发环境中。此操作，单台电脑只需要操作一次就够了。

#### 3.3.2 生成补丁按钮

Source ...run.tcl后，IP补丁相当于已经添加Vivado开发环境中了，为了简化后续任何工程添加补丁的工作量，在Vivado开发软件的Tcl窗口，输入指令create\_ip\_patch\_button，若下图所示，这样就生成了添加IP补丁和删除IP补丁的两个按钮。



结果如下图所示，“+”表示本工程添加IP补丁，“-”表示本工程删除IP补丁。



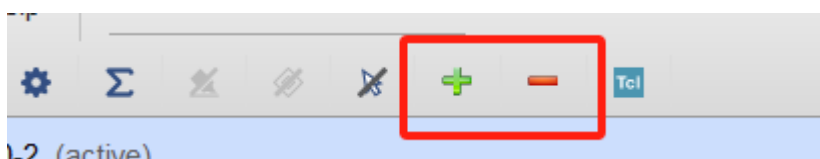
以上步骤，一台电脑或者一个Vivado开发软件只要做一次就够了，跟开发工程没有关系。此操作，单台电脑只需要操作一次就够了。

## 4 更新Vivado库文件

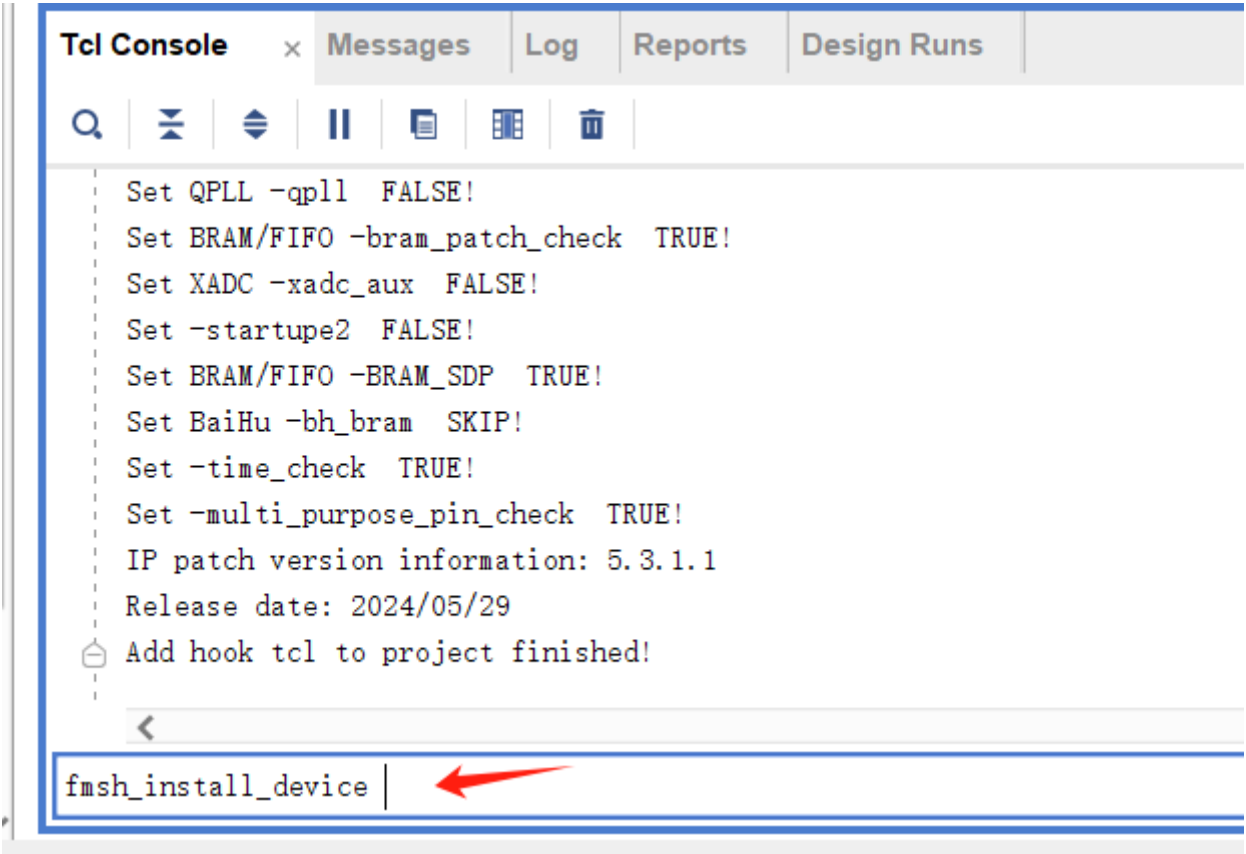
这个更新Vivado库文件包括更新Vivado中的器件库和编译库，在7系列中，替换库文件，只适用对标器件的GT是GTP的芯片，如XC7A50芯片、XC7Z015等，或者我司带有AI功能的芯片，其他芯片不需要更新Vivado库文件。

### 4.1 添加器件库

打开Vivado，打开已有的xc7z015clg485-2工程，或者新建一个xc7z015clg485-2工程，或者仅仅打开hardware Manger窗口。点击“+”添加了IP补丁，目的是为了调用IP补丁中的指令。



而在5.3.1之后的补丁上，在TCL窗口输入fmsht\_install\_device，也可可实现添加FMSH特有的器件及时序库。此操作，单台电脑只需要操作一次就够了。



指令fmsht\_install\_device执行成功后，器件库就会多出11个可选器件，如下图所示。此操作，单台电脑只需要操作一次就够了。

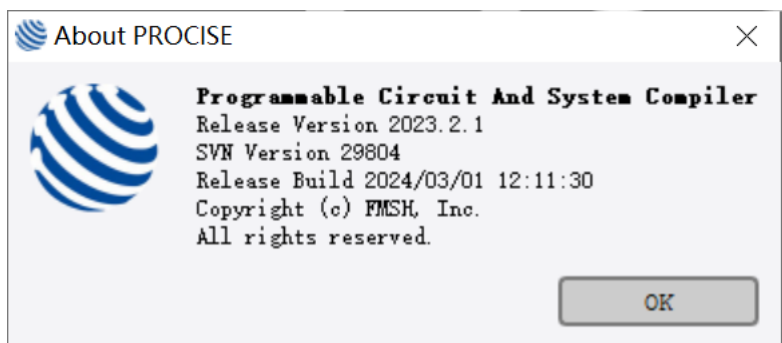
Search: Q: fm		(11 matches)								
Part	I/O Pin C...	Available I...	LUT Elem...	FlipFl...	Block R...	Ultra R...	D...	Gb Transcei...	GTPE2 Transcei...	GTXE2 T
xc7k325t_fmk230tffg676-3	676	400	188200	407600	445	0	840	8	0	8
xc7k325t_fmk230tffg676-2	676	400	188200	407600	445	0	840	8	0	8
xc7k325t_fmk230tffg676-2L	676	400	188200	407600	445	0	840	8	0	8
xc7k325t_fmk230tffg676-1	676	400	188200	407600	445	0	840	8	0	8
xc7k325t_fmk300tffg900-3	900	500	188200	407600	445	0	840	16	0	16
xc7k325t_fmk300tffg900-2	900	500	188200	407600	445	0	840	16	0	16
xc7k325t_fmk300tffg900-2L	900	500	188200	407600	445	0	840	16	0	16
xc7k325t_fmk300tffg900-1	900	500	188200	407600	445	0	840	16	0	16
xc7z045_fmql15clg485-3	485									
xc7z045_fmql15clg485-2	485									
xc7z045_fmql15clg485-1	485									

## 5 安装Procise软件

此Procise可以实施bit补丁。

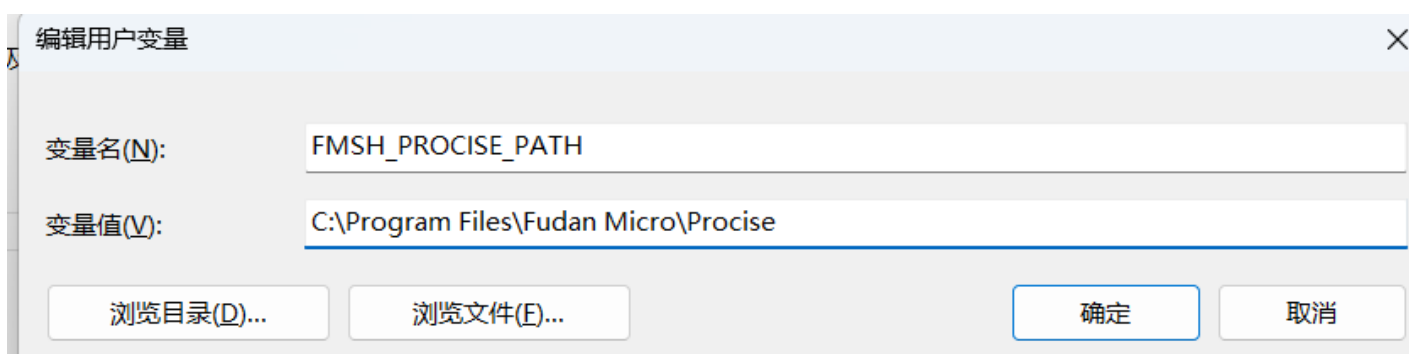
### 5.1 安装Procise软件

安装20240301之后的版本，如下所示。



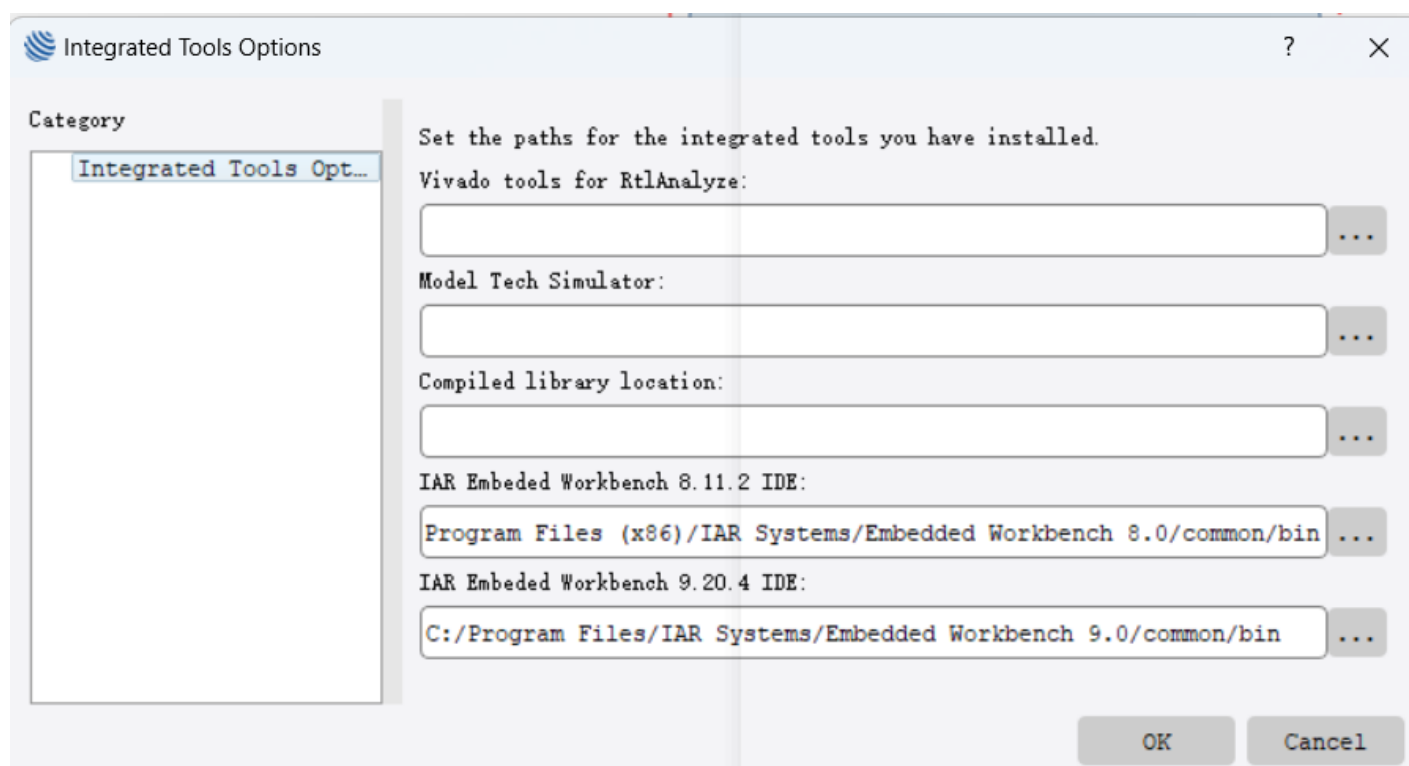
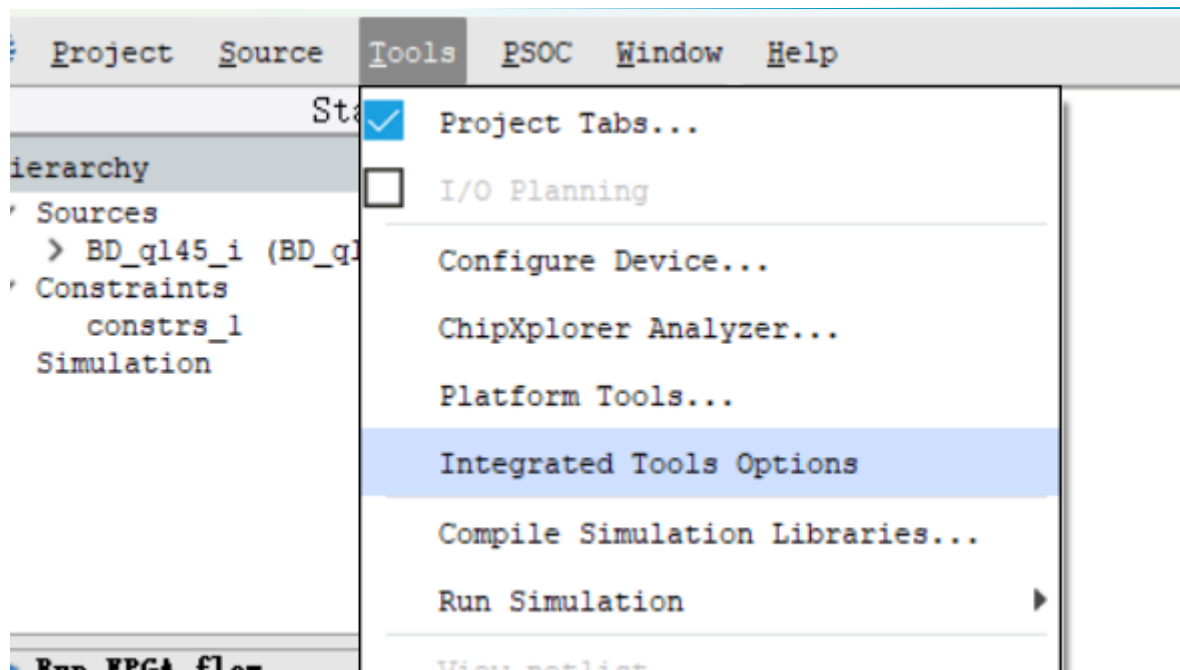
### 5.2 设置Procise环境变量

新建用户变量，建立变量名为FMSH\_PROCISE\_PATH，变量值为刚刚Procise安装位置。环境变量添加完成后，需要重启 VIVADO。这样Vivado在位流生成后会自动调用 Procise对位流打Bit补丁。此操作，单台电脑只需要操作一次就够了。



### 5.3 设置IAR安装路径

在上方菜单选择 Tools->Integrated Tools Options。



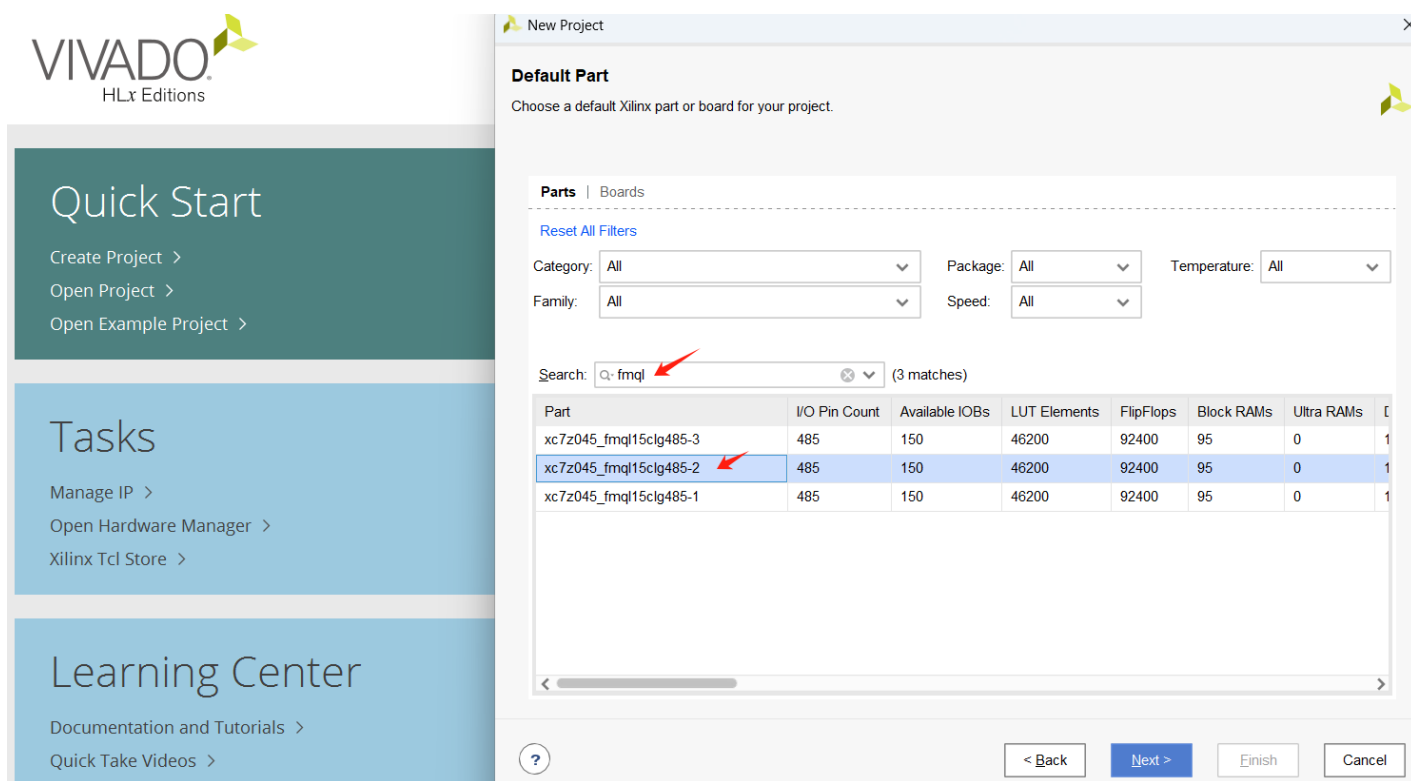
## 6 开发PL端工程

### 6.1 建立Vivado工程

#### 6.1.1 建立新工程



在新建工程中，选择元器件xc7z045\_fmql15clg485-2, 然后跟Xilinx器件一样，添加IP和代码。

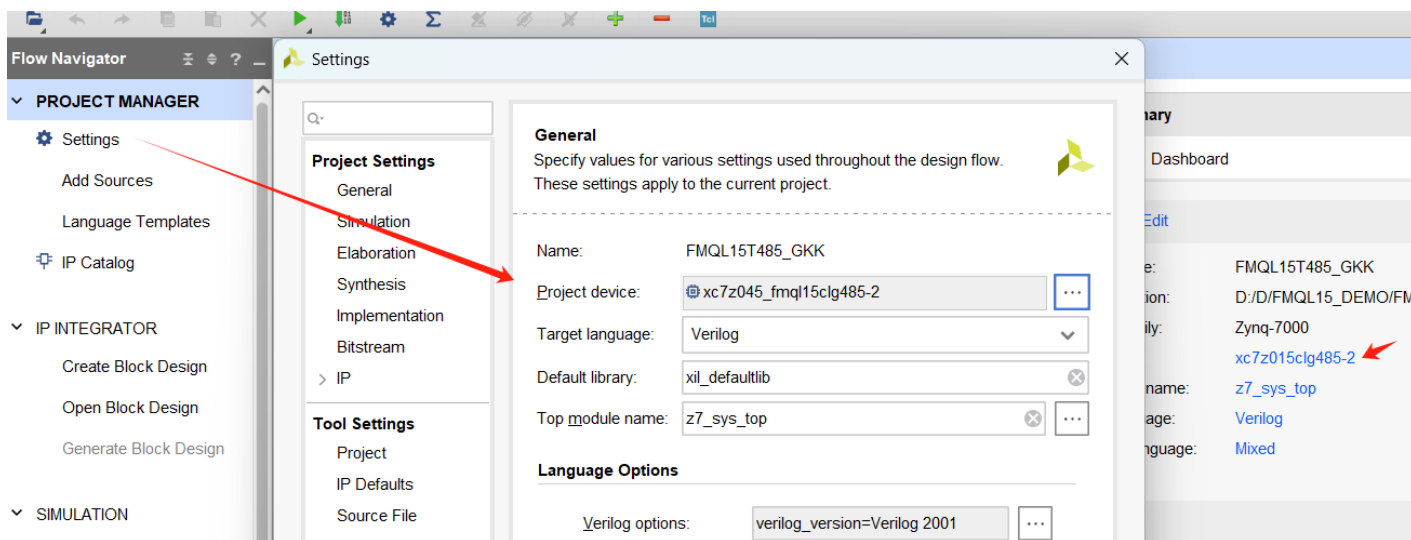
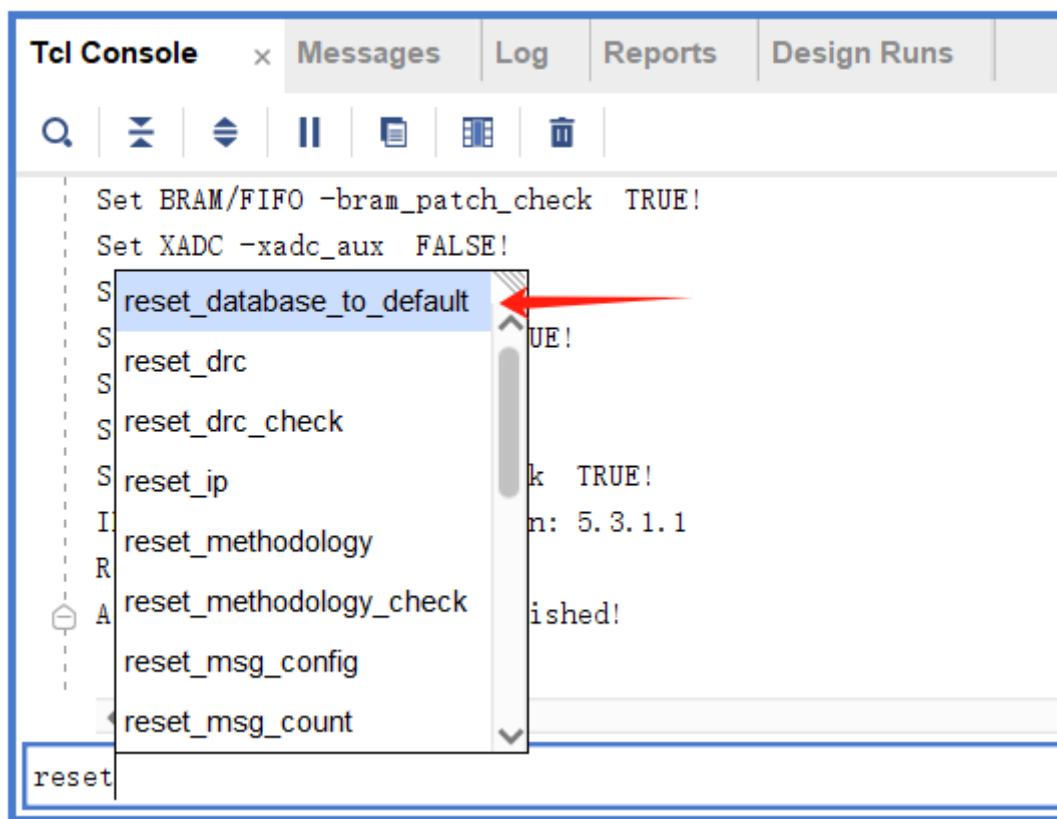


### 6.1.2 更改老工程

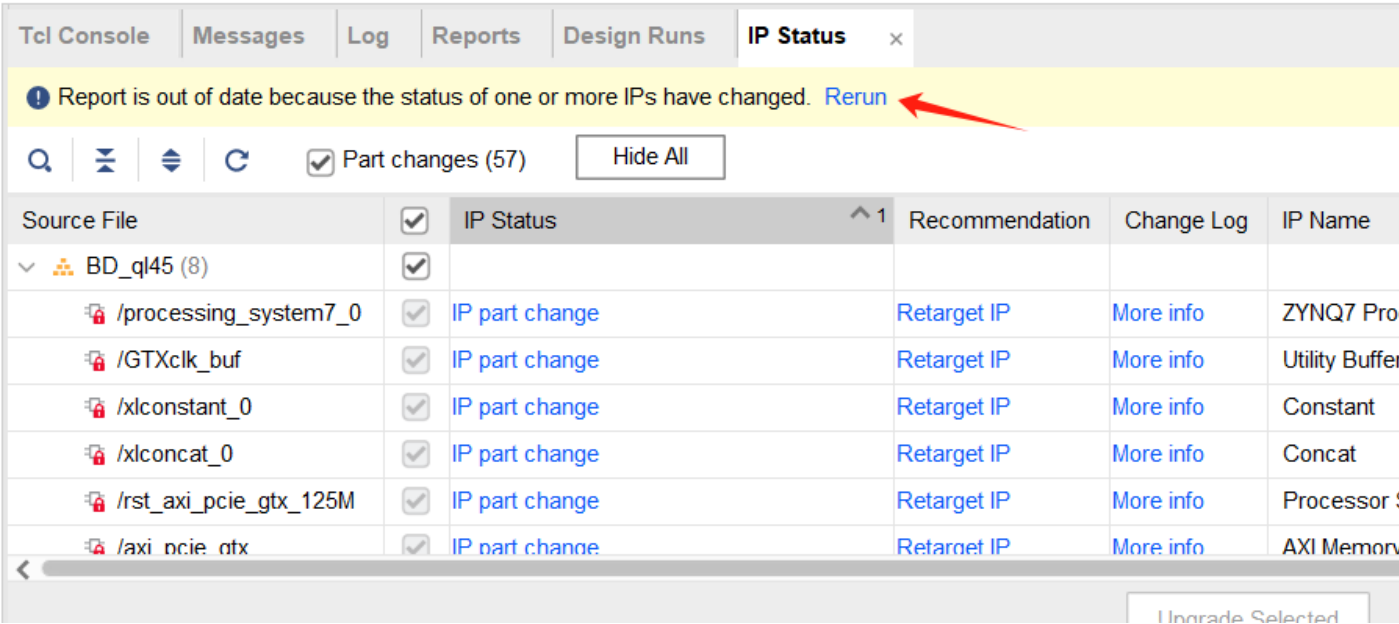
之前老流程：器件选择xc7z015clg485-2，在tcl输入replace\_7z015\_file指令替换编译库。

现有新流程：器件选择xc7z045\_fmql15clg485-2，不需要输入任何指令，开发GT资源也不需要先切到xc7z045上选择GTX了。

在已有老流程工程的情况下，首先执行reset\_database\_to\_default指令，然后器件设置更改为xc7z045\_fmql15clg485-2，即可完成从老流程工程切换到新流程工程，完成工程移植。在IP补丁5.3.0之前，开发JFMQL15T485器件，使用的都是老流程，涉及到GT资源时，首先在xc7z045器件上添加GTX对应的IP, 然后打IP补丁，再切到xc7z015器件上完成综合、实现及生成BIT位流。使用新流程后，再也不用切换器件了，因为新流程能选gtx IP。



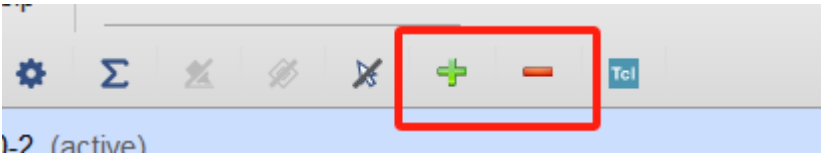
器件更改为xc7z045\_fmql15clg485-2后，涉及的GTX方面的IP需要Upgrade，还需要reset output products和Generate output products。



6.2 一键式补丁

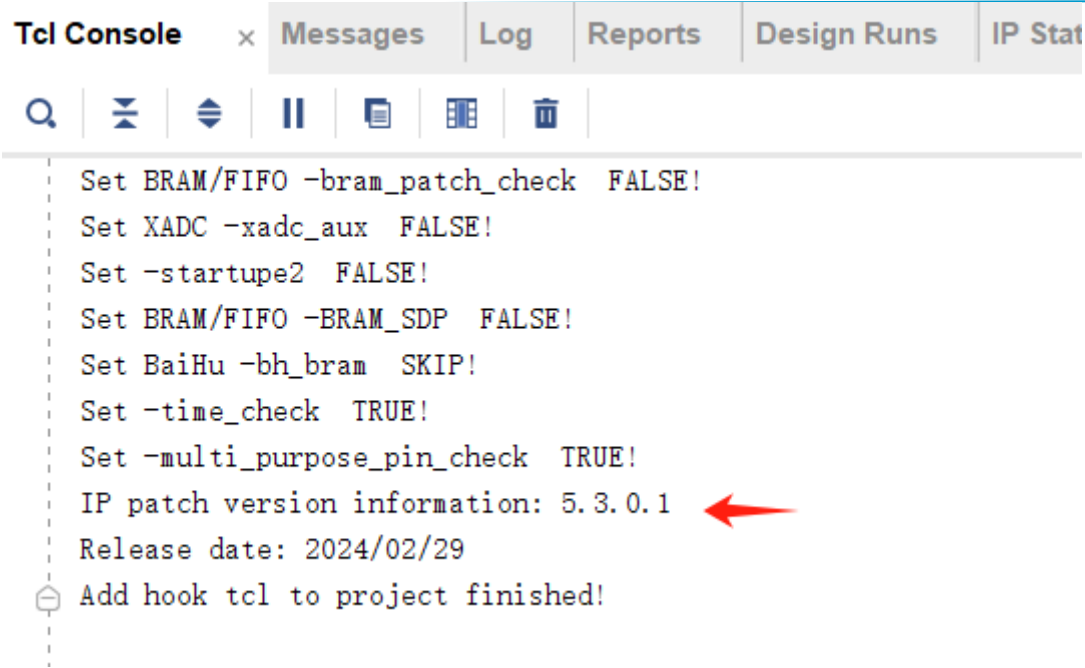
6.2.1 给工程加IP补丁

FPGA工程建立完成后，点击“+”给工程添加了IP补丁，如下图所示。此操作，跟工程有关，因此每个工程都需要操作一次。若本工程不想打补丁了，也可以点击“-”。

















TCL窗口，会弹出如下信息，说明本工程添加了IP补丁。

```
add_hook_tcl_to_prj;
No subcore find in the project!
xfft, fir_compiler, dds_compiler, cpri, jesd, cordic, divgen, floating point IPs files modified successfully! (Total 26 files)
Set -device to 0!
Set PCIE -pcie_version NEW!
Set MIG -mig_version NORMAL!
Set MIG -mig_comp FALSE!
Set MIG -idel_dec_cnt 1!
Set MIG -mig_ecc OFF!
Set MIG -precharge_check FALSE!
Set MIG -max_dc_cnt 100000!
```

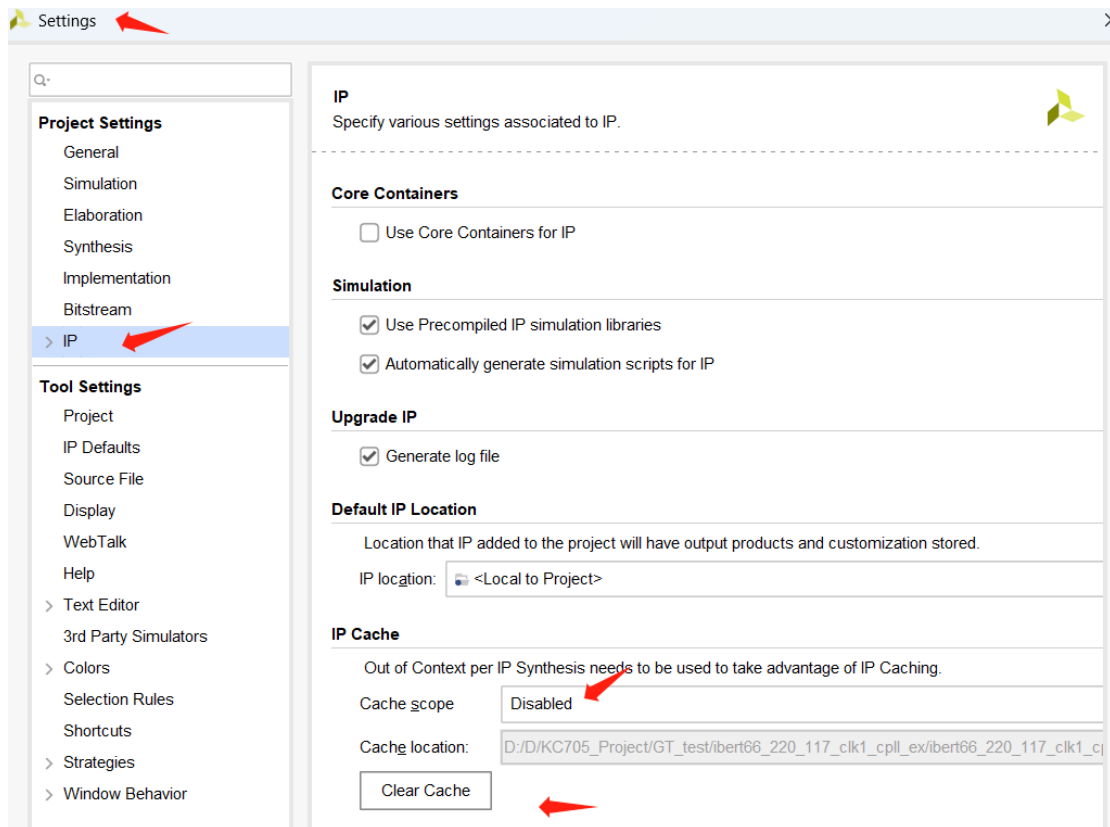


末尾显示本次补丁的版本。点击“+”后，在工程目录下，会多出一个IP Patch文件夹，如下图。

 ip_patch	2024/4/1 13:32	文件夹	
 all_ip_list.log	2024/4/1 14:00	文本文档	1 KB
 call_procise.bat	2024/4/1 14:09	Windows 批处理...	1 KB
 call_procise_disable_icap.bat	2024/4/1 14:09	Windows 批处理...	1 KB
 FMQL15T485_GKK.xpr	2024/4/1 17:16	Vivado Project F...	58 KB
 ip_need_patch_list.cfg	2024/3/21 10:49	CFG 文件	1 KB
 ip_need_patch_list.log	2024/4/1 14:00	文本文档	1 KB
 ip_not_ooc_list.log	2024/4/1 14:00	文本文档	1 KB
 ip_patch.log	2024/4/1 14:00	文本文档	1 KB
 ip_patch_version_info.log	2024/4/1 14:00	文本文档	1 KB
 ip_patched_time_list.log	2024/4/1 14:01	文本文档	4 KB
 ip_upgrade.log	2024/4/1 13:40	文本文档	19 KB
 procise_incr_cfg.txt	2024/3/21 9:50	文本文档	1 KB
 procise_run.tcl	2024/4/1 14:09	TCL 文件	1 KB

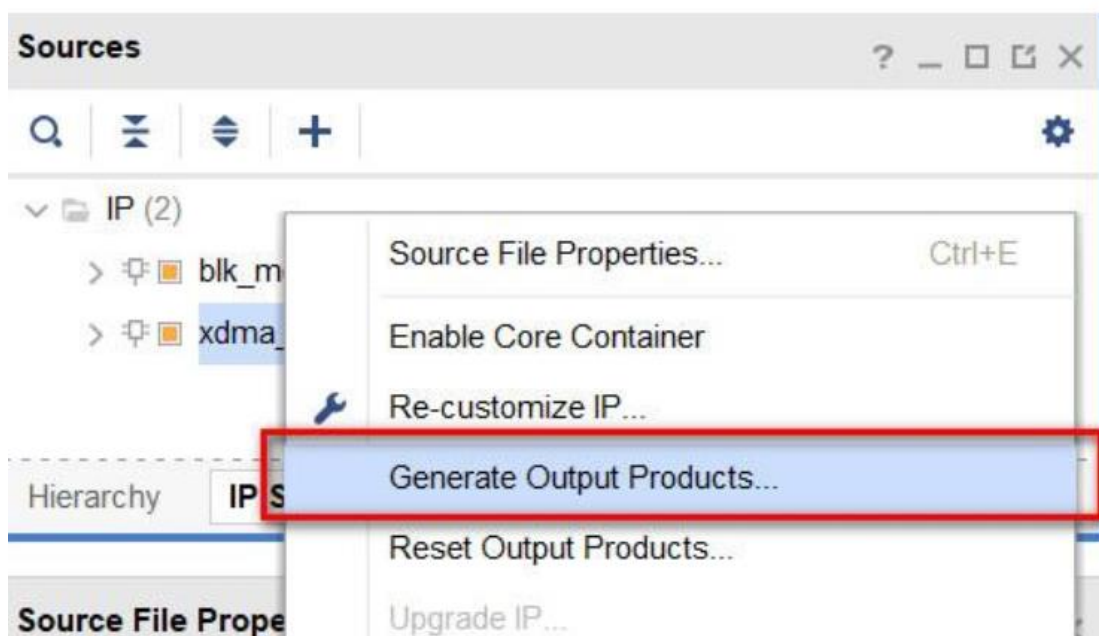
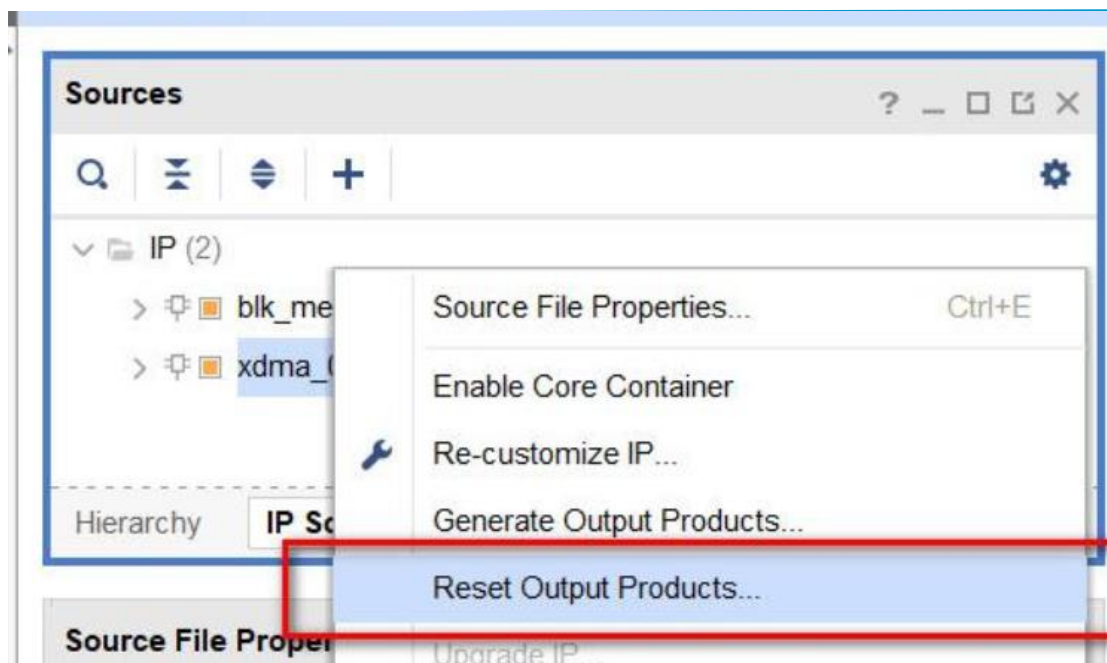
## 6.2.2 IP Cache设置

打IP补丁，必须要将 IP Cache 下的 Cache scope 选项设置为 Disabled。Tools/Settings下，将 IP Cache 下的 Cache scope 选项设置为 Disabled。此操作，跟工程有关，因此每个工程都需要操作一次。



## 6.2.3 IP综合的要求

IP综合必须要是OOC模式，如下图所示。IP 综合选项要选择 “Out of context per IP”。此操作，跟工程的IP有关，因此每个工程的IP都需要操作一次。



### Synthesis Options

☐ Global

☒ Out of context per IP

### Run Settings

Number of jobs: 4

?

Apply

Generate

















Cancel

The screenshot shows the 'Design Runs' window in Vivado. The table displays the following information:

Name	Constraints	Status
synth_1 (active)	constrs_1	Not started
impl_1	constrs_1	Not started
Out-of-Context Module Runs		
blk_mem_gen_1_synth_1	blk_mem_gen_1	synth_design Complete!
xdma_0_synth_1	xdma_0	synth_design Complete!
mig_7series_0_synth_1	mig_7series_0	synth_design Complete!

### 6.2.4 位流bit补丁的设置

把工程中IP Patch文件夹中的procise\_incr\_cfg.txt拷贝出来，放到工程目录下。

 ip_patch 	2024/4/1 13:32	文件夹	
 all_ip_list.log	2024/4/1 14:00	文本文档	1 KB
 call_procise.bat	2024/4/1 14:09	Windows 批处理...	1 KB
 call_procise_disable_icap.bat	2024/4/1 14:09	Windows 批处理...	1 KB
 FMQL15T485_GKK.xpr	2024/4/1 17:16	Vivado Project F...	58 KB
 ip_need_patch_list.cfg	2024/3/21 10:49	CFG 文件	1 KB
 ip_need_patch_list.log	2024/4/1 14:00	文本文档	1 KB
 ip_not_ooc_list.log	2024/4/1 14:00	文本文档	1 KB
 ip_patch.log	2024/4/1 14:00	文本文档	1 KB
 ip_patch_version_info.log	2024/4/1 14:00	文本文档	1 KB
 ip_patched_time_list.log	2024/4/1 14:01	文本文档	4 KB
 ip_upgrade.log	2024/4/1 13:40	文本文档	19 KB
 procise_incr_cfg.txt 	2024/3/21 9:50	文本文档	1 KB
 procise_run.tcl	2024/4/1 14:09	TCL 文件	1 KB

打开procise\_incr\_cfg.txt，更改里面的器件，根据实际顶层文件的名称更改位流的名称，因为QL15器件不需要再打GT补丁，因此-gt\_cfg serdes\_parameter.cfg前面必须有##，同时若需要添加其他功能补丁，删除指令前的##即可，具体如下所示。



```
##### device #####
device_type FMQL15T485

##### input and output bitstream name #####
##input_bit z7_sys_wrapper.bit
##output_bit z7_sys_wrapper_incr_cfg.bit

##input_bit z7_sys_top.bit
##output_bit z7_sys_top_incr_cfg.bit

input_bit FMQL15T485_GKK_top.bit
output_bit FMQL15T485_GKK_top_incr_cfg.bit

##### increment configuration options #####
## -extra_dummy 10

## -gt_cfg serdes_parameter.cfg
## -disable_abort
## -disable_icap
```

### 6.2.5 工程编译及位流生成

在完成5.2.4后，工程编译和位流生成就跟开发进口器件是一样的。综合后，会出现很多有关IP Patch相关的文件，通过这些文件可以知道哪些IP打了补丁，补丁版本等，同时Generate bitstream会产生3个bit，由此可知，Vivado后台调用了Procise打bit补丁，如下图所示。

ip_patch	2024/4/1 13:32	文件夹	
all_ip_list.log	2024/4/1 14:00	文本文档	1 KB
call_procise.bat	2024/4/1 14:09	Windows 批处理...	1 KB
call_procise_disable_icap.bat	2024/4/1 14:09	Windows 批处理...	1 KB
FMQL15T485_GKK.xpr	2024/4/1 17:16	Vivado Project F...	58 KB
ip_need_patch_list.cfg	2024/3/21 10:49	CFG 文件	1 KB
ip_need_patch_list.log	2024/4/1 14:00	文本文档	1 KB
ip_not_ooc_list.log	2024/4/1 14:00	文本文档	1 KB
ip_patch.log	2024/4/1 14:00	文本文档	1 KB
ip_patch_version_info.log	2024/4/1 14:00	文本文档	1 KB
ip_patched_time_list.log	2024/4/1 14:01	文本文档	4 KB
ip_upgrade.log	2024/4/1 13:40	文本文档	19 KB
procise_incr_cfg.txt	2024/3/21 9:50	文本文档	1 KB

z7_sys_top.bit	z7_sys_top_incr_cfg.bit
z7_sys_top.hwdef	z7_sys_top_incr_cfg_disable_icap.bit
z7_sys_top.ltx	z7_sys_top_io_placed.rpt

6.3 工程下载及调试

在Vivado环境下, 可以使用Xilinx的JTAG仿真器下载和调试bit文件。在Procise环境下, 可以使用Xilinx的JTAG仿真器或者J-link仿真器下载bit, 若有ltx调试文件, 建议还是使用Vivado+Xilinx的JTAG仿真器来调试。

在Vivado环境下, 只支持最长的bit下载, 其余bit无法下载, 可以使用我司Procise下载, 现象如下所示。

```

refresh_hw_device [lindex [get_hw_devices xc7z015_0] 0]
INFO: [Labtools 27-1435] Device xc7z015 (JTAG device index = 0) is not programmed (DONE status = 0).
set_property PROBES.FILE {D:/FMQL15_DEMO/FMQL15T485_PCIE_slotref_fmsh/FMQL15T485_GKK.runs/impl_2/z7_sys_top.ltx} [get_hw_devices xc7z015_0]
set_property FULL_PROBES.FILE {D:/FMQL15_DEMO/FMQL15T485_PCIE_slotref_fmsh/FMQL15T485_GKK.runs/impl_2/z7_sys_top.ltx} [get_hw_devices xc7z015_0]
set_property PROGRAM.FILE {D:/FMQL15_DEMO/FMQL15T485_PCIE_slotref_fmsh/FMQL15T485_GKK.runs/impl_2/z7_sys_top.bit} [get_hw_devices xc7z015_0]
program_hw_devices [get_hw_devices xc7z015_0]
ERROR: [Labtools 27-3303] Incorrect bitstream assigned to device. Bitfile is incompatible for this device.
ERROR: [Labtools 27-3165] End of startup status: LOW
ERROR: [Common 17-39] 'program_hw_devices' failed due to earlier errors.
set_property PROBES.FILE {D:/FMQL15_DEMO/FMQL15T485_PCIE_slotref_fmsh/FMQL15T485_GKK.runs/impl_2/z7_sys_top.ltx} [get_hw_devices xc7z015_0]
set_property FULL_PROBES.FILE {D:/FMQL15_DEMO/FMQL15T485_PCIE_slotref_fmsh/FMQL15T485_GKK.runs/impl_2/z7_sys_top.ltx} [get_hw_devices xc7z015_0]
set_property PROGRAM.FILE {D:/FMQL15_DEMO/FMQL15T485_PCIE_slotref_fmsh/FMQL15T485_GKK.runs/impl_2/z7_sys_top_incr_cfg.bit} [get_hw_devices xc7z015_0]
program_hw_devices [get_hw_devices xc7z015_0]
ERROR: [Labtools 27-3303] Incorrect bitstream assigned to device. Bitfile is incompatible for this device.
ERROR: [Labtools 27-3165] End of startup status: LOW
ERROR: [Common 17-39] 'program_hw_devices' failed due to earlier errors.
refresh_hw_device [lindex [get_hw_devices xc7z015_0] 0]
INFO: [Labtools 27-1435] Device xc7z015 (JTAG device index = 0) is not programmed (DONE status = 0).
set_property PROBES.FILE {D:/FMQL15_DEMO/FMQL15T485_PCIE_slotref_fmsh/FMQL15T485_GKK.runs/impl_2/z7_sys_top.ltx} [get_hw_devices xc7z015_0]
set_property FULL_PROBES.FILE {D:/FMQL15_DEMO/FMQL15T485_PCIE_slotref_fmsh/FMQL15T485_GKK.runs/impl_2/z7_sys_top.ltx} [get_hw_devices xc7z015_0]
set_property PROGRAM.FILE {D:/FMQL15_DEMO/FMQL15T485_PCIE_slotref_fmsh/FMQL15T485_GKK.runs/impl_2/z7_sys_top_incr_cfg_disable_icap.bit} [get_hw_devices xc7z015_0]
program_hw_devices [get_hw_devices xc7z015_0]
INFO: [Labtools 27-3164] End of startup status: HIGH
program_hw_devices: Time (s): cpu = 00:00:07 ; elapsed = 00:00:08 . Memory (MB): peak = 1855.281 ; gain = 0.000
refresh_hw_device [lindex [get_hw_devices xc7z015_0] 0]
INFO: [Labtools 27-2302] Device xc7z015 (JTAG device index = 0) is programmed with a design that has 1 ILA core(s).
display_hw_ila_data [get_hw_ila_data hw_ila_data_1 -of-objects [get_hw_ilas -of-objects [get_hw_devices xc7z015_0] -filter {CELL_NAME=~"ila_inst"}]]

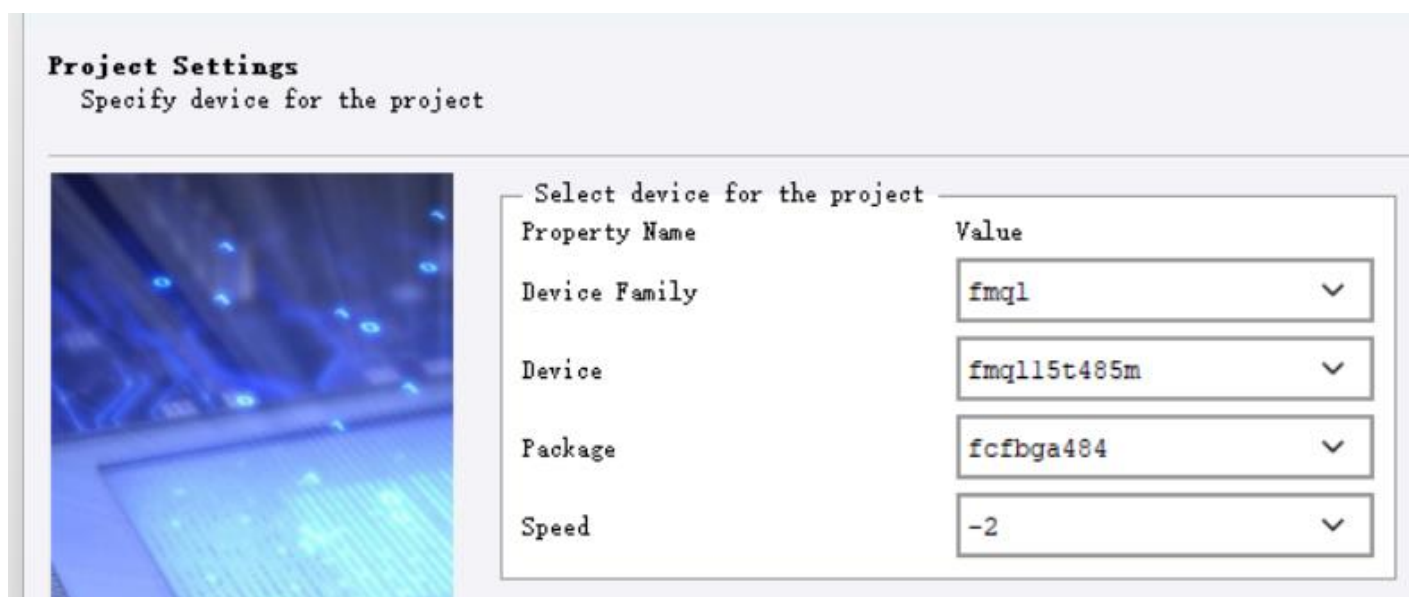
```

## 7 开发PS端工程

### 7.1 构造IAR工程

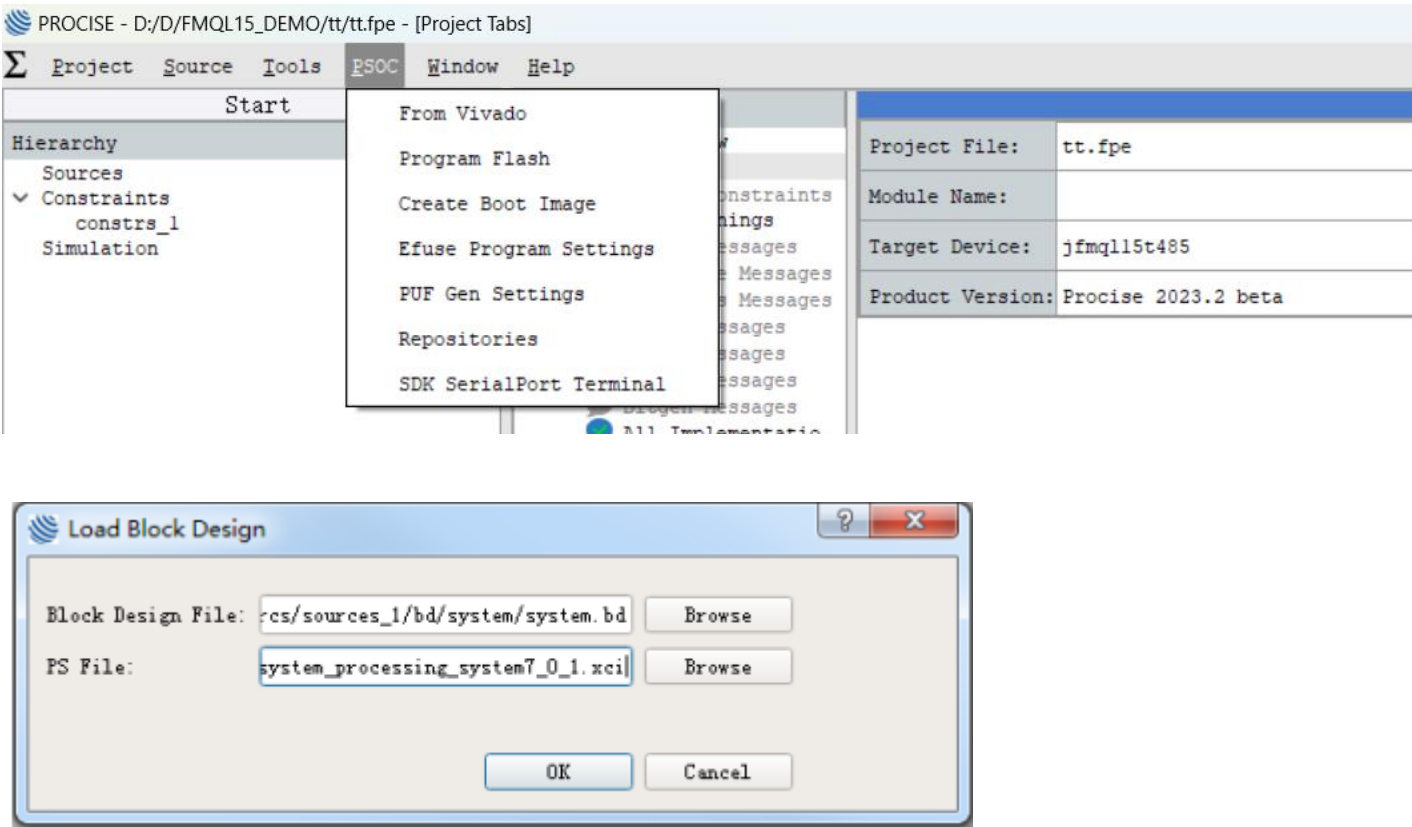
#### 7.1.1 建立Procise工程

新建一个空的 Procise 工程，器件选择 fmql115t485，其余点击NEXT即可，如下图所示。



7.1.2 导入Vivado工程

在上方菜单选择 PSOC->From Vivado。

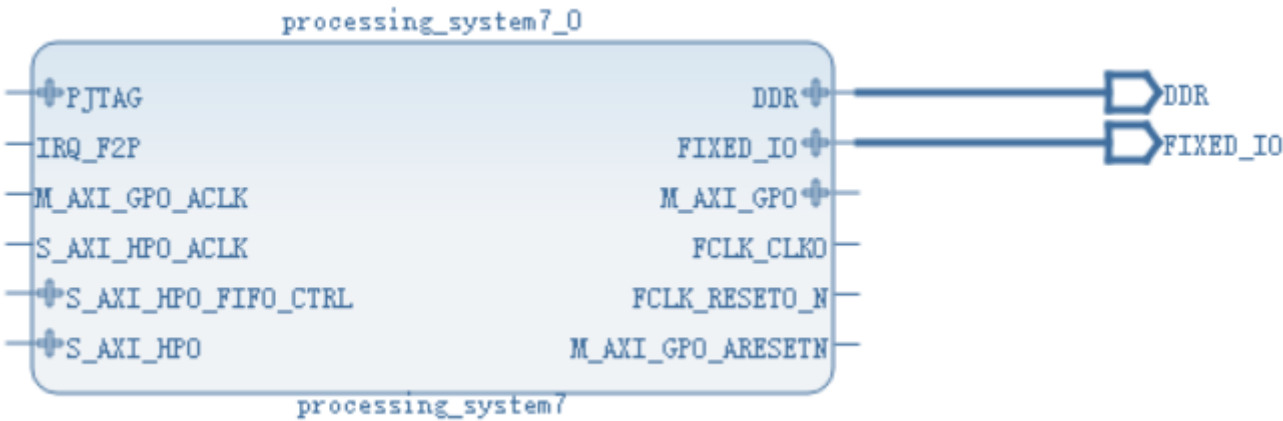


加入 Vivado 生成的 system.bd 文件与 PS 的 IP 配置（.xci）文件  
其中，Vivado 的Block Design 文件(.bd)的路径一般是：

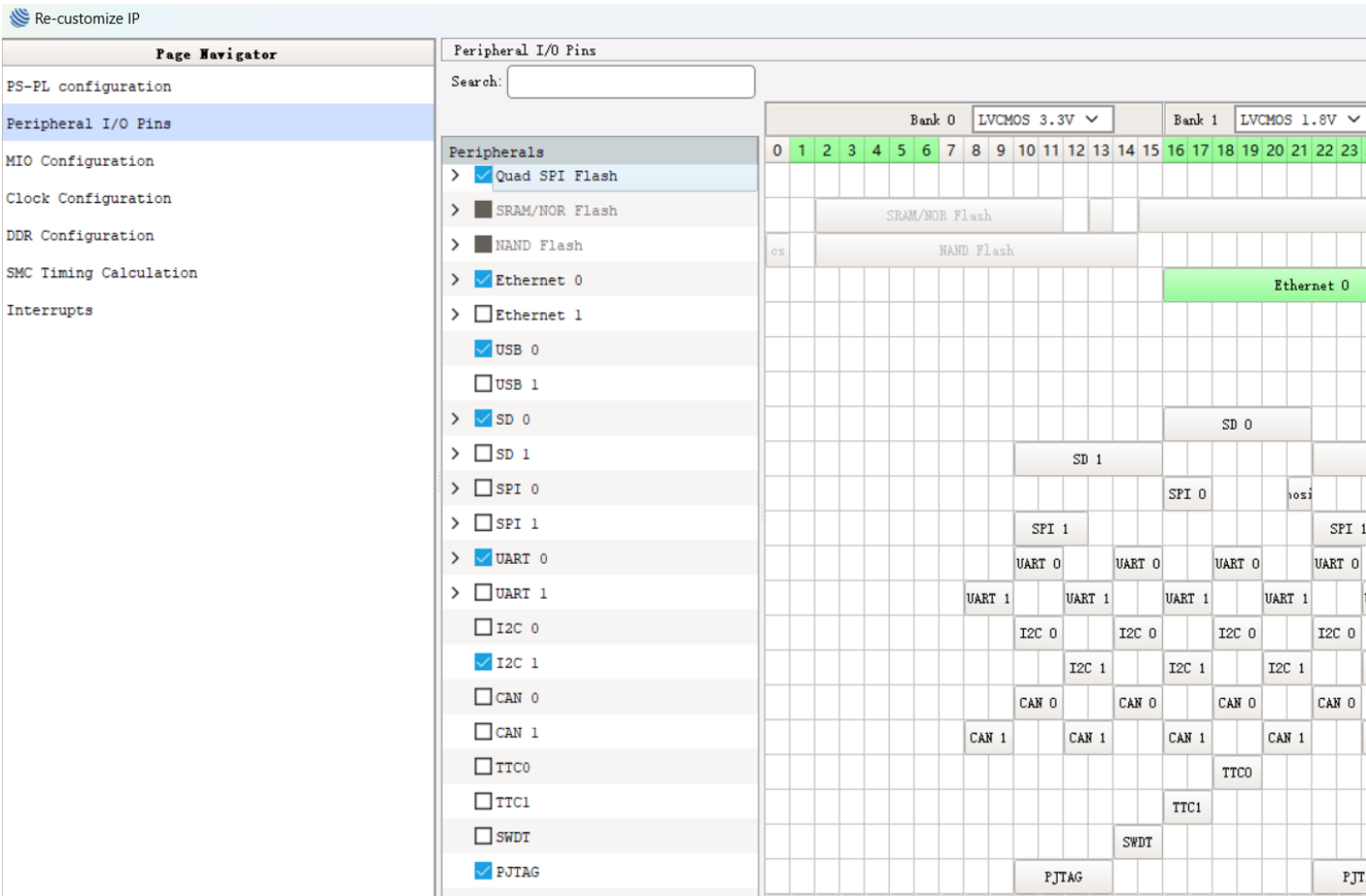
{prj\_name}\{prj\_name}.srcs\sources\_1\bd\{bd\_name}\ {bd\_name}.bd

Vivado PS 的配置文件(.xci) 的路径一般是：

{prj\_name}\{prj\_name}.srcs\sources\_1\bd\{bd\_name}\ip\{bd\_name}\_processing\_system7\_0\_0\{bd\_name}\_processing\_system7\_0\_0.xci



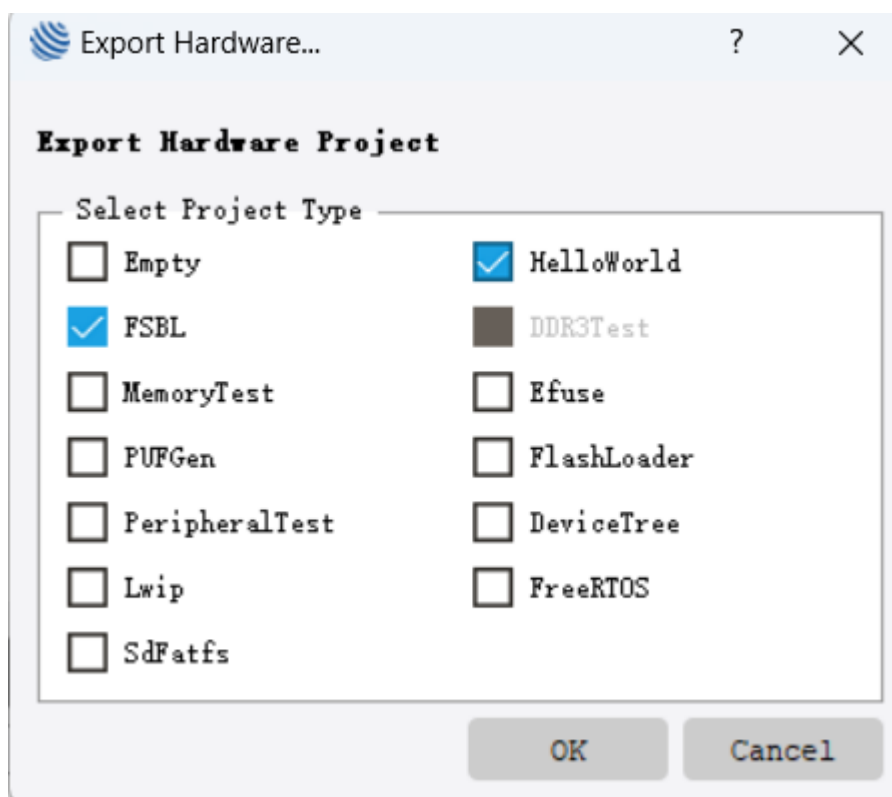
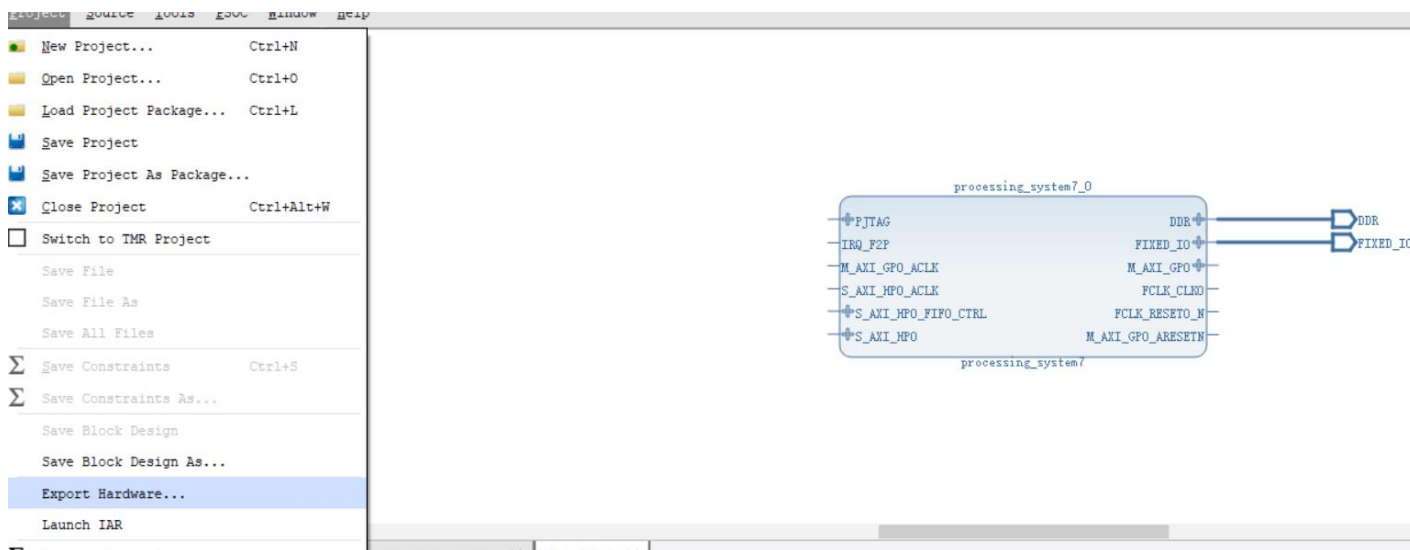
双击PS核，在VVD配置PS的基础上可以做相应地更改。

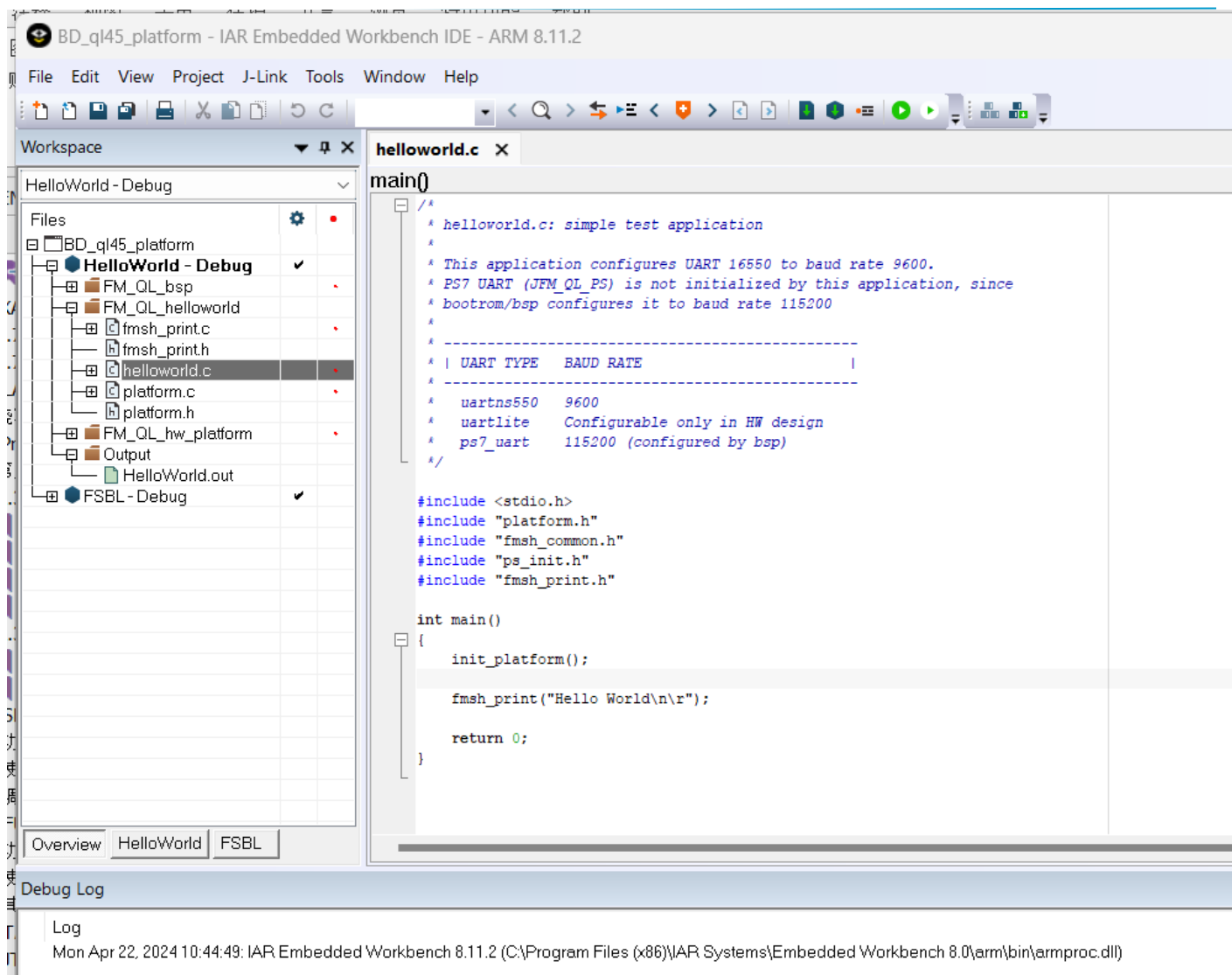


7.1.3 启动IAR工程

执行Export Hardware,弹出第三方开发平台IAR模板工程，选择FSBL和Hellworld，接着执行Launch IAR，

打开IAR开发软件。PS端的裸机开发和FSBL开发必须要在IAR开发环境中进行。





注意，在级联模式下，在访问任何 PL资源之前，需要使能 USER\_LEVEL\_SHIFTER寄存器。

```

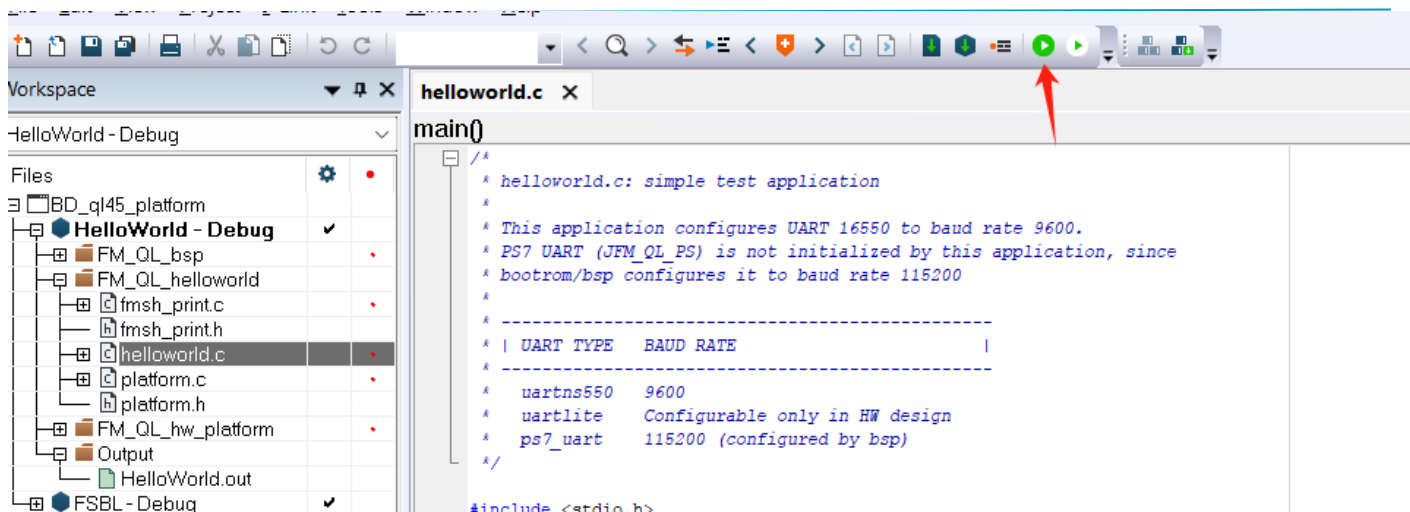
FMSH_WriteReg(FPS_SLCR_BASEADDR, 0x008, 0xDF0D767BU); //unlock
FMSH_WriteReg(FPS_SLCR_BASEADDR, 0x838, 0xf); //Open USER_LVL_SHFTR_EN_A and USER_LVL_SHFTR_EN_5
FMSH_WriteReg(FPS_SLCR_BASEADDR, 0x004, 0xDF0D767BU); //lock

```

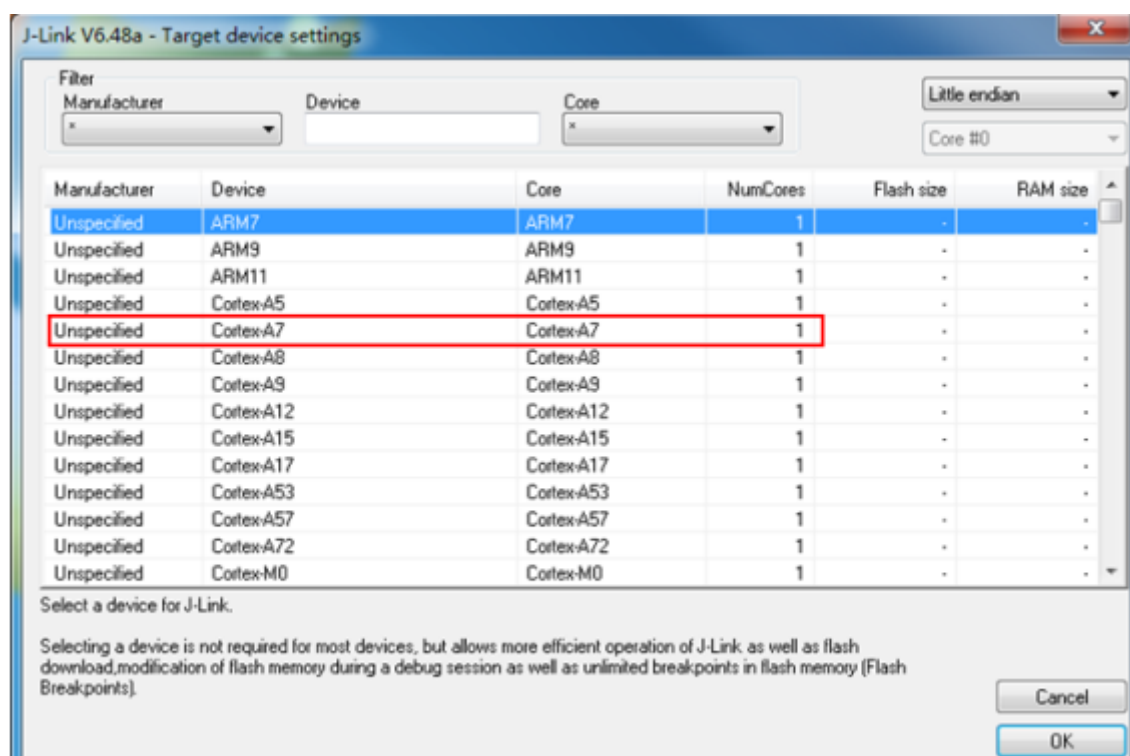
## 7.2 工程下载及调试

若需要调试裸机PS，则必须是要在IAR下进行。在IAR环境下，使用J-link仿真器下载.out文件调试。

在 IAR\_SDK 中，点击上方的绿色小箭头， Download and Debug



弹出提示，点击 OK，接着选择 Cortex-A7，点击 OK。





成功进入 Debug 模式。与其他嵌入式工程一样，可以进行单步调试，设置断点等操作。按小箭头按钮 GO 可以看到流水灯运行。



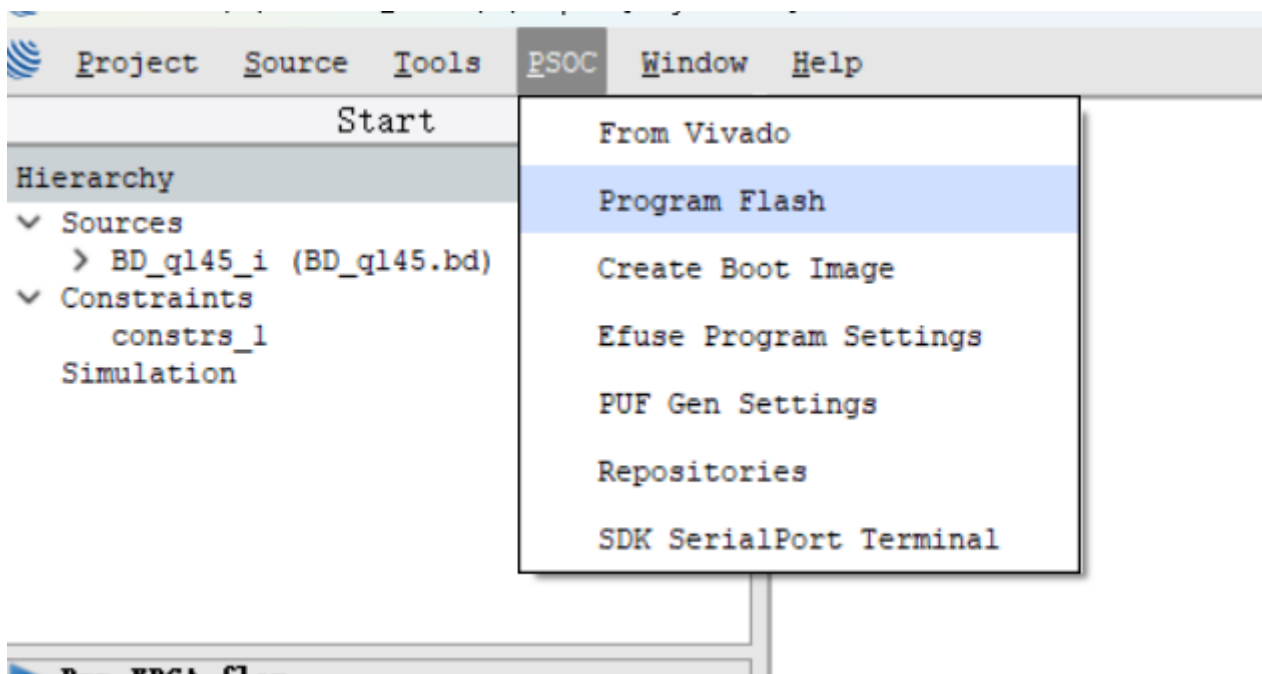
## 8 联合调试PL+PS

单独调试PL端，芯片处于级联模式或者独立模式都是可以的。单独调试PS端，芯片也可以处于级联模式或者独立模式。

若是需要同时调试PL端和PS端裸机APP或者FABL程序，则必须要在独立模式下进行。因为PL端调试需要使用 Vivado+Xilinx的JTAG仿真器来调试，而PS端的裸机APP或者FABL程序需要在IAR+JLINK来调试，因此需要两个独立的JTAG端口。

## 9 烧写QSPI

在上方菜单选择 PSOC->Program Flash。



在弹出的界面有Procise和IAR，可以选择使用Procise来烧写QSPI，也可以选择IAR中的flashloader来烧写QSPI。

Program Flash

Optional option, importing a config file can automatically configure the programming arguments.

Auto config file:  Export Import

**IAR** **Procise**

### Program Flash Memory Using Jlink

Program Flash Memory via In-system Programmer.

Hardware Platform:

Connection:  New

Device:  Select...

Image File:  Browse

Offset:  Hex

Flash Type:

FSBL File:  Browse

Cable Speed:  khz

☒ Convert ELF to bootloadable SREC format and program Erase Timeout:  S

☐ Blank check after erase Hardware reset, halt after delay:  ms

☐ Verify after flash Scan jtag, halt after max delay:  ms

Advance

Flash operation

Readback Erase

Debug button in independent JTAG boot mode.

Reset PS Scan Jtag

JTAG scan chain configuration.

Program Cancel

Program Flash

Optional option, importing a config file can automatically configure the programming arguments.

Auto config file:  Export Import

IAR Procise

### Program Flash Memory IAR Using I-jet

Program Flash Memory via iar flashloader

Image File:  Browse

Target Device:

Flash Configuration

Flash Type:

Chip Erase before Program: ☐

Offset:

Jtag Configuration

Jtag Mode:

Jtag Speed:  khz

Reset:

Delay:  ms