



PYNQ4CV工程搭建

简介(Introduction)

通过本文档，你将能够了解：

- 如何使用Vivado搭建PYNQ-Z2上可以使用的摄像头输入、HDMI输出的视频通路Overlay
- 如何在Overlay中添加自定义图像处理IP
- 如何在SDK中如何初始化摄像头、配置图像处理IP

所需硬件(Hardware Requirements)

- The PYNQ-Z2 board
- MicroUSB to USB-A cable
- OV5640 camera
- PMOD adapter
- HDMI cable
- HDMI screen

所需软件(Software Requirements)

- Vivado® Design Suite 2018.2
- Board files for PYNQ-Z2 should be installed

Vivado 2018.2默认并不包含PYNQ-Z2的板级支持文件，所以需要将

`{path/to/doc}\files\pynq-z2`文件夹拷贝至
`{path/to/vivado}\2018.2\data\boards\board_file`

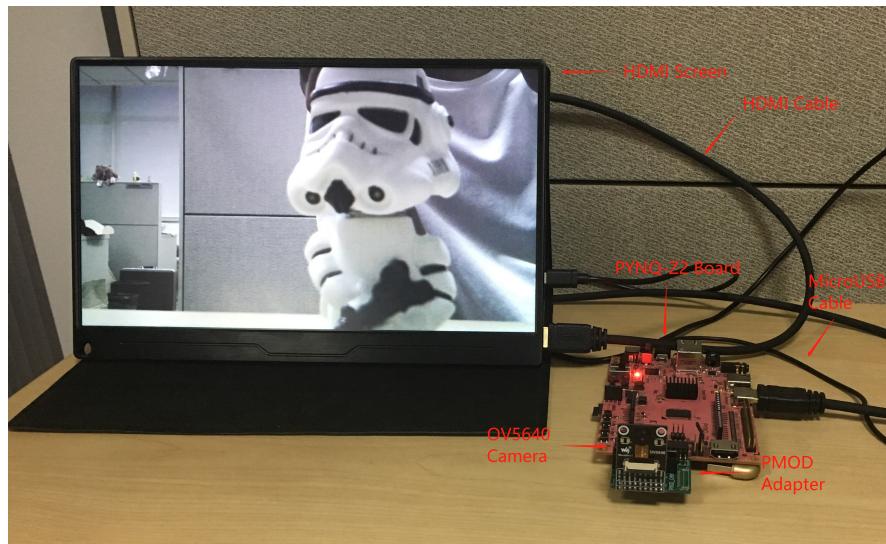
- Xilinx SDK 2018.2
- Vivado HLS 2018.2
- HLS Image Processing IP Core

注意事项(Note)

- {path/to/doc}表示文档根目录
- {path/to/vivado}表示Vivado安装目录
- {path/to/prj}表示所创建工程根目录

硬件连接(Hardware Setup)

将OV5640 camera接到PMOD adapter上并连接至PYNQ-Z2 board PMODA及PMODB，使用HDMI cable连接PYNQ-Z2 HDMI OUT及HDMI screen，使用MicroUSB to USB-A cable连接PROG UART及电脑USB端口。

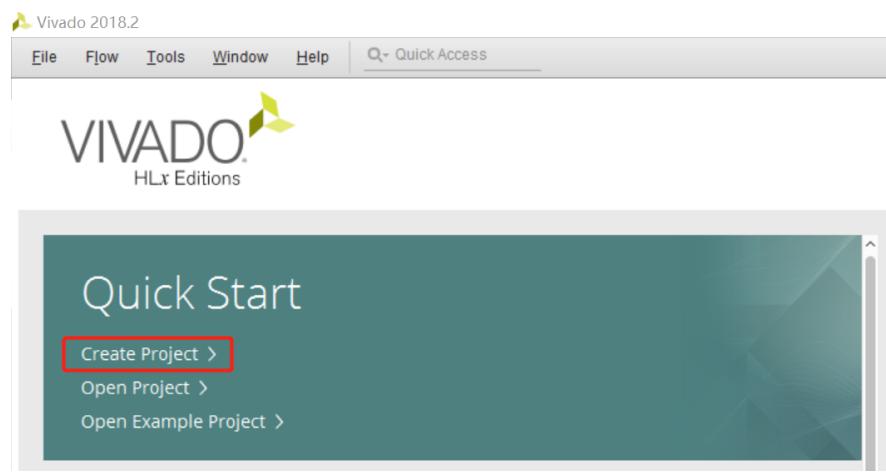


创建工程(Create a Vivado Project)

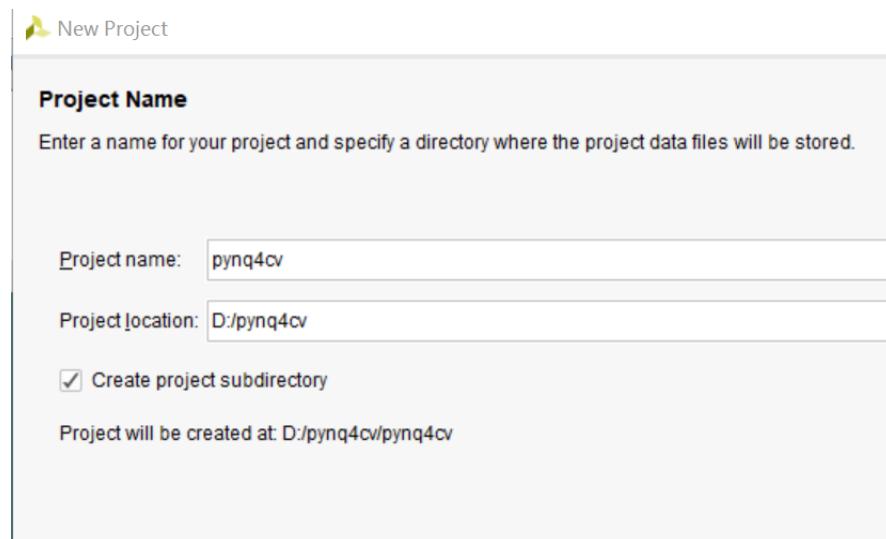
1. 单击 Start > Xilinx Design Tools > vivado 2018.2 打开Vivado。



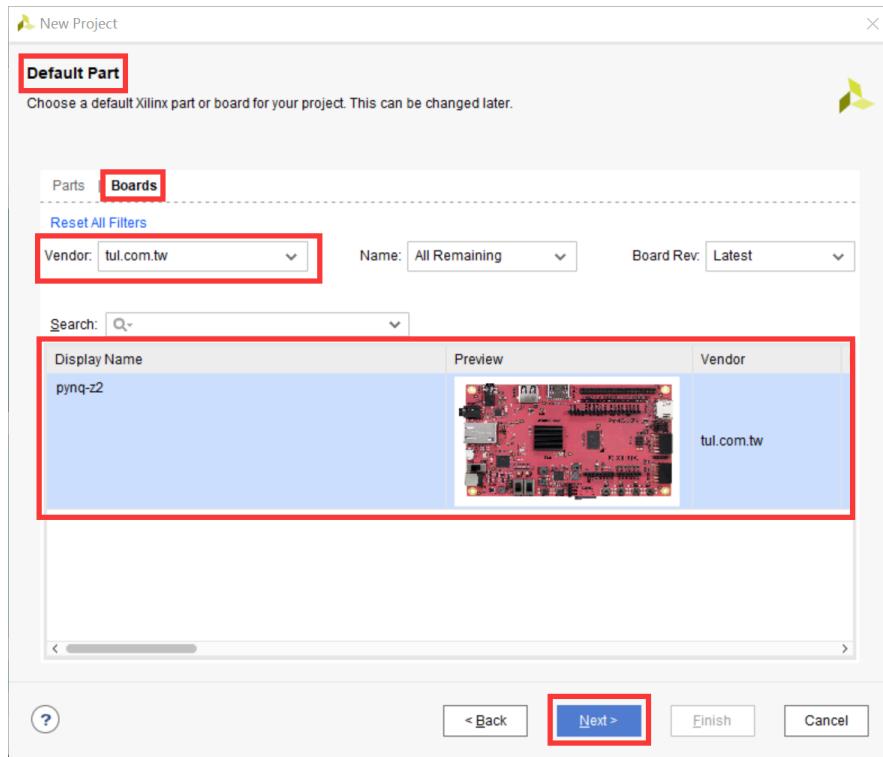
2. 在打开的Vivado窗口 Quick Start 部分单击 Create Project，打开 New Project 窗口，此时处于 Create a New Vivado Project 界面，单击 Next 进入 Project Name 界面。



3. 在 Project name: 栏中输入工程名 pynq4cv , 在 Project location: 栏中选择工程所在位置，即 {path/to/prj} , 单击四次 Next , 跳过 Project Type 、 Add Sources 、 Add Constraint(optional) 界面，进入 Default Part 界面。



4. 在 Default Part 界面，单击 Boards 栏，在 vendor 栏中选择 tul.com.tw , 此时 display Name 中显示 pynq-z2 开发板，如果没有显示 pynq-z2 开发板，请检查是否已经安装 PYNQ-Z2 的板级支持文件。单击显示的 pynq-z2 开发板一栏，显示浅蓝色后单击 Next 进入 New Project Summary 界面。

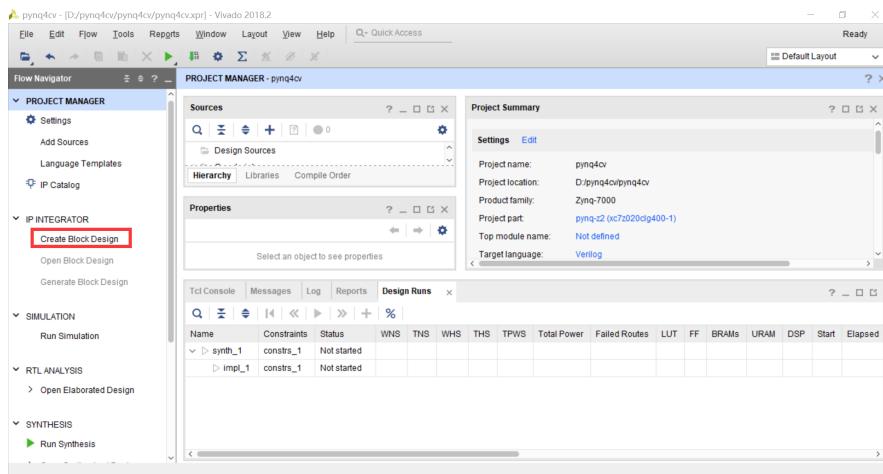


5. 在 New Project Summary 界面单击 Finish 完成工程创建，稍等片刻，进入工程界面，

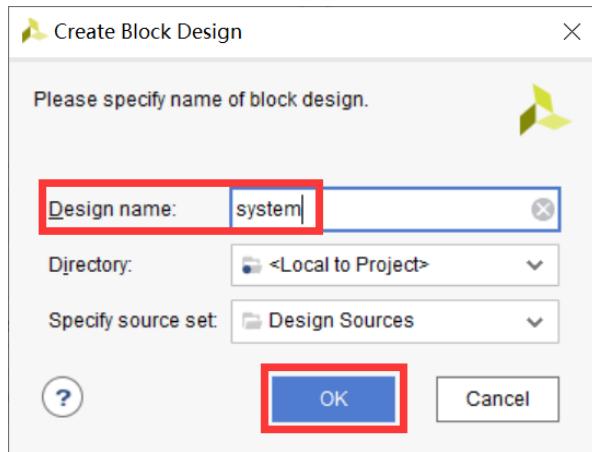
搭建Overlay(Create a Block Design)

输入模块

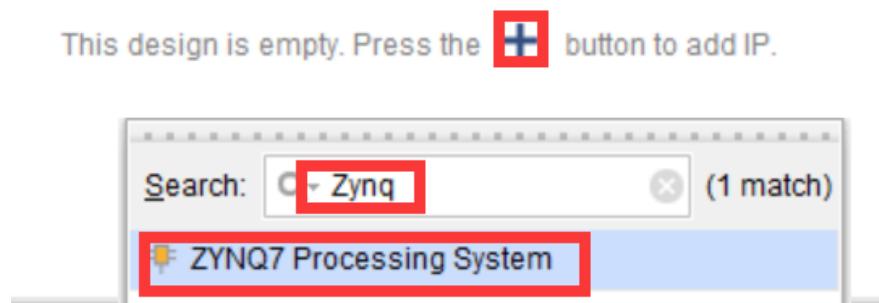
1. 单击 IP INTERGRATOR > Create Block Design，弹出 Create Block Design 对话框。



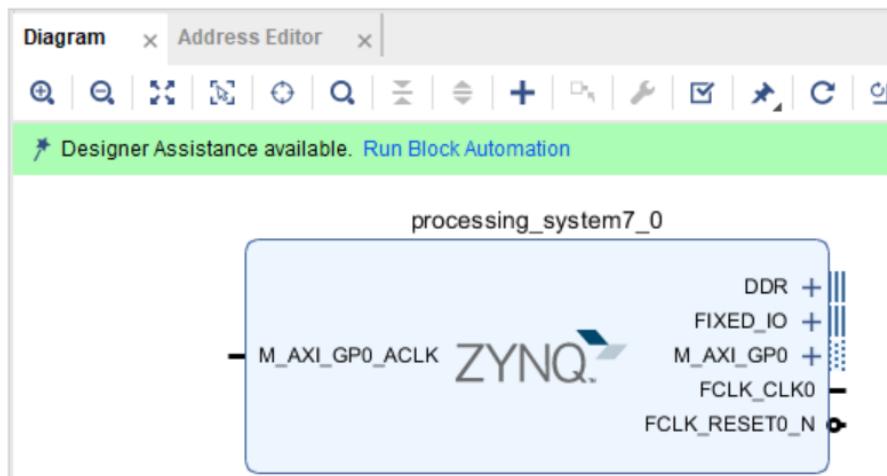
2. 在 Create Block Design 对话框中，修改 Design name: 为 system，单击 OK。



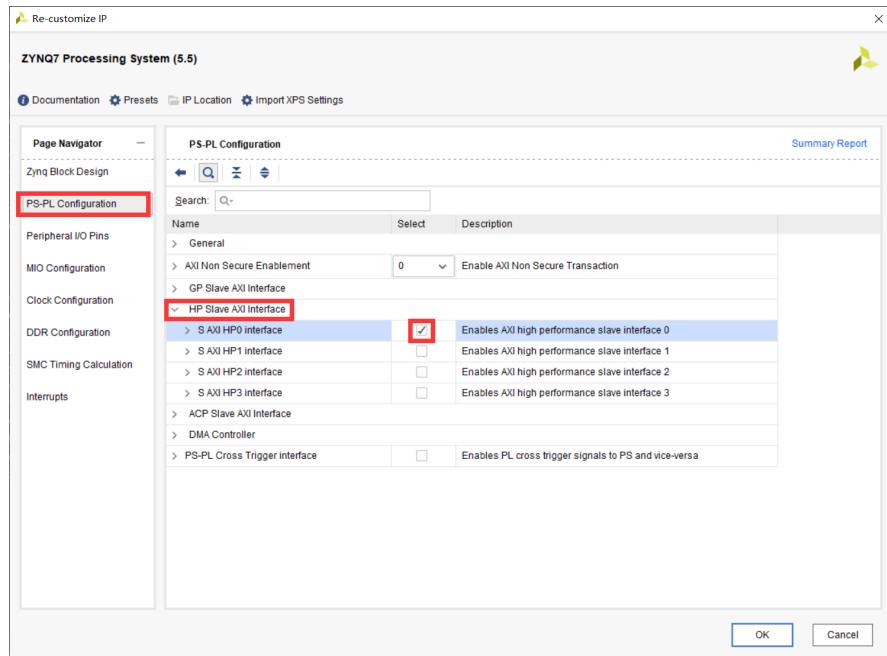
3. 在 Diagram 窗口单击蓝色加号，在 search: 栏中输入 zynq， 并双击 ZYNQ7 Processing System 添加ZYNQ IP。



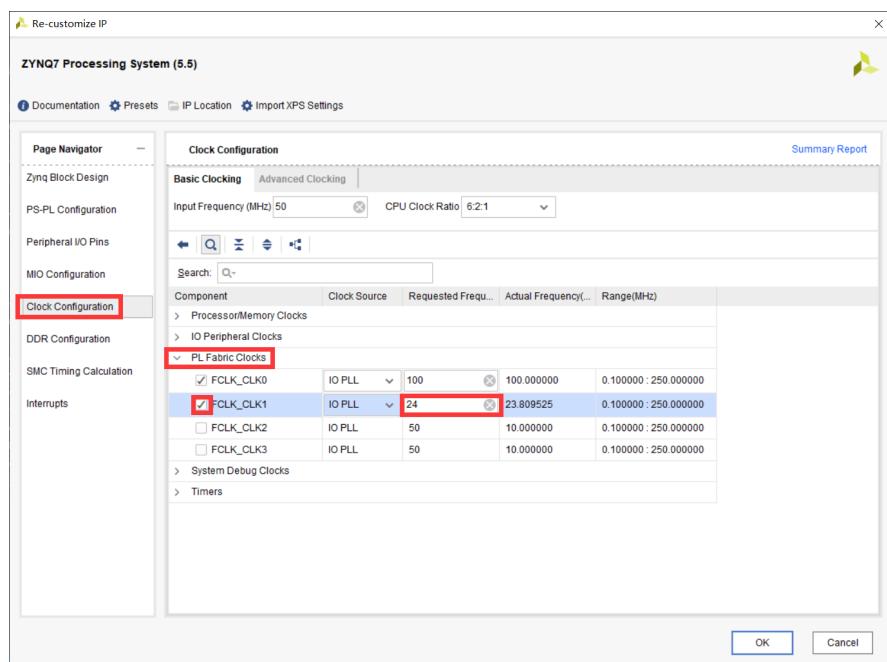
4. 单击 Diagram 窗口左上角 Run Block Automation， 在弹出对话框单击 OK， 自动配置ZYNQ IP。



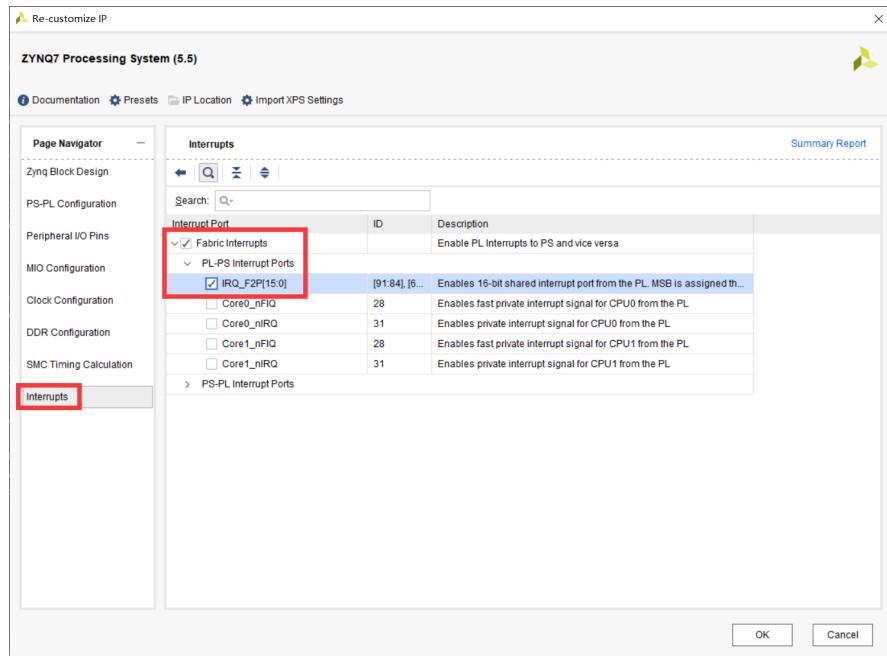
5. 双击 ZYNQ7 Processing System， 打开 Re-customize IP， 单击并勾选 PS-PL Configuration > HP Slave AXI Interface > S AXI HP0 interface。



6. 单击并勾选 Clock Configuration > PL Fabric Clocks > FCLK_CLK1，并修改频率为24。



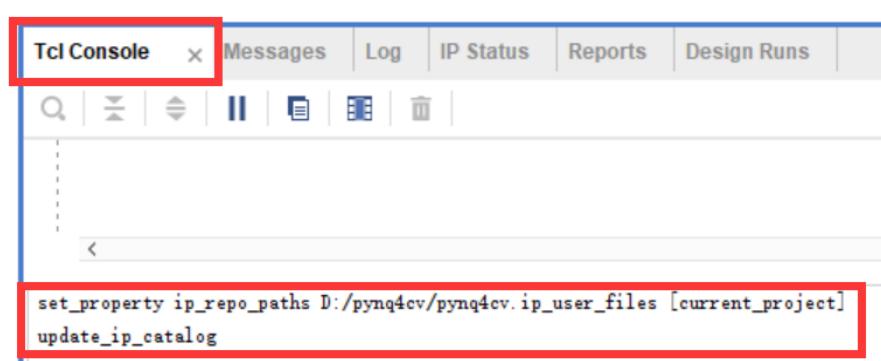
7. 单击并勾选 Interrupts > Fabric Interrupts > PL-PS Interrupts Ports > IRQ_F2P[15:0]，单击 OK，完成配置。



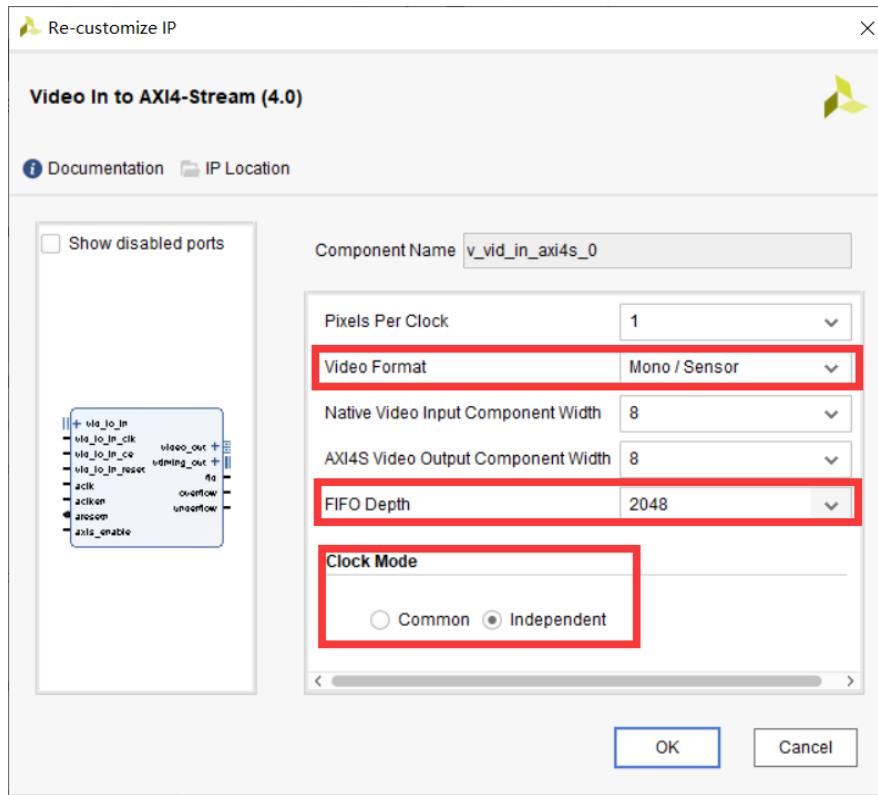
8. 将 {path/to/doc}\files\ip 文件夹拷贝至 {path/to/prj}\pynq4cv.ip_user_files，在Tcl Console 的输入栏，输入

```
set_property ip_repo_paths {path/to/prj}/pynq4cv.ip_user_files [current_project]
update_ip_catalog
```

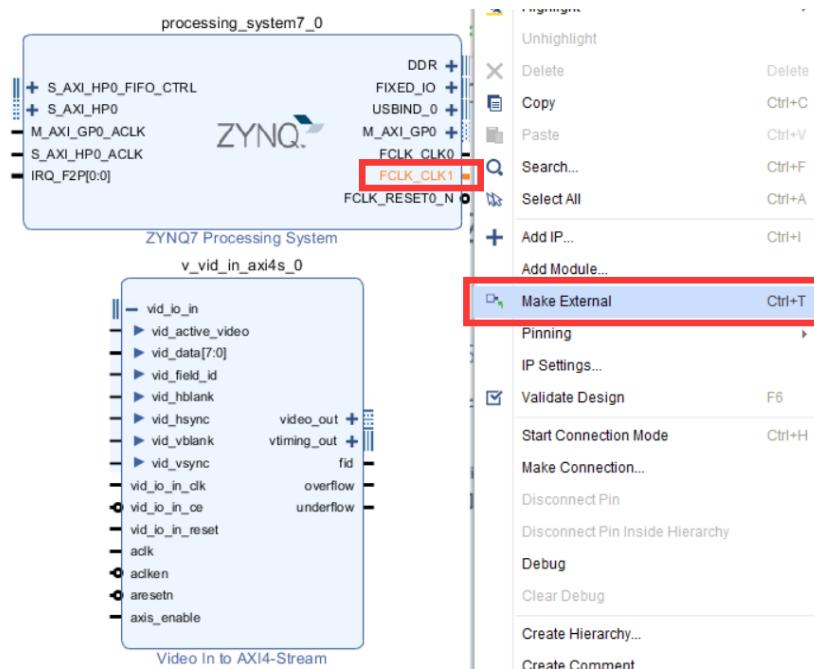
WARNING: {path/to/prj} 要替换成相应工程路径，如 D:/pynq4cv/，此时应该使用 / 而非 \。



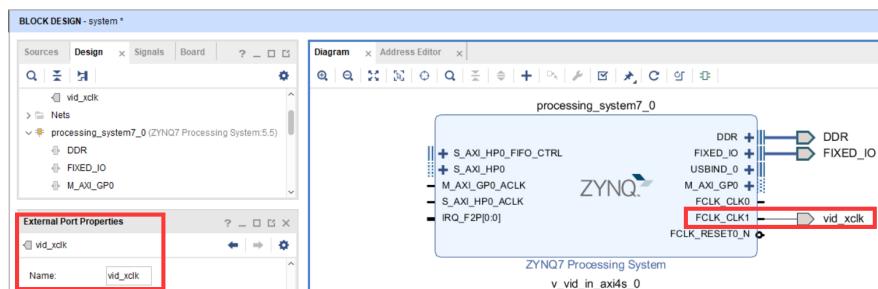
9. 单击 Diagram 界面蓝色加号 Add IP，或者使用快捷键 Ctrl+I，在 Search: 栏中输入 video to stream，双击 Video In to AXI4-Stream，双击添加到 Diagram 界面的IP，修改 video Format 为 Mono/Sensor，修改 FIFO Depth 为 2048，修改 Clock Mode 为 Independent，单击 OK 确认。



10. 单击选中 processing_system7_0 的 FCLK_CLK1 引脚，单击邮件，选择 Make External，也可以使用快捷键 **ctrl+T** 引出端口。

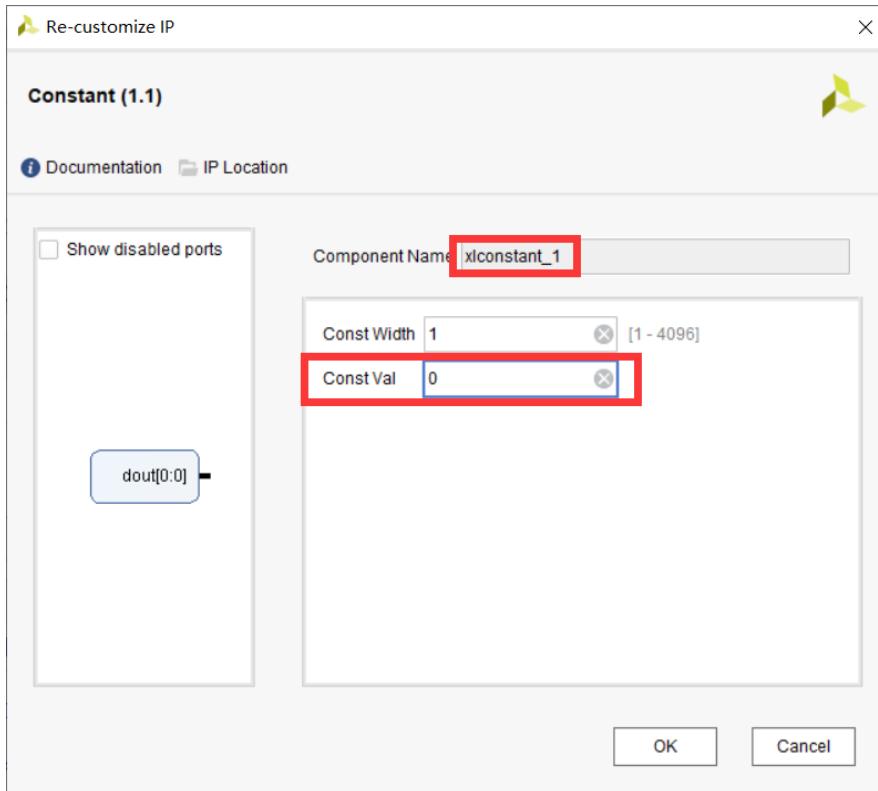


11. 单击选中 FCLK_CLK1_0 端口，修改名字为 vid_xclk。

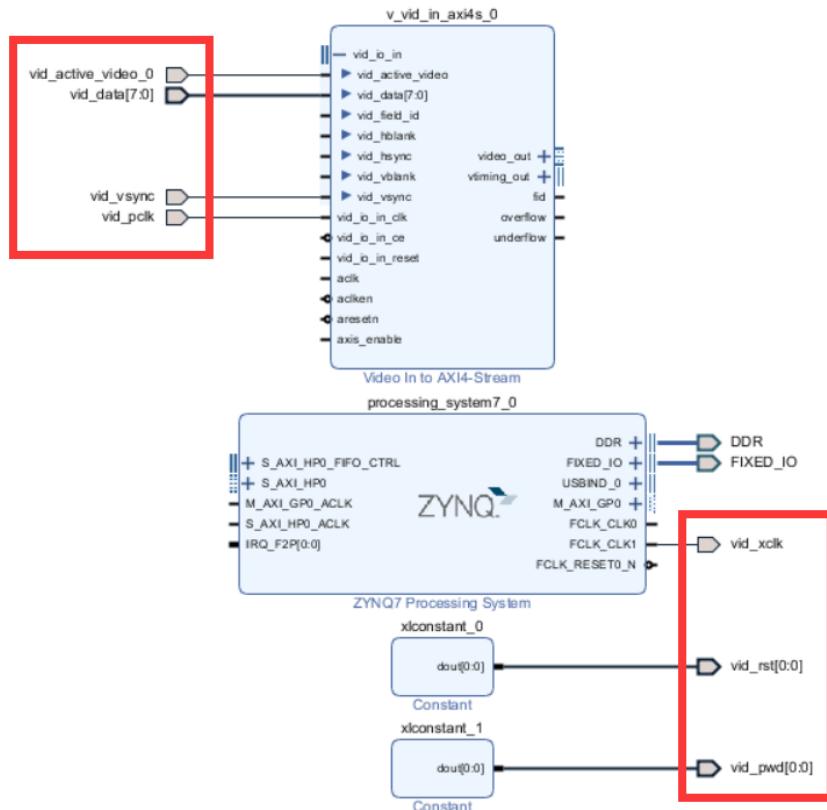


12. 单击 v_vid_in_axi4s_0 的 vid_io_i 接口上的蓝色加号，分别引出 vid_active_video、vid_data，vid_vsync，vid_io_in_clk，并修改端口名为 vid_hsync，vid_data，vid_vsync，vid_pcclk。

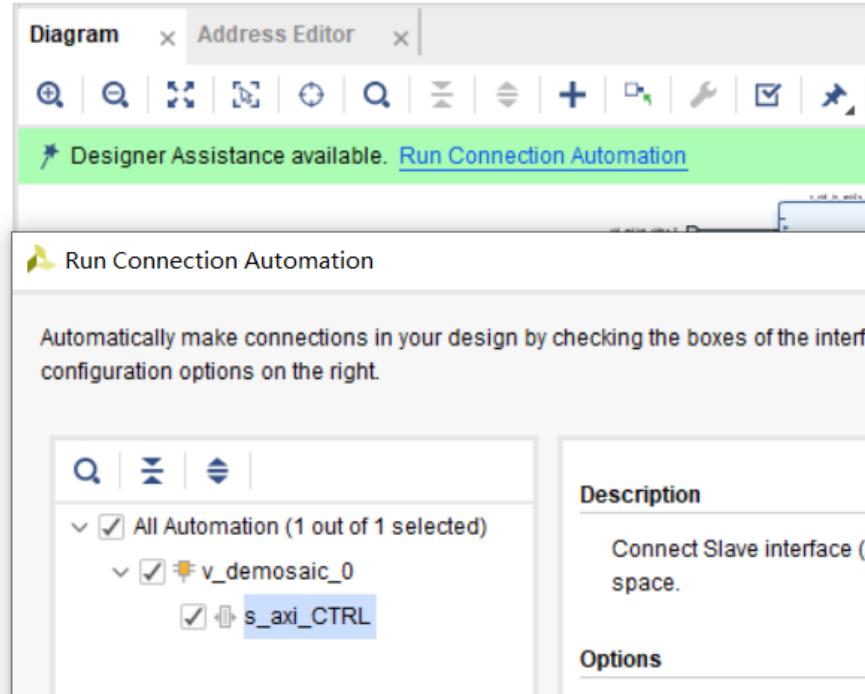
13. 添加两个 Constant，双击 x1constant_1，修改 x1constant_1 的 Const Val 为 0。



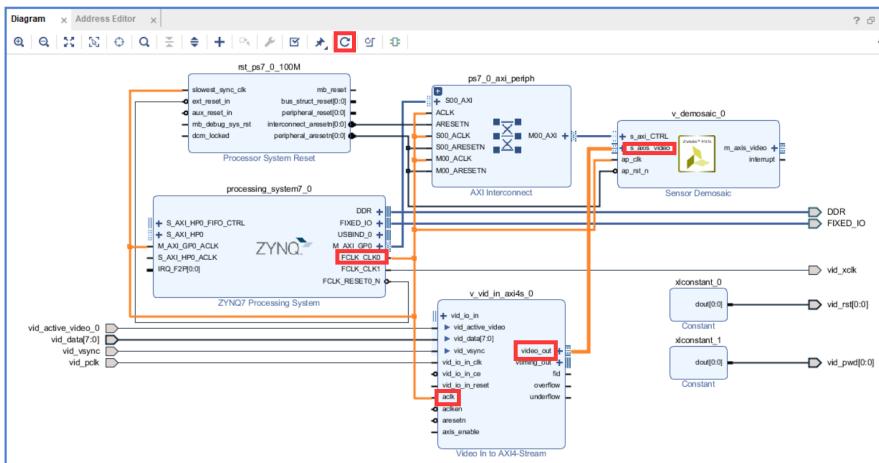
14. 引出 x1constant_0 的 dout 并修改名字为 vid_RST，引出 x1constant_1 的 dout 并修改名字为 vid_PWD。



15. 添加 Sensor Demosaic，单击 Diagram 左上角 Run Connection Automation，在弹出窗口，确认 s_axi_CTRL 勾选，单击 OK。

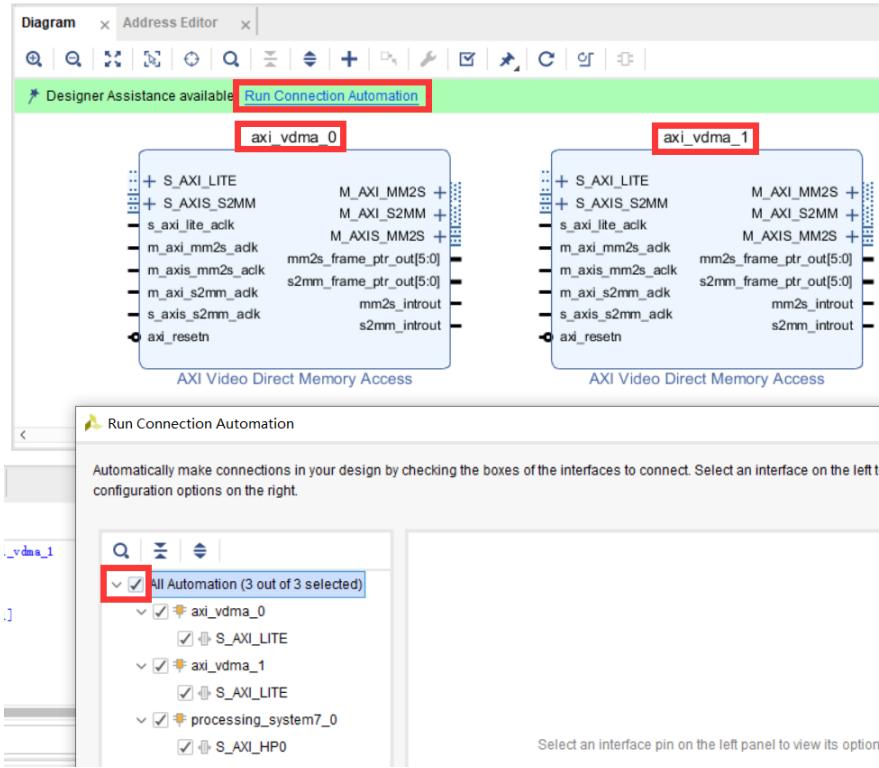


16. 连接 `v_vid_in_axi4s_0 > aclk -> processing_system7_0 > FCLK_CLK0` 和
`v_vid_in_axi4s_0 > video_out --> v_demosaic_0 > s_axis_video`, 单击 Regenerate Layout, 完成视频输入部分搭建。

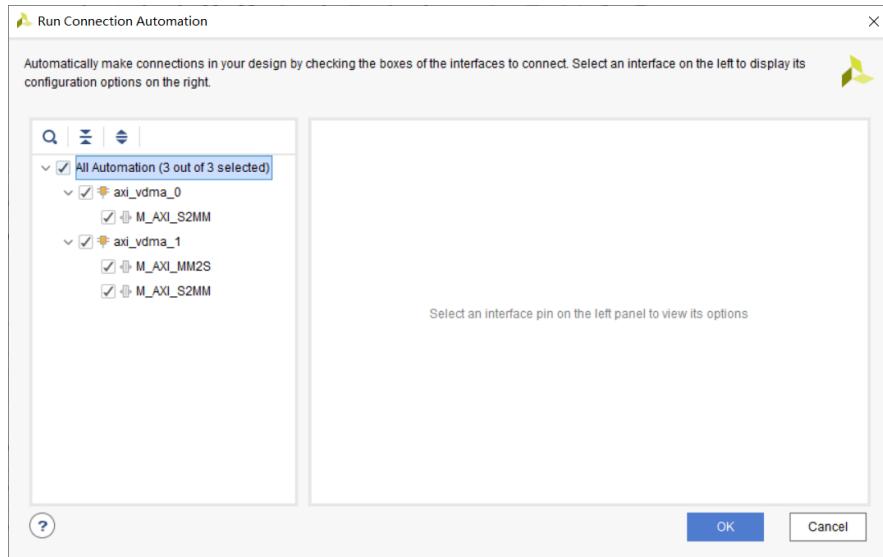


缓存模块

1. 添加两个 AXI Video Direct Memory Access 模块, 即 VDMA, 此时出现 `axi_vdma_0` 和 `axi_vdma_1`, 单击 Diagram 左上角 Run Connection Automation。



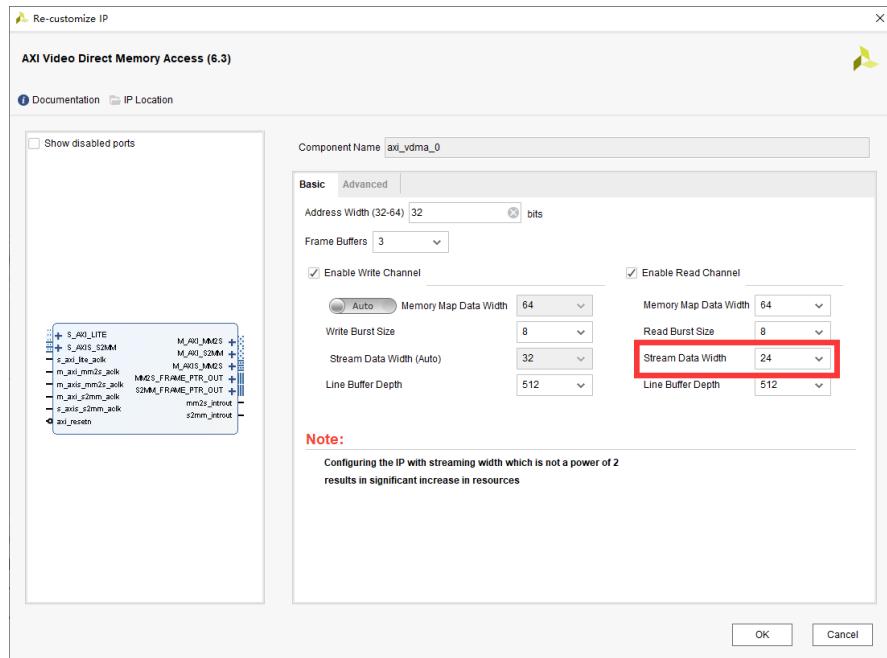
2. 在弹出窗口，勾选 All Automation，单击 OK。此时会自动连接时钟、复位、HP等接口。



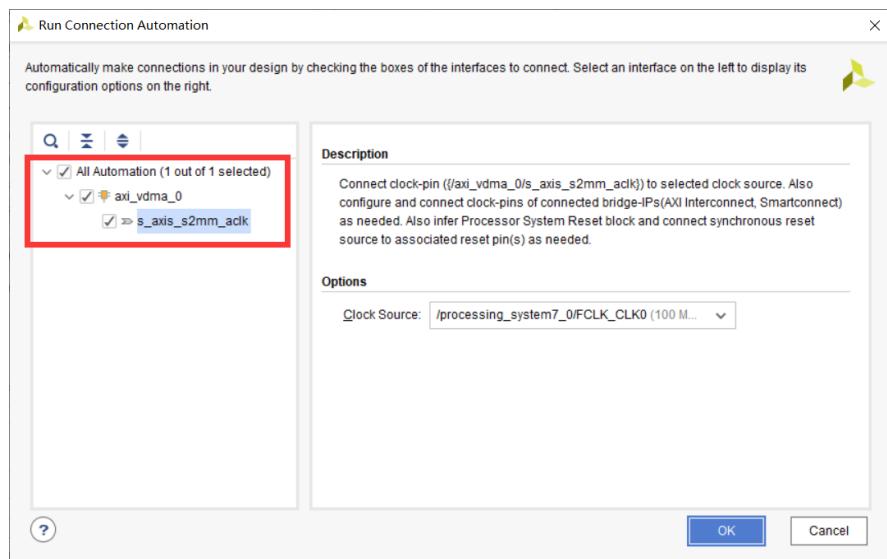
3. 单击 diagram 旁的 Address Editor，可以看到 VDMA 已经自动分配好了地址。

Cell	Slave Interface	Base Name	Offset Address	Range	High Address
processing_system7_0					
	Data (32 address bits : 0x40000000 [1G])				
axi_vdma_0					
	Data_MM2S (32 address bits : 4G)	processing_system7_0	S_AXI_HP0	HP0_DDR_LOWOCM	0x0000_0000
					512M ▾
	Data_S2MM (32 address bits : 4G)	processing_system7_0	S_AXI_HP0	HP0_DDR_LOWOCM	0x0000_0000
					512M ▾
axi_vdma_1					
	Data_MM2S (32 address bits : 4G)	processing_system7_0	S_AXI_HP0	HP0_DDR_LOWOCM	0x0000_0000
					512M ▾
	Data_S2MM (32 address bits : 4G)	processing_system7_0	S_AXI_HP0	HP0_DDR_LOWOCM	0x0000_0000
					512M ▾

4. 切换回 Diagram，双击 axi_vdma_0 和 axi_vdma_1，修改 Stream Data Width 为 24。

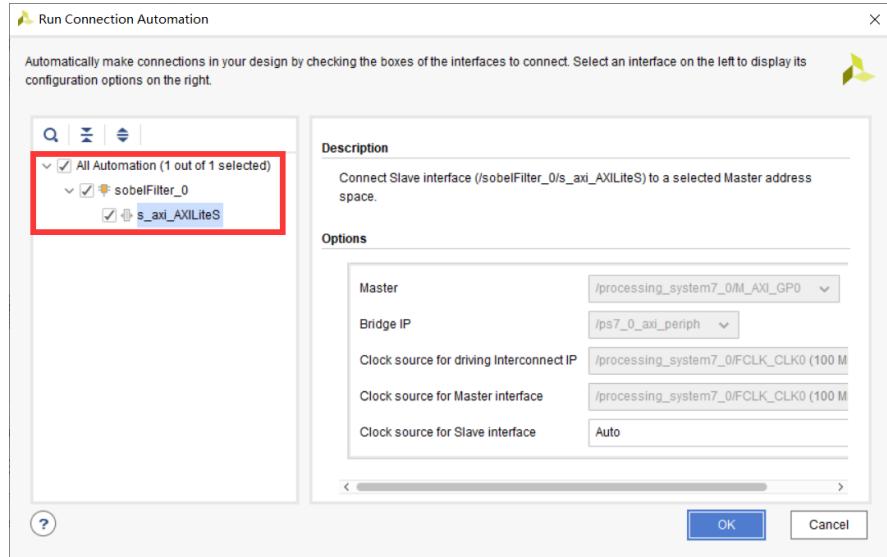


5. 连接 v_demosaic_0 > m_axis_video 和 axi_vdma_0 > s_AXIS_S2MM，单击 Diagram 左上角 Run Connection Automation，在弹出窗口，确认 All Automation 勾选，单击 OK。

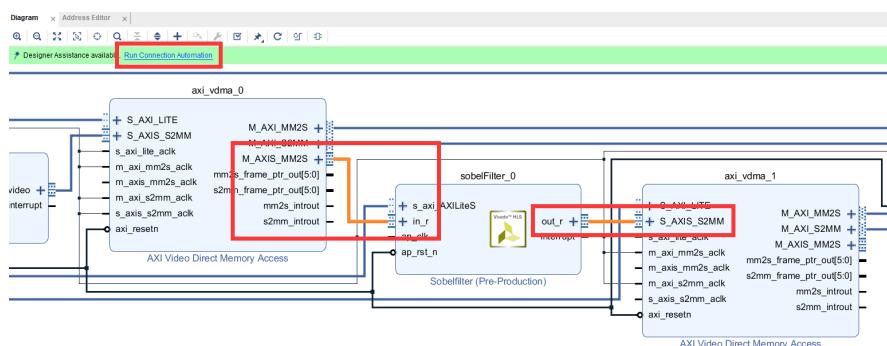


处理模块

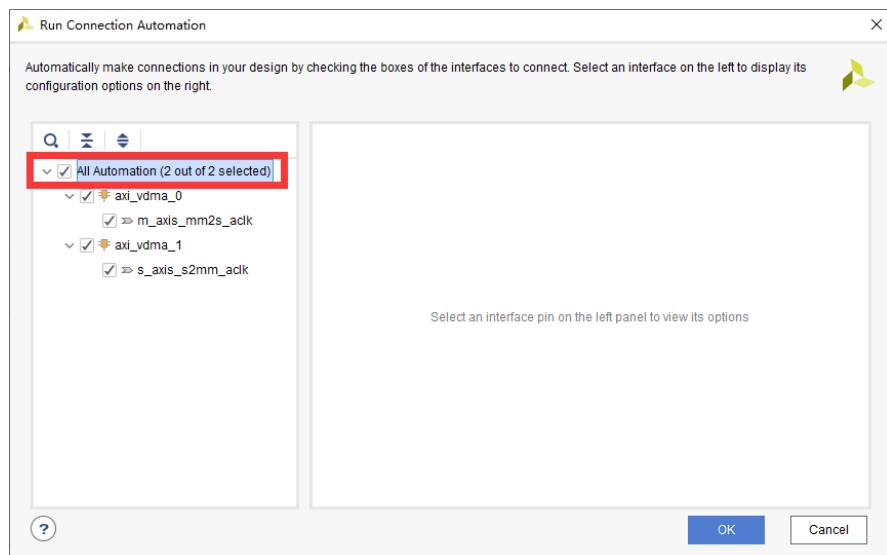
1. 添加 sobelfilter，单击 Diagram 左上角 Run Connection Automation，在弹出窗口，确认 All Automation 勾选，单击 OK。



2. 连接 `axi_vdma_0 > M_AXIS_MM2S --> sobelFilter_0 > in_r` 和 `sobelFilter_0 > out_r --> axi_vdma_1 > S_AXIS_S2MM`，再次单击 Diagram 左上角 Run Connection Automation。

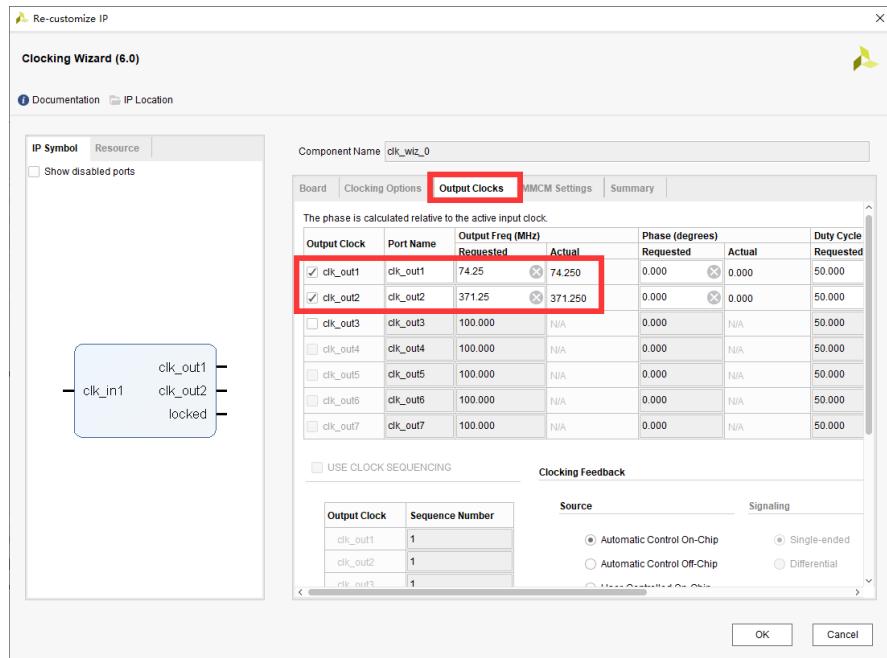


2. 在弹出窗口，确认 All Automation 勾选，单击 OK。

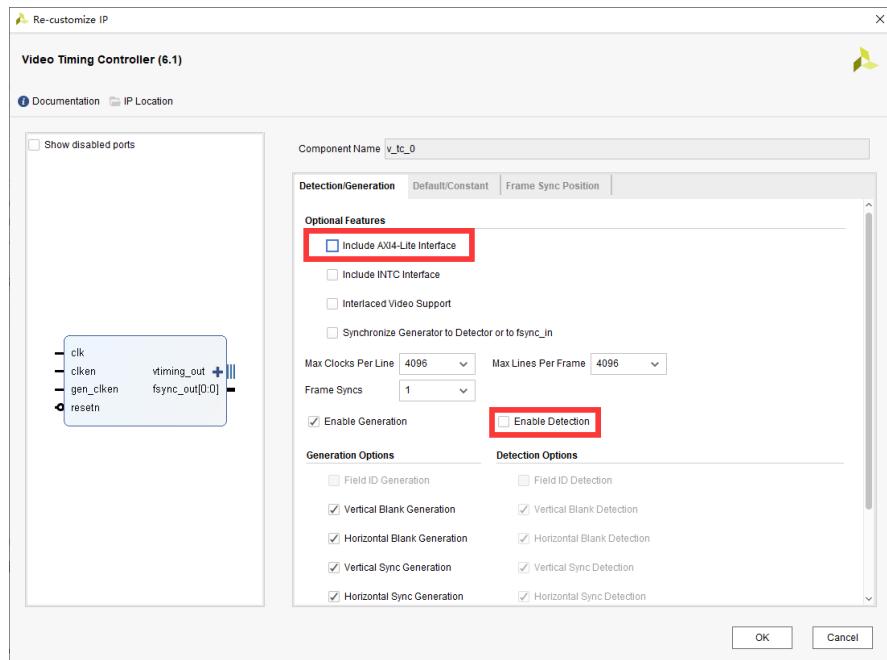


输出模块

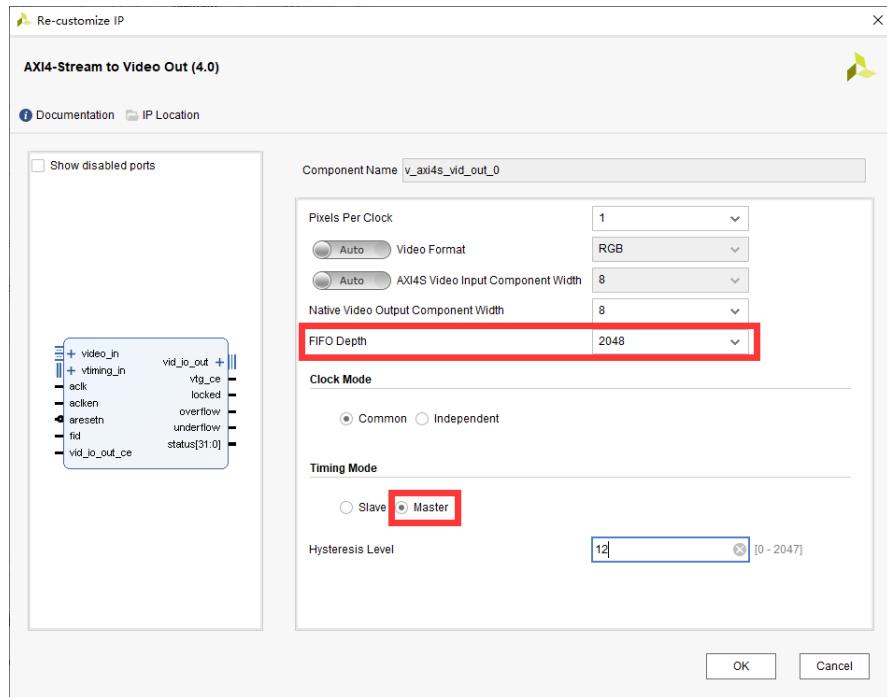
- 添加 `clocking wizard`、`Video Timing Controller`、`AXI4-Stream to Video Out` 和 `RGB to DVI Video Encoder` 模块。
- 双击 `clocking wizard`，单击 `output clocks` 界面，修改 `clk_out1` 的 `output Freq` 为 `74.25`，勾选 `clk_out2`，修改 `output Freq` 为 `371.25`，向下拉到 `Enable Optional Inputs/Outputs for MMCM/PLL`，取消勾选 `reset`，单击 `OK` 确认。



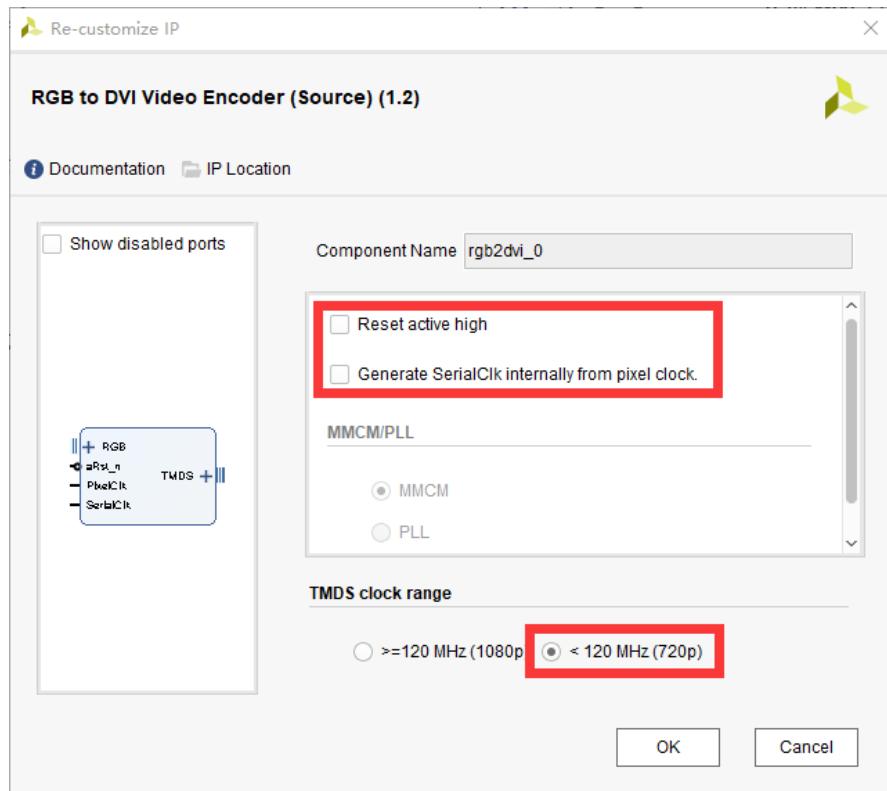
3. 双击 Video Timing Controller , 取消勾选 Optional Features 栏中 Include AXI4-Lite Interface 和 Enable Detection , 单击 OK 确认。



4. 双击 AXI4-Stream to Video Out , 修改 FIFO Depth 为 2048 , 修改 Timing Mode 为 Master , 单击 OK 确认。



5. 双击 `rgb2dvi_0`，取消勾选 `Reset active high`, `Generate SerialClk internally from pixel clock.`，在 `TMDS clock range` 栏中，勾选 `<120MHz(720p)`，单击 `OK` 确认。



6. 连接下列端口

```

processing_system7_0 > FCLK_CLK1 --> clk_wiz_0 > clk_in1

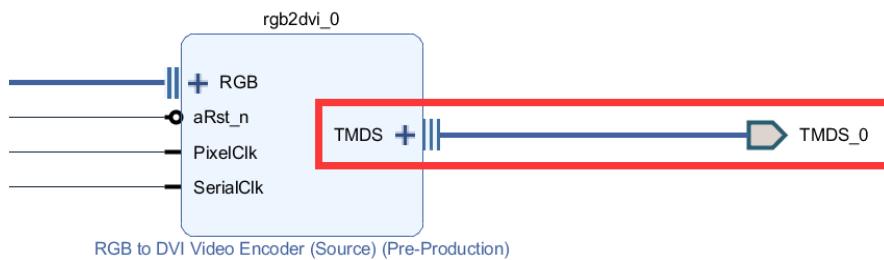
clk_wiz_0 > clk_out1 --> v_tc_0 > clk
clk_wiz_0 > clk_out1 --> v_axi4s_vid_out_0 > aclk
clk_wiz_0 > clk_out1 --> axi_vdma_1 > m_axis_mm2s_aclk

clk_wiz_0 > clk_out1 --> rgb2dvi > PixelClk
clk_wiz_0 > clk_out2 --> rgb2dvi > SerialClk
clk_wiz_0 > locked --> rgb2dvi > aRst_n

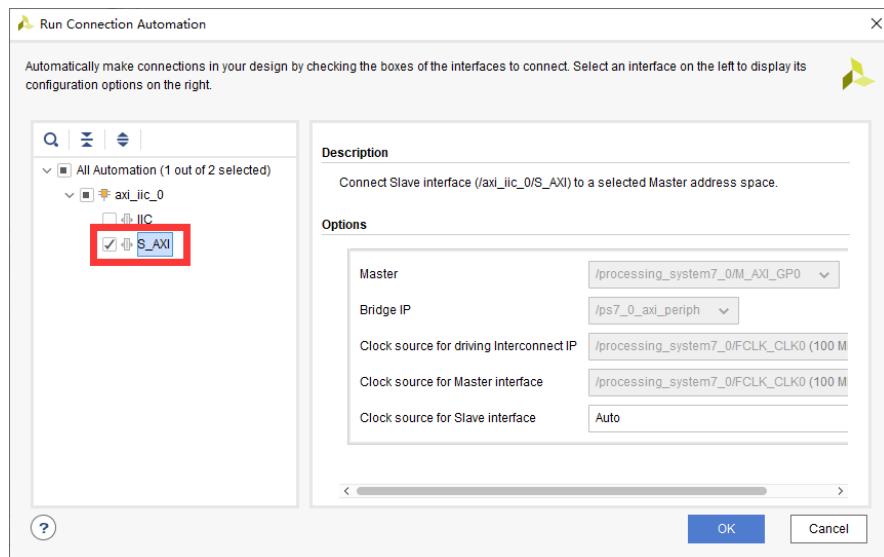
v_tc_0 > vtiming_out --> v_axi4s_vid_out_0 > vtiming_in
axi_vdma_1 > M_AXIS_MM2S --> v_axi4s_vid_out_0 > video_in
v_axi4s_vid_out_0 > vid_io_out --> rgb2dvi_0 > RGB

```

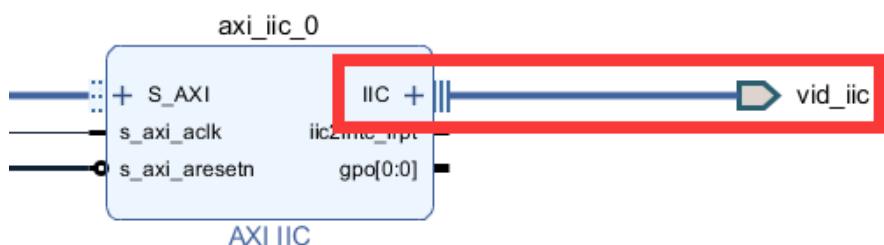
7. 单击选中 `rgb2dvi_0` 的 TMDS 并引出端口。



8. 添加 AXI IIC 模块，单击 Diagram 左上角 Run Connection Automation，在弹出窗口，确认勾选 S_AXI，取消勾选 IIC，单击 OK 确认。

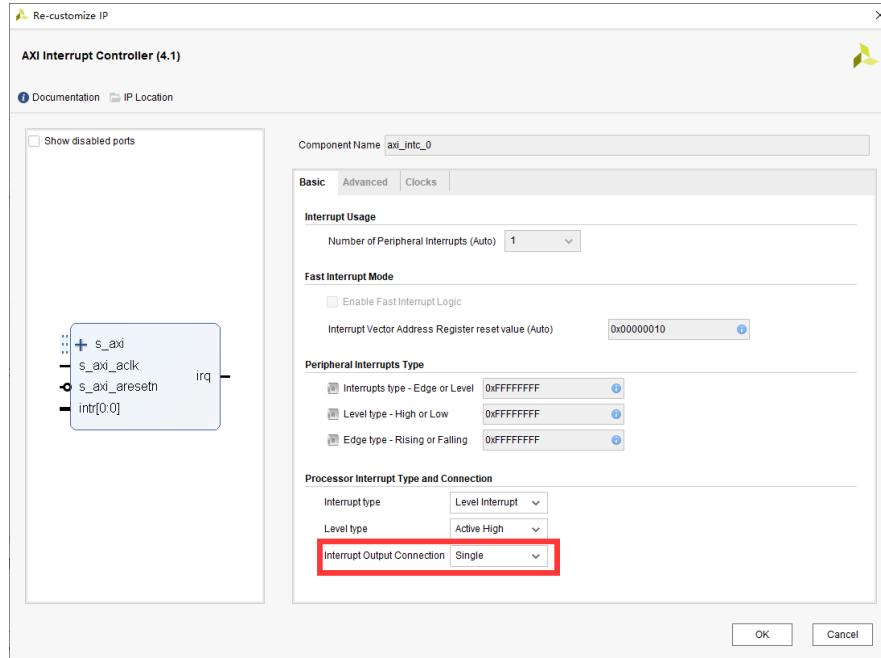


7. 单击选中 `axi_iic_0` 的 IIC 并引出端口，修改端口名为 `vid_iic`。

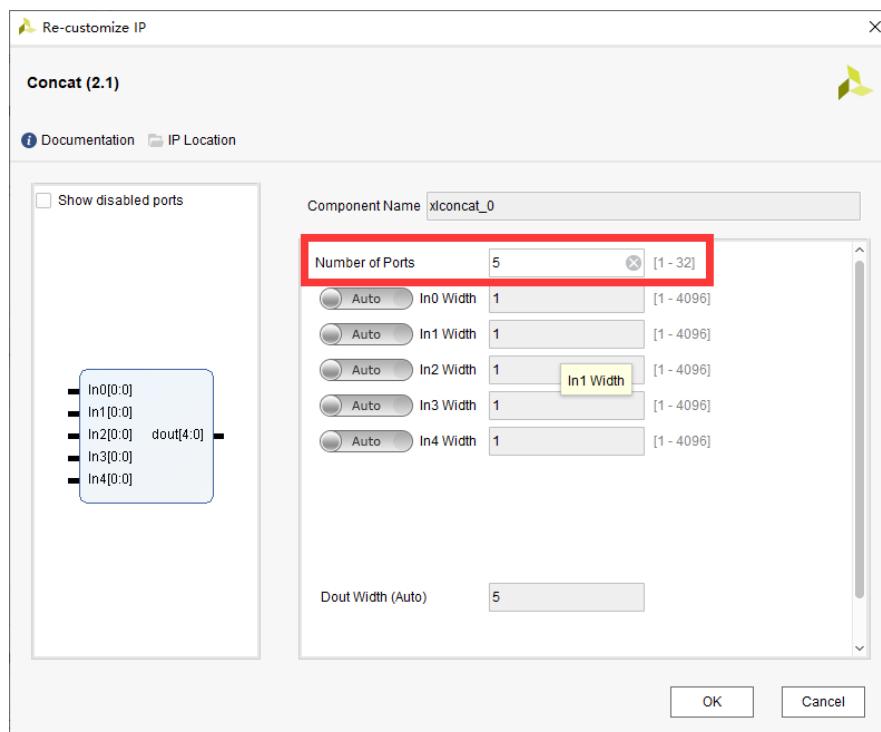


添加中断

- 添加 AXI Interrupt Controller 模块，单击 Diagram 左上角 Run Connection Automation，在弹出窗口，确认勾选 All Automation，单击 OK 确认。
- 双击 axi_intc_0，修改 Interrupt Output Connection 为 single，单击 OK 确认。



- 添加 concat 模块，双击 xlconcat_0，修改 Number of Ports 为 5，单击 OK 确认。



- 连接下列端口

```

processing_system7_0 > FCLK_CLK1 --> clk_wiz_0 > clk_in1

axi_vdma_0 > mm2s_introut --> xlconcat_0 > In0
axi_vdma_0 > s2mm_introut --> xlconcat_0 > In1

axi_vdma_1 > mm2s_introut --> xlconcat_0 > In2
axi_vdma_1 > s2mm_introut --> xlconcat_0 > In3

axi_iic_0 > iic2intc_irpt --> xlconcat_0 > In4

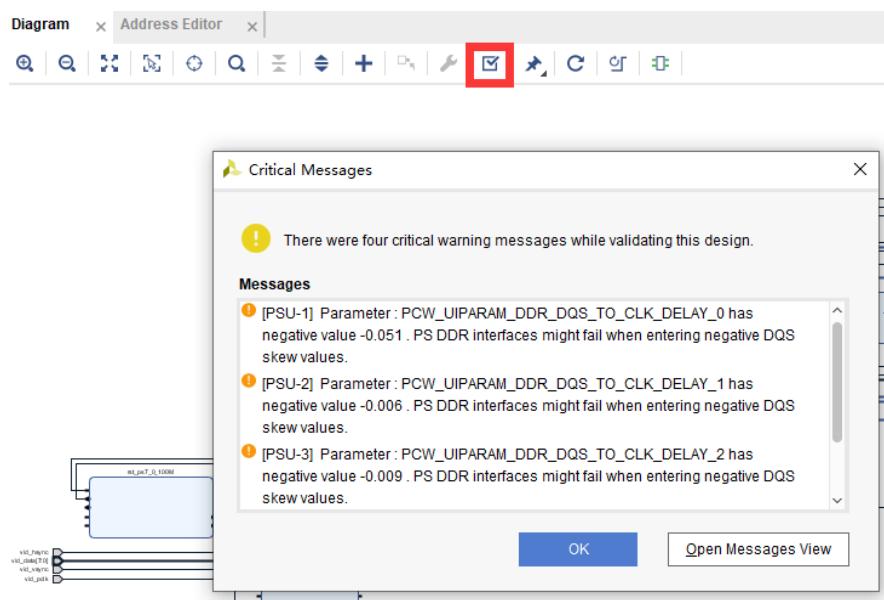
xlconcat_0 > dout --> axi_intc_0 > intr

axi_intc_0 > irq --> processing_system7_0 > IRQ_F2P

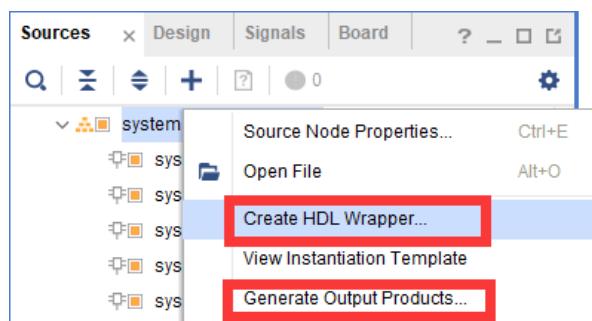
```

创建顶层

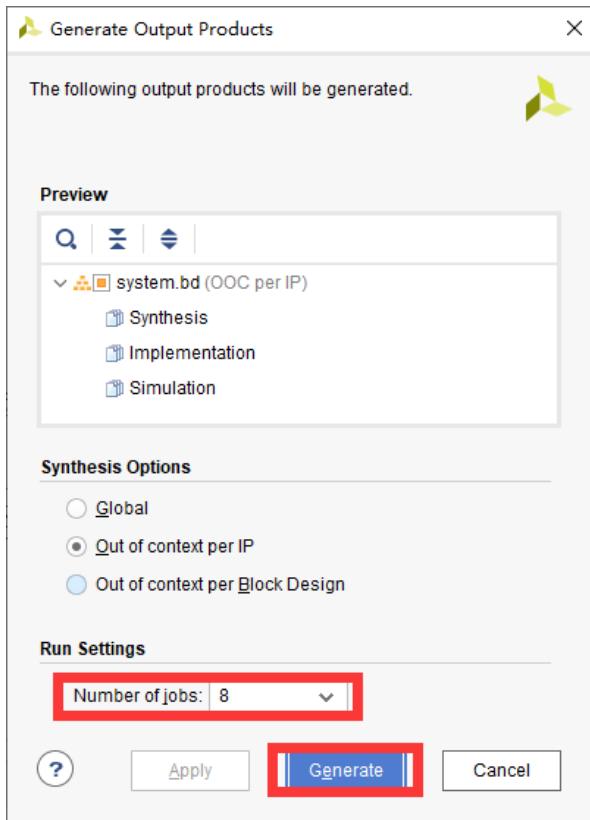
- 单击 validate Design 或者按下 F6，如果弹出如下警告单击 OK 忽略。



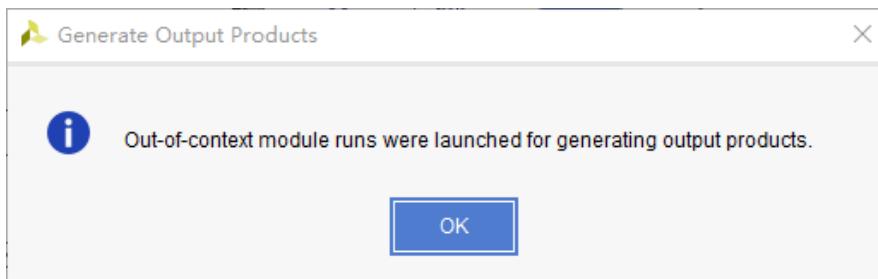
- 在 Sources 栏中右键单击 system，选择 Generate Output Products...，



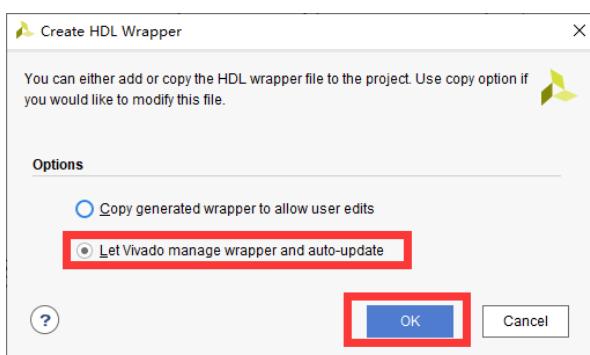
- 在弹出窗口单击 Generate，其中 Number of jobs: 和电脑配置相关，可修改为能够选择的最大数。



4. 稍等片刻，当生成文件完成后，会弹出 Generate Output Products 窗口，单击 OK，在 Sources 栏中右键单击 system，选择 Create HDL Wrapper...。

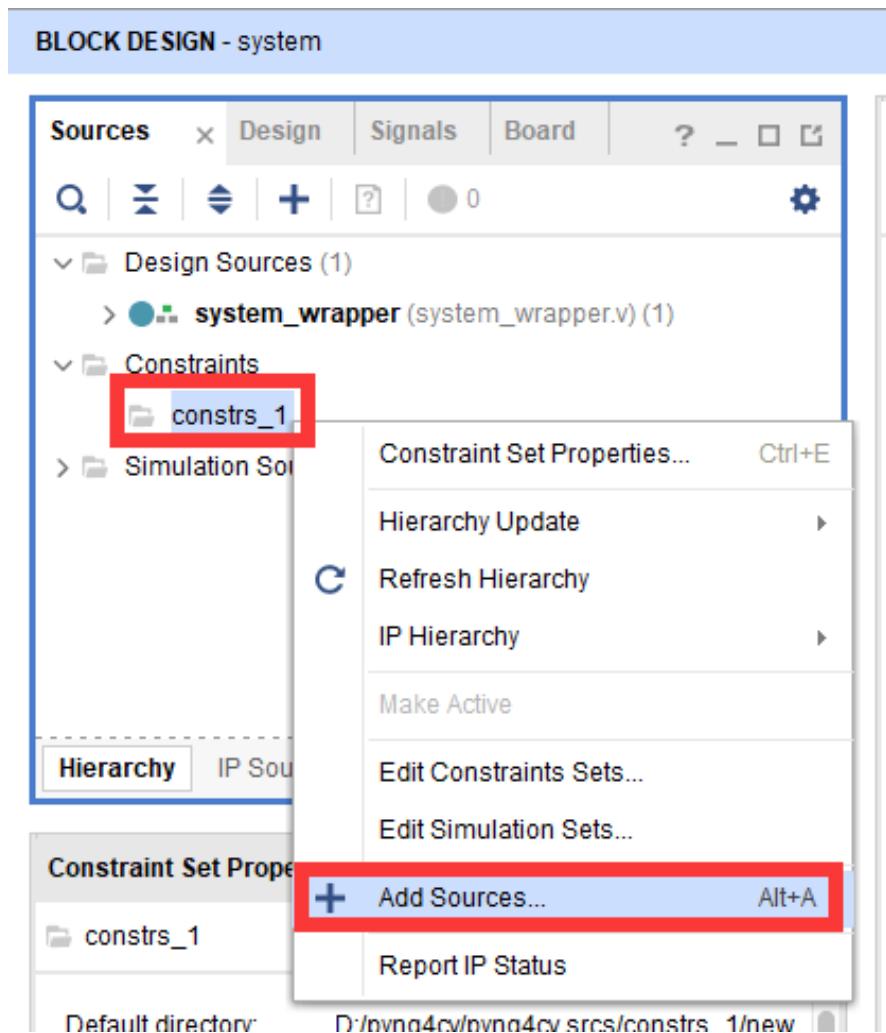


5. 确认默认勾选为 Let Vivado manage wrapper and auto-update，在弹出窗口单击 OK，生成顶层文件。

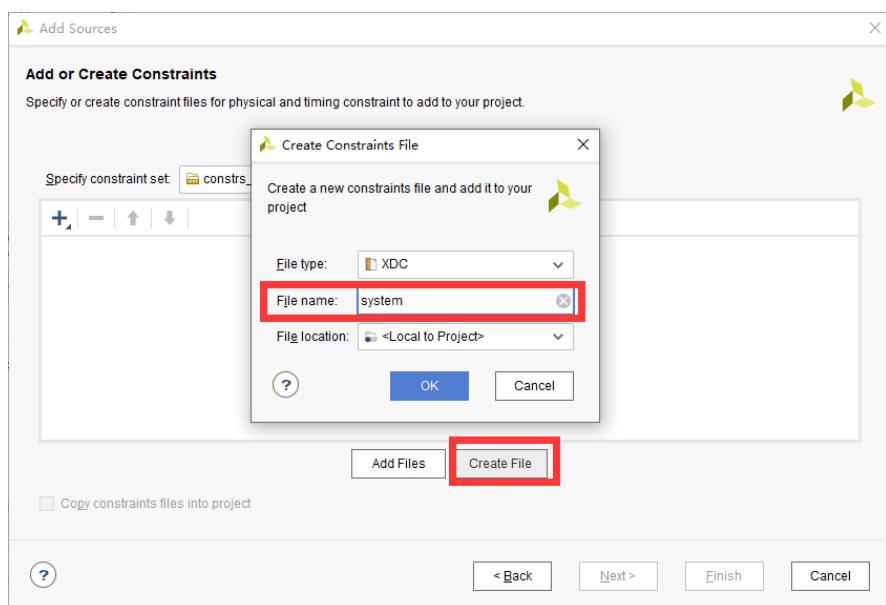


添加约束

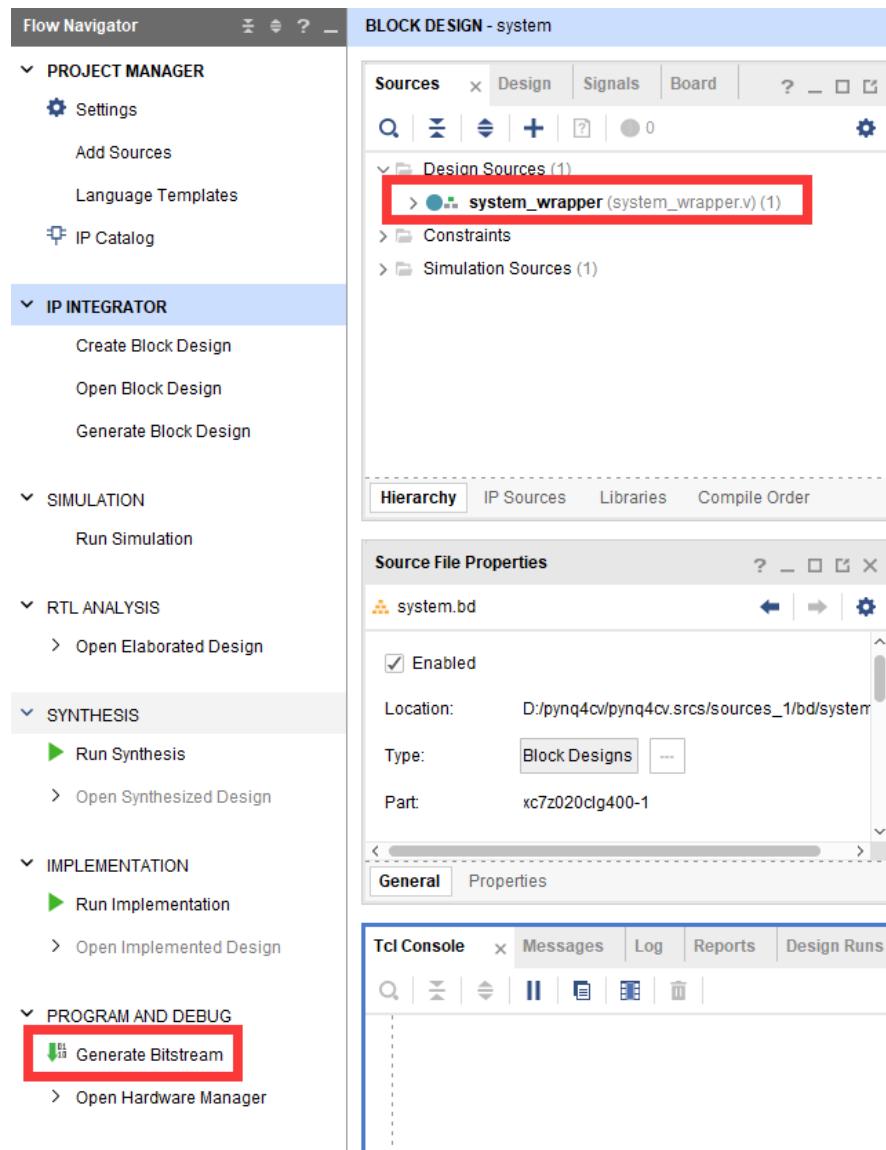
1. 当 sources 栏中出现 system_wrapper 后，右键单击 sources > Constraints 下的 constrs_1，单击 Add Sources...。



- 在弹出窗口单击 Next，在 Add or Create Constraints 窗口，单击 Create File，在弹出窗口修改 File Name: 为 system，单击 OK 和 Finish。

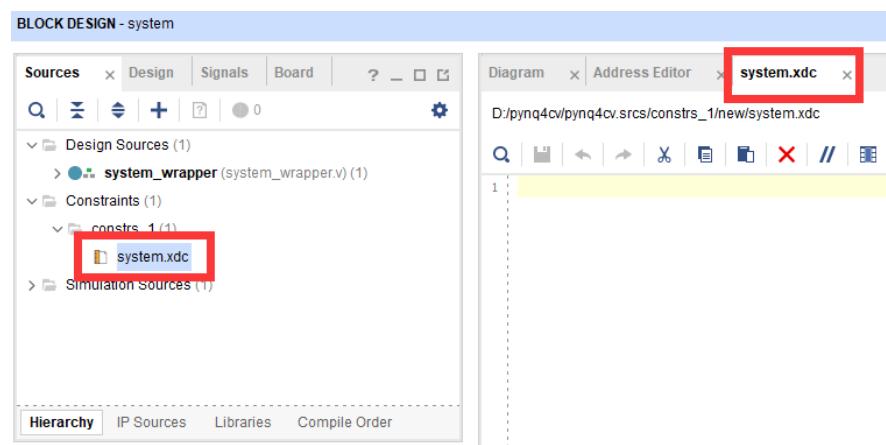


- 双击打开 Sources > Constraints > constrs_1 > system.xdc, 将 {path/to/doc}\files\system.xdc 中的内容拷贝至文本栏中并保存。

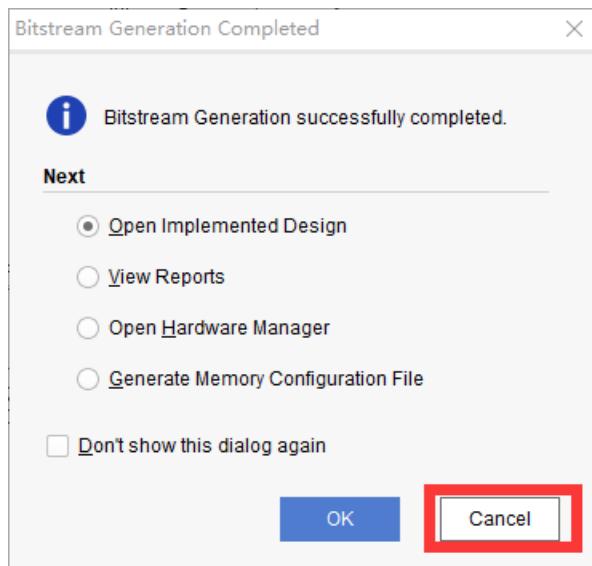


生成Bitstream

1. 单击 Flow Navigator 下的 Generate Bitstream，在弹出窗口单击 Yes，在弹出 Launch Runs 窗口单击 OK，之后生成 Bitstream 文件的过程需要等待十五分钟左右。

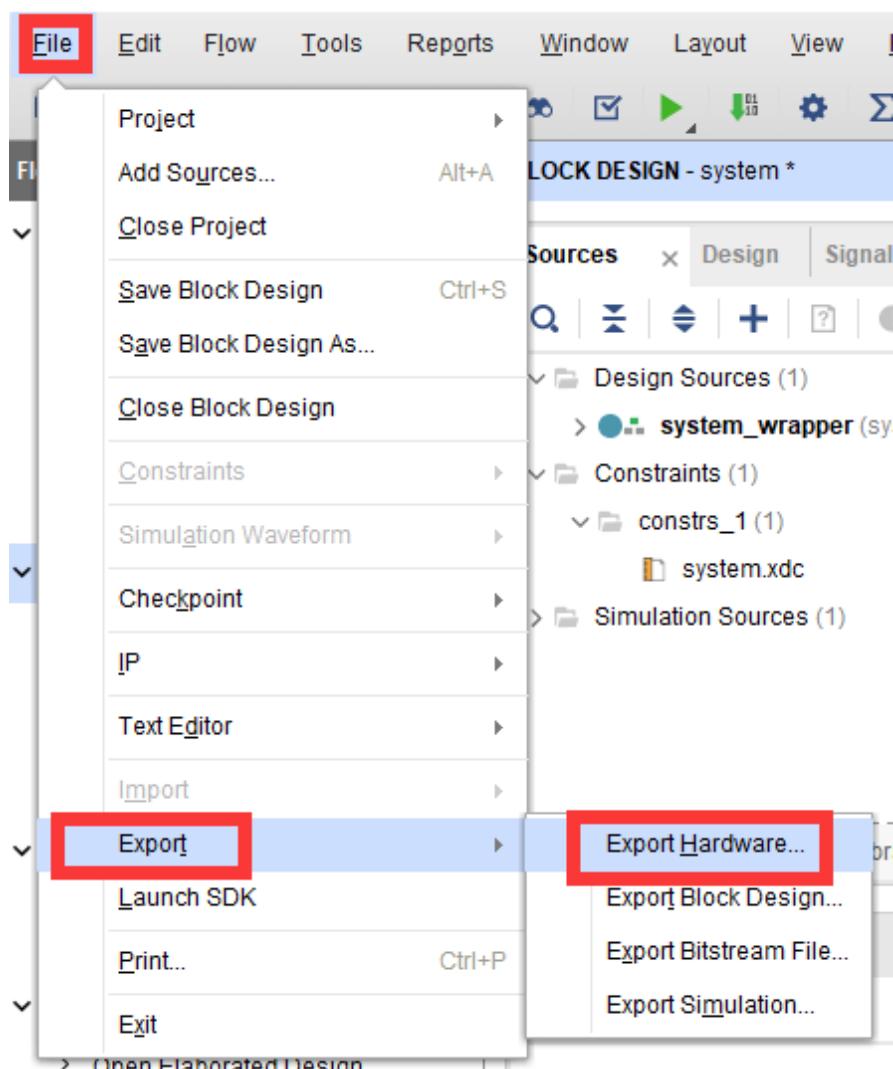


2. 当 Bitstream 生成完成后，弹出 Bitstream Generation Completed 窗口，单击 cancel 完成工程搭建。

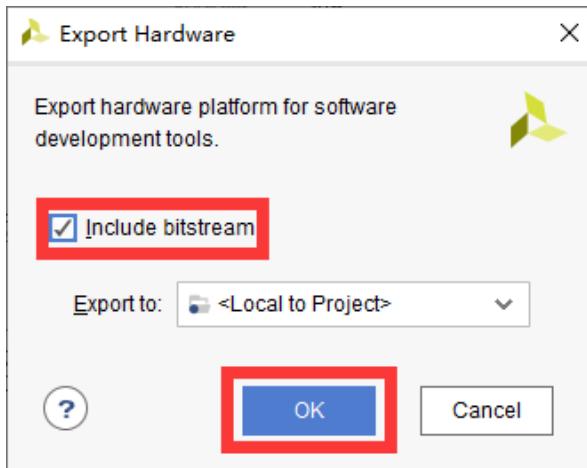


导出工程

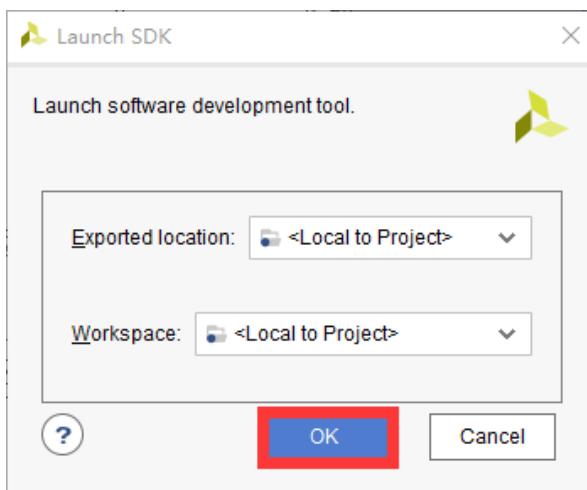
1. 单击 File > Export > Export Hardware...。



2. 在弹出窗口勾选 `Include bitstream`，并单击 `OK`，如果提醒 `Save project before exporting?`，单击 `Save`。

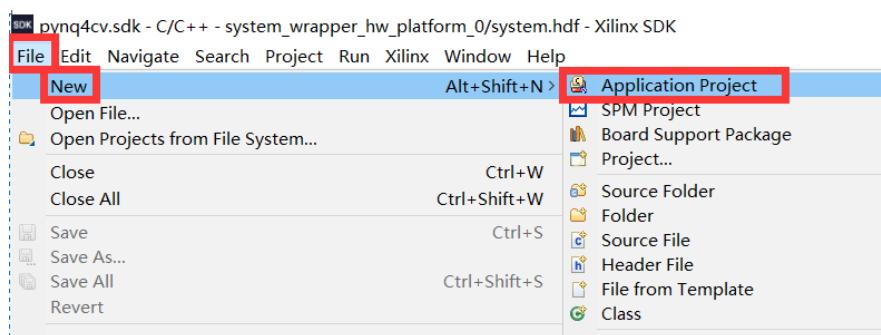


3. 单击 File > Launch SDK , 在弹出窗口单击 OK , 等待片刻, 打开 SDK 工作窗口。

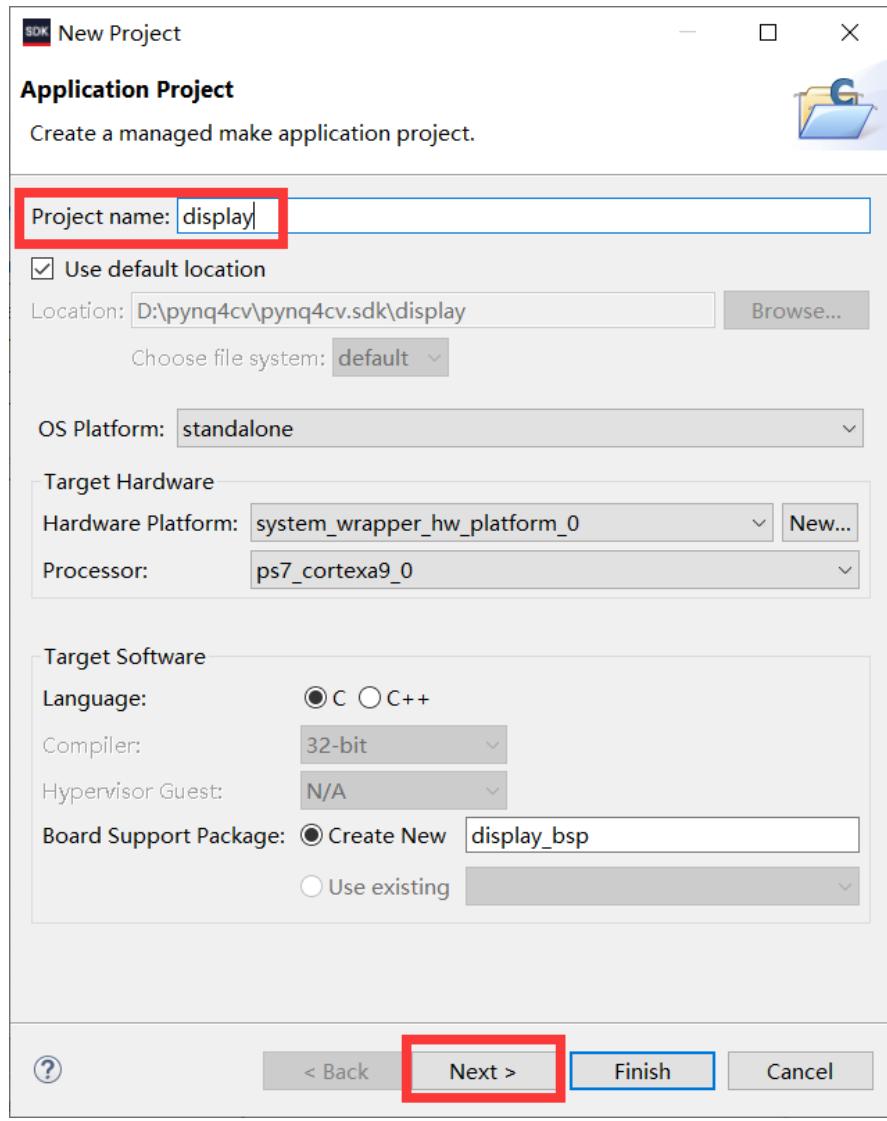


创建应用(Create an SDK Software Project)

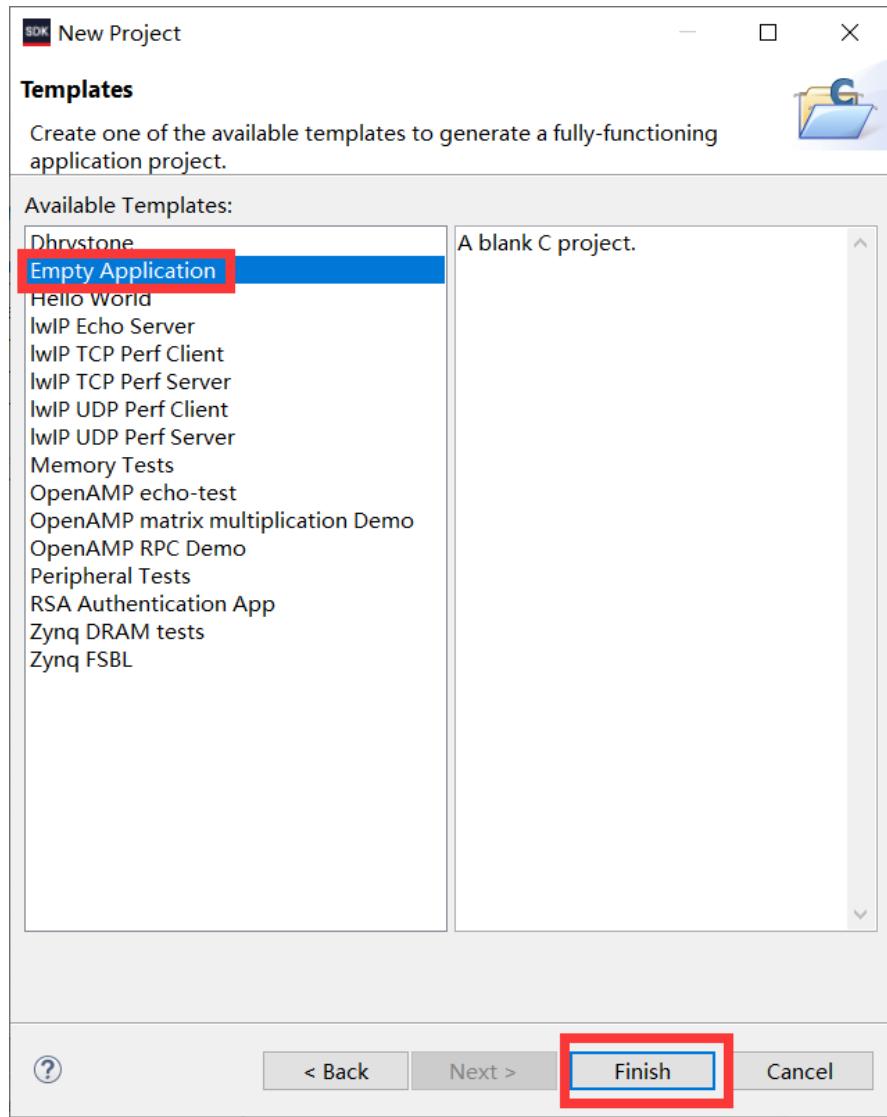
1. 在 SDK 窗口单击 File > New > Application Project。



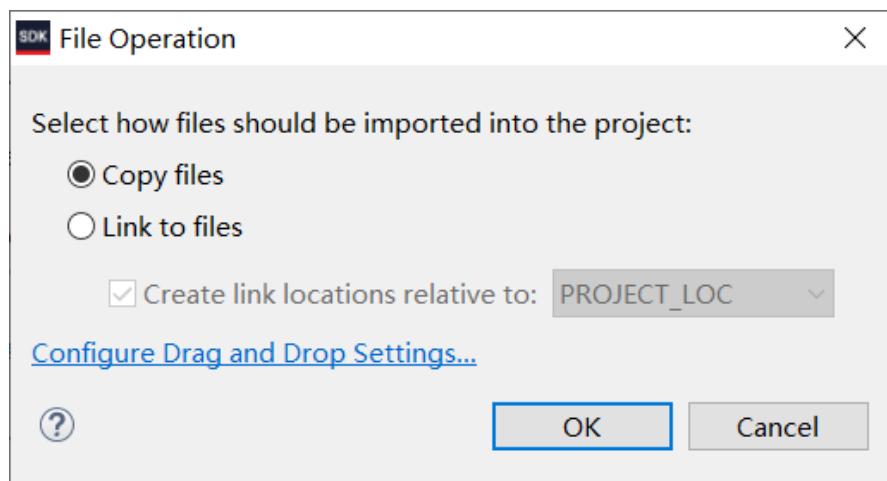
2. 在弹出窗口, 修改 Project name: 为 display , 单击 Next。



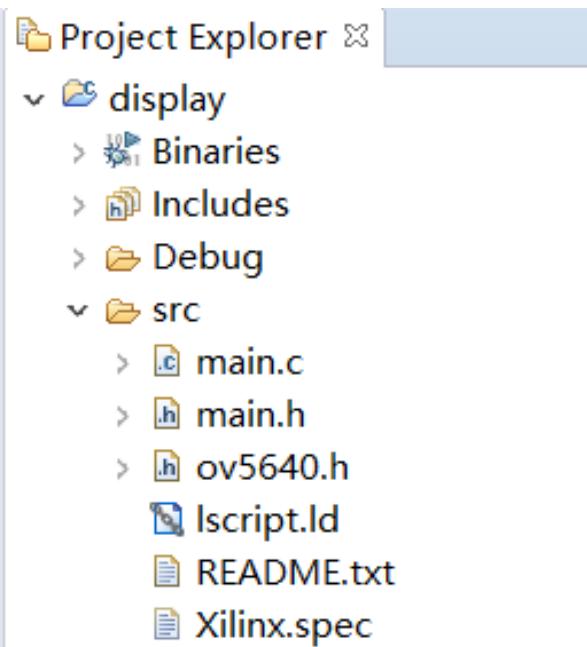
3. 选择 Empty Application，单击 Finish，完成应用创建。



4. 将 {path/to/doc}\files\src 下的 main.c、main.h 和 ov5640.h 拖动到工程左侧栏中
Project Explorer > display > src，在弹出窗口单击 OK。

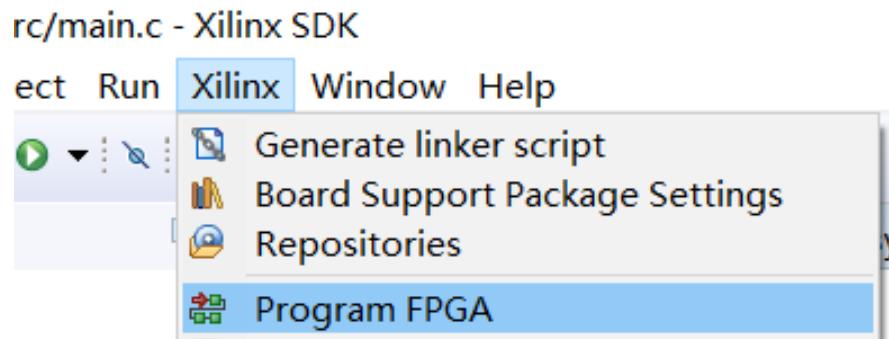


5. 此时左侧栏中会出现相应文件，完成应用创建。



上板测试(Test in Hardware)

1. 连接开发板，单击 `xilinx > Program FPGA`，在弹出窗口单击 `Program`。



2. 右键单击 `Project Explorer > display`，选择 `Run As > Launch on Hardware(System Debugger)`。

