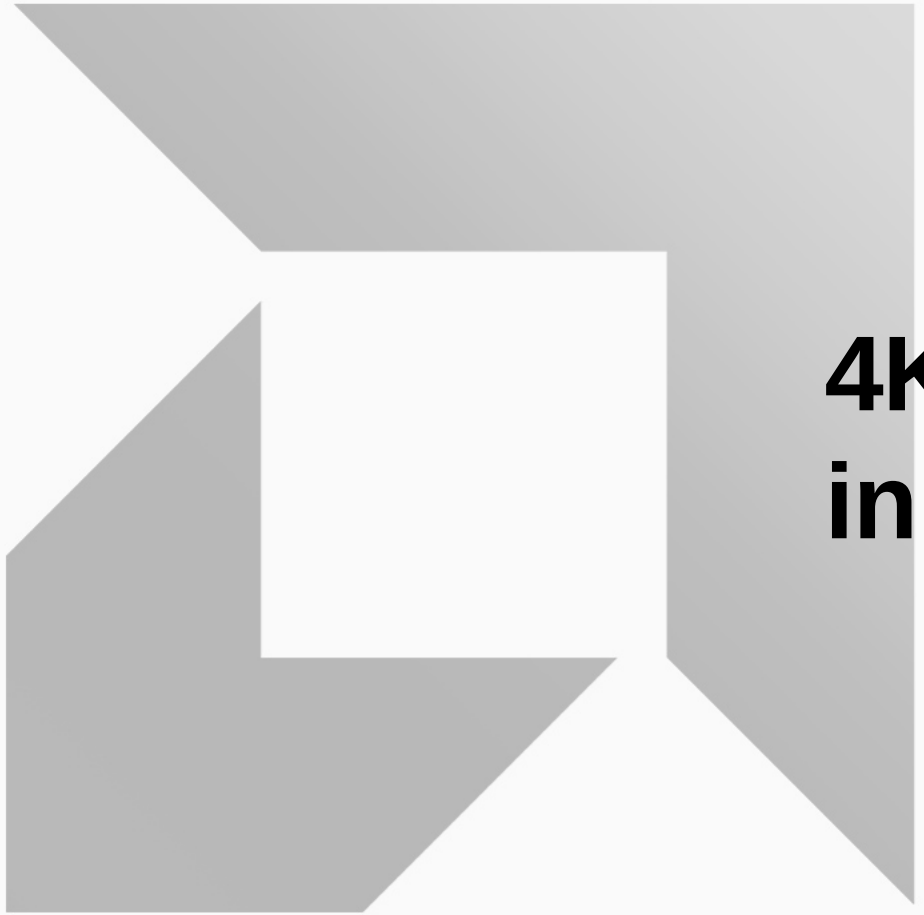
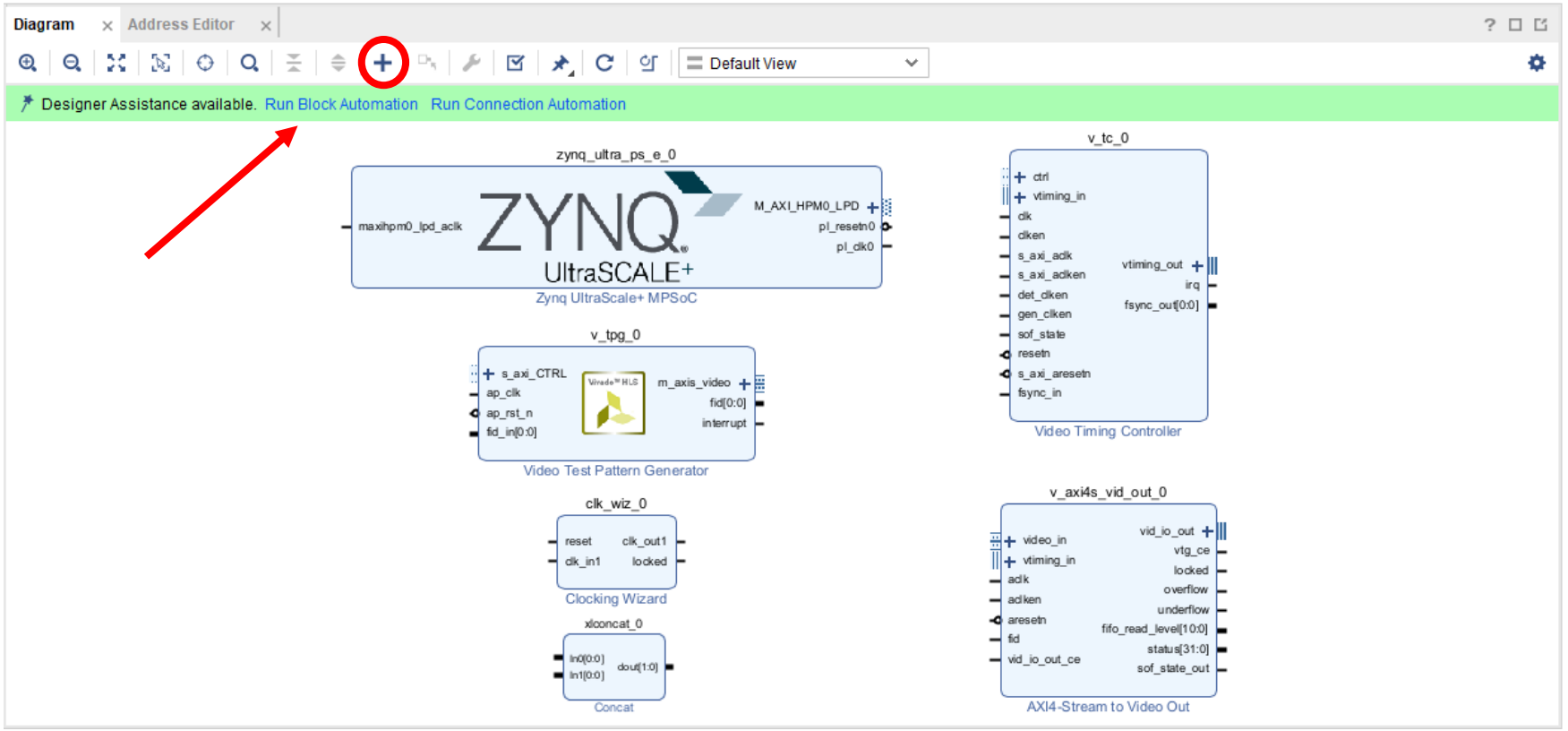


4K TPG Video Streaming in ZCU104

AMD
together we advance_

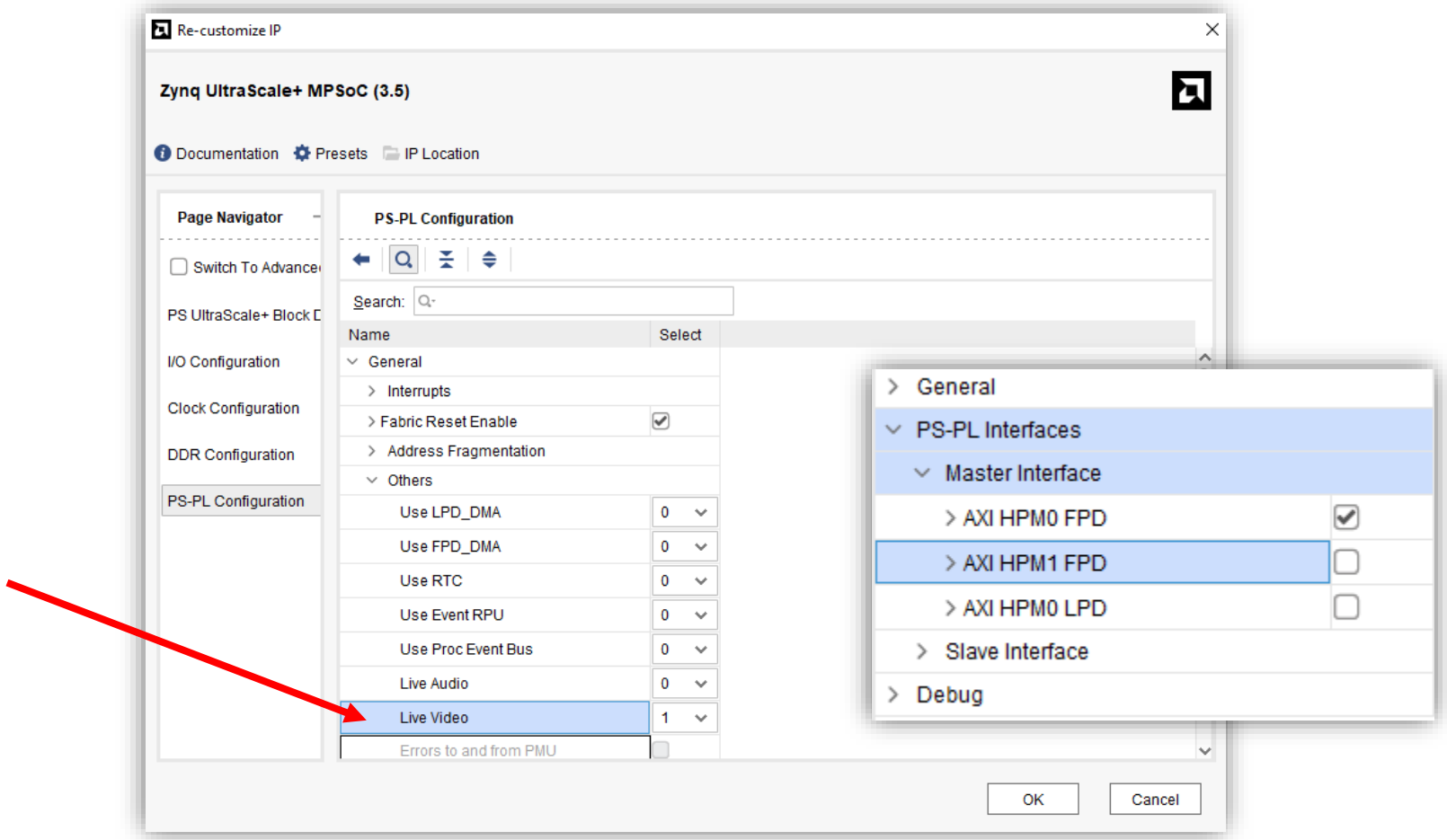
Vivado IP Integrator(VIPI)

- Zynq UltraScale+ MPSoC Processing System (PS)
- Video Test Pattern Generator (TPG)
- Clocking Wizard
- Video Timing Controller (VTC)
- AXI4-Stream to Video Out
- Concat



Zynq UltraScale+ MPSoC PS

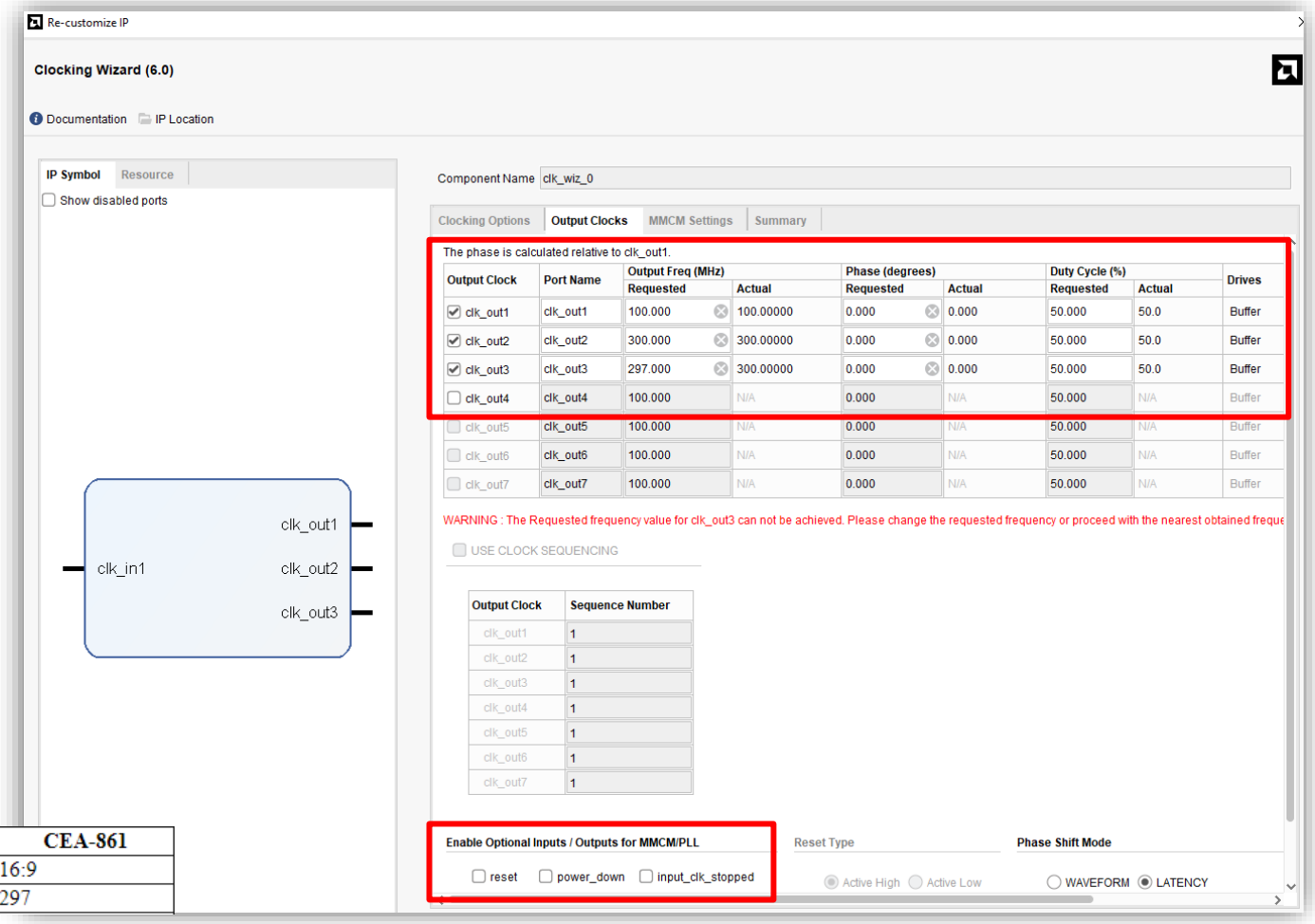
KV260 Display Port (DP) is coming out from the PS side, not from the PL side.
We have to enable Live Mode in PS. This will add a parallel video interface in the PS IP block.
So, we can connect the PL video interface to the PS video interface.



Clocking Wizard

In our design, the IP is configured to generate three clocks as follows:

- clk_out1 - 100MHz for Axilite interface
- clk_300 - 300MHz for AXI Stream
- clk_297 - 297MHz for Video clock



Input Parameters

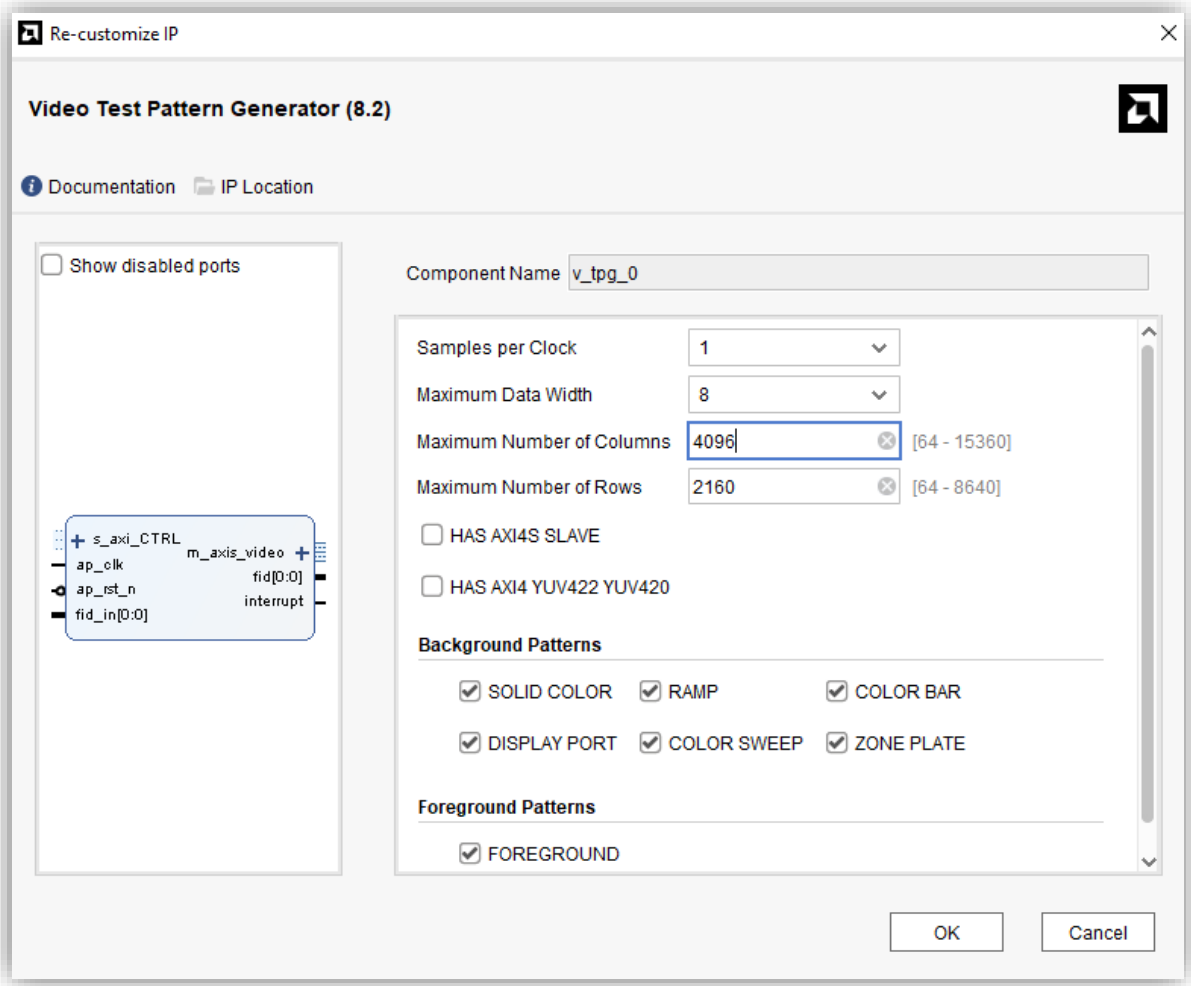
Predefined Mode	4K / 60	
Horizontal Pixels	3840	Ok
Vertical Pixels	2160	Ok
Refresh Rate (Hz)	30	Ok
Margins	No	
Interlaced	No	
Bits per Component	12	
Color Format	RGB 4:4:4	
Video Optimized	No	

	CVT	CVT-RB	CVT-RBv2	CEA-861
Aspect Ratio	16:9	16:9	16:9	16:9
Pixel Clock	338.75	262.75	257.661	297

Video Test Pattern Generator (VTPG)

Use Xilinx's VTPG IP as our pipeline video source.

To support 4K resolution, we must have
column and row values be 4096 and 2160 respectively.



Video Timing Controller (VTC)

This IP is responsible for generating as well as detecting the video timing regarding video resolution.

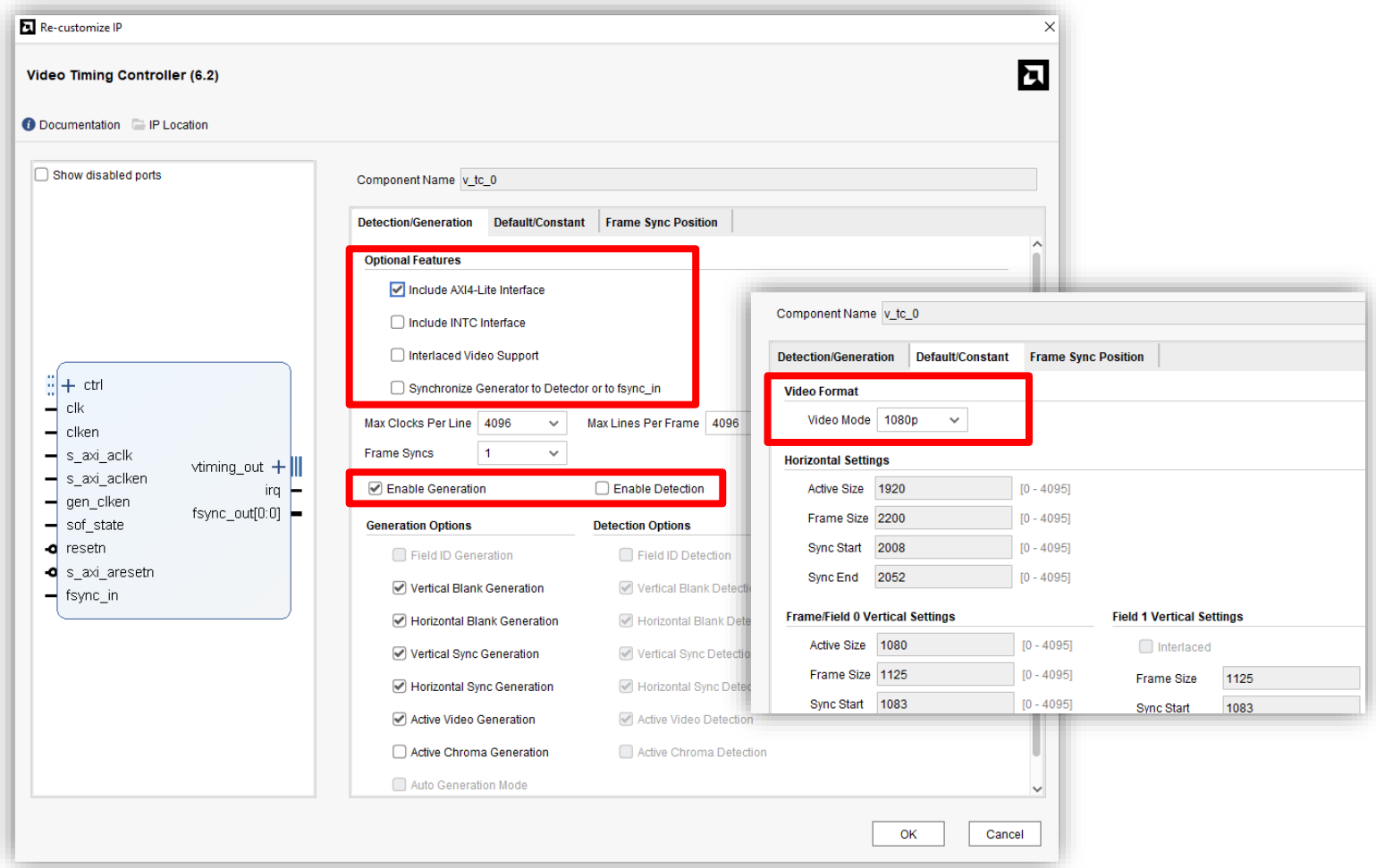
Here in this design,

- We need to generate the video timing.

So, IP is customized for a generation.

- Select custom mode to manually set the video timing for custom resolution.

- Enabled the AXI Lite interface so that we can program this IP to generate video timing for 4K resolution.



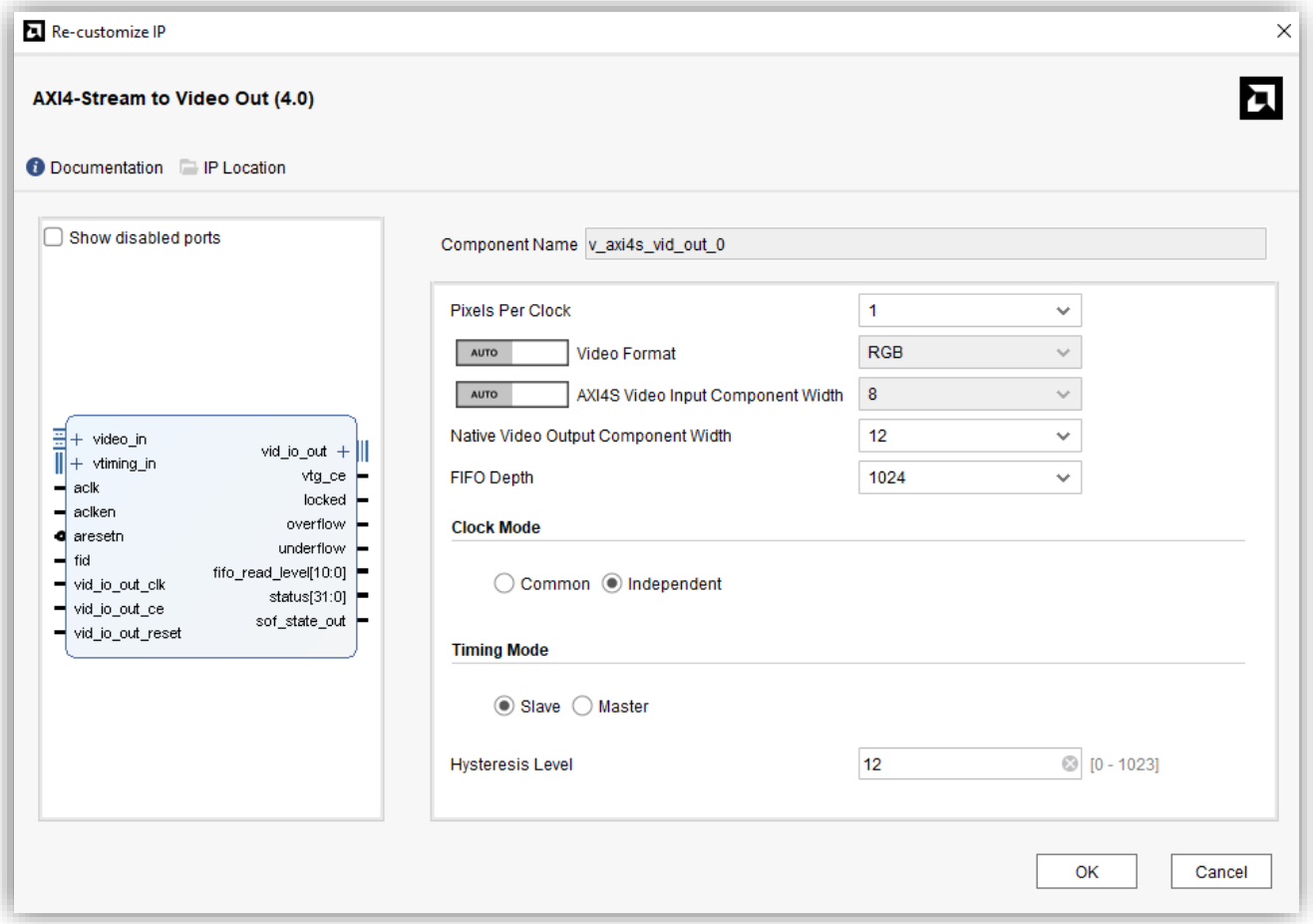
AXI4-Stream to Video Out (Axis2Video)

This IP takes AXI4 Stream and Video Timing as Input and converts it into Parallel Video.
We can then connect this video to the DP video interface in the Zynq PS block.

As DP video data width is 36 bit,
we also need to set parallel video data width to 36 bit by setting Native Video Output Component Width 12.

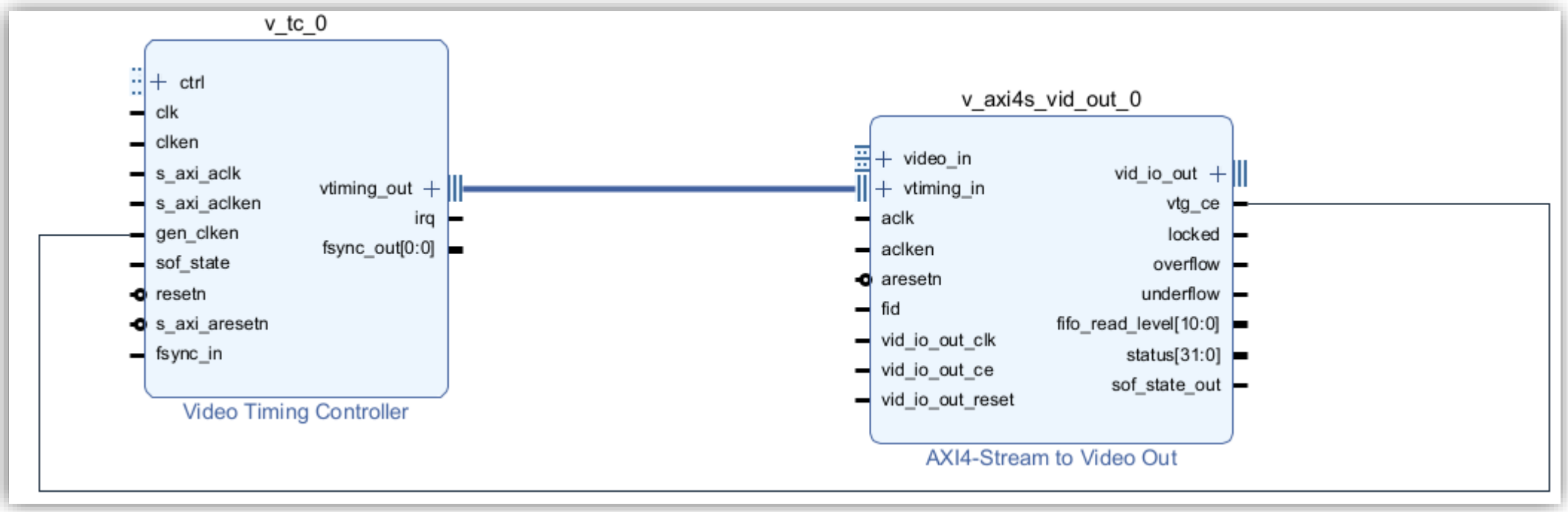
We also set clock mode Independent
so that we can give separate video clock.

We let other parameters as it is.



AXI4-Stream to Video Out (Axis2Video)

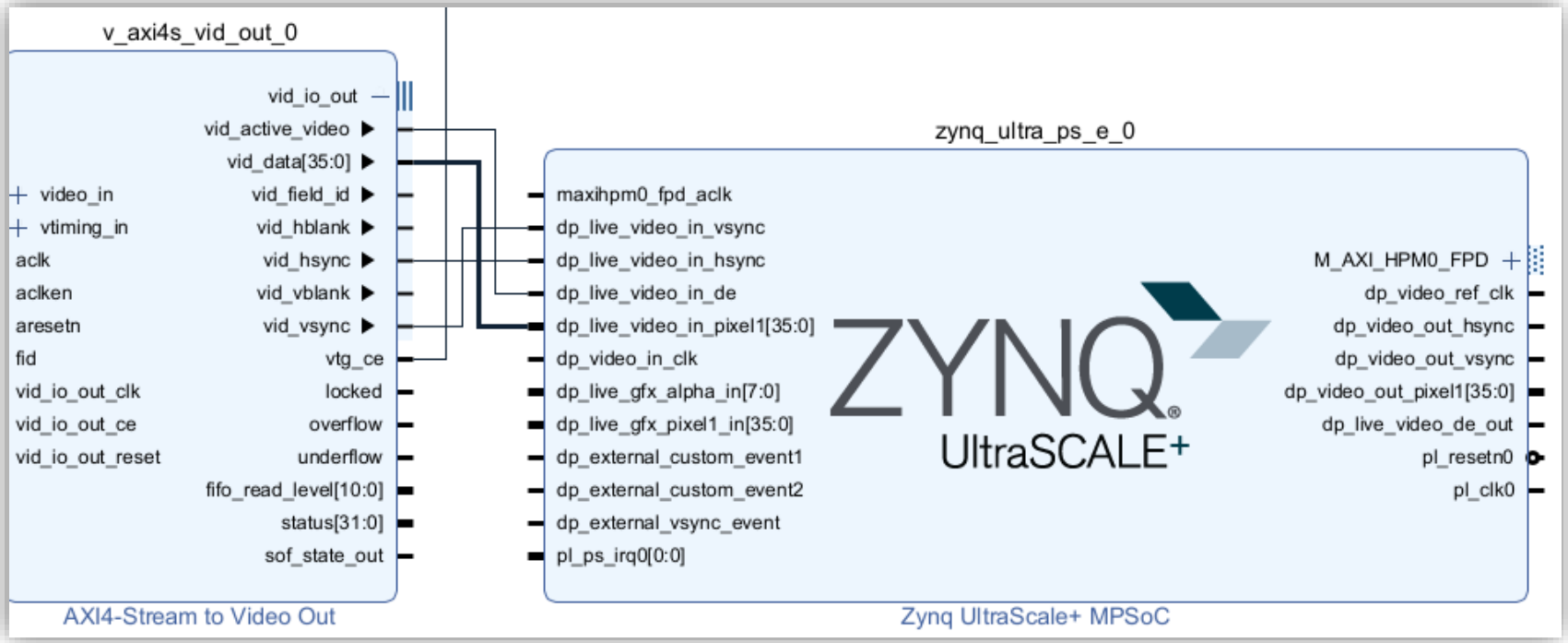
The timing generation is controlled by Axis2Video for the synchronization between the stream and its timing. So, we need to connect VTC's gen_clken pin to Axis2Video's vtg_ce pin,
And connect vtiming_out pin to vtiming_in pin



AXI4-Stream to Video Out (Axis2Video)

Axis2Video's parallel video interface

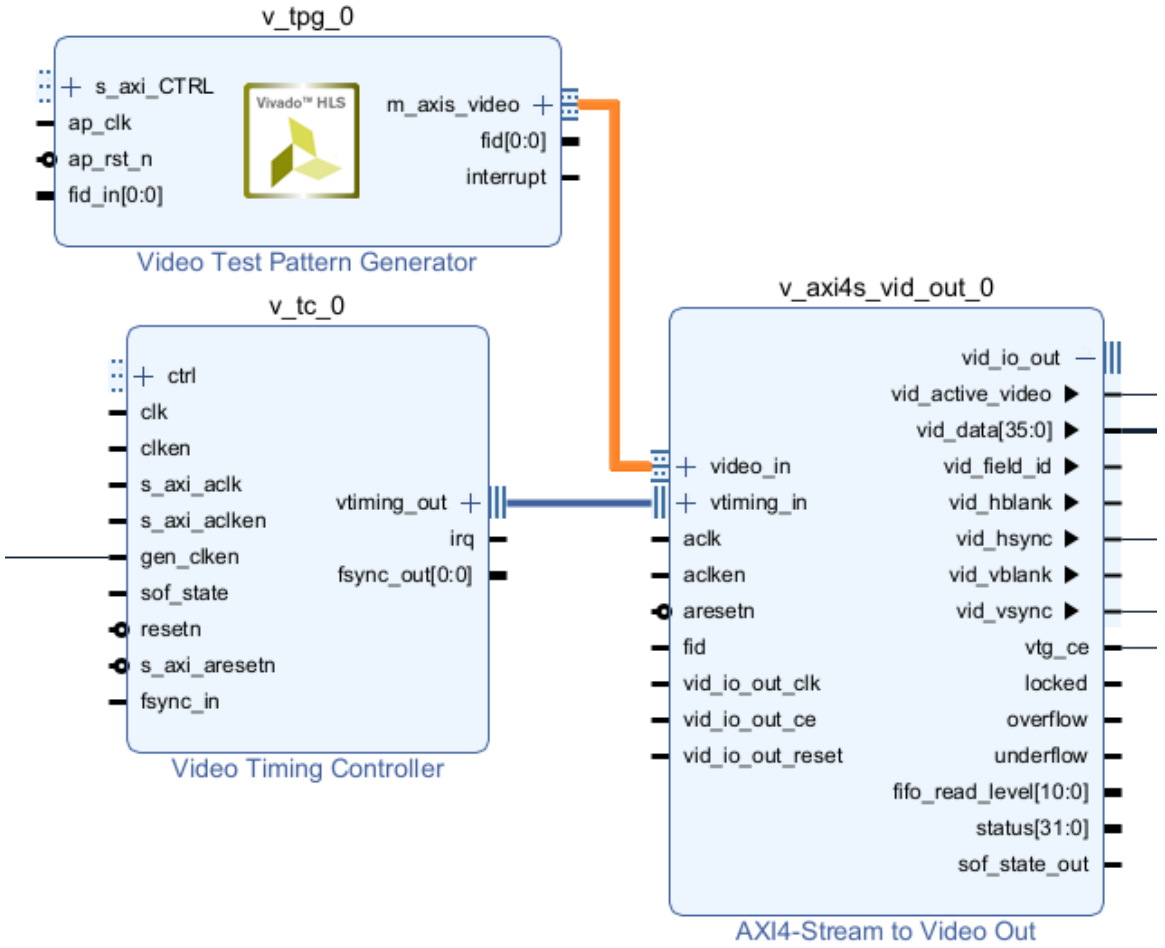
(Axis2Video) (Zynq U+ MPSoC PS)
vid_active video ------> dp_live_video_in_de
vid_data ----------------> dp_live_video_in_pixel1[35:0]
vid_hsync----------------> dp_live_video_in_hsync
vid_vsync----------------> dp_live_video_in_vsync



AXI Stream and Clock Connections

We need to establish the AXI4Stream interface connection between **VTPG** and **Axis2Video** IPs to complete a pipeline. Let's connect interface as shown below:

m_axis_video (VTPG) -------> **video_in** (Axis2Video)



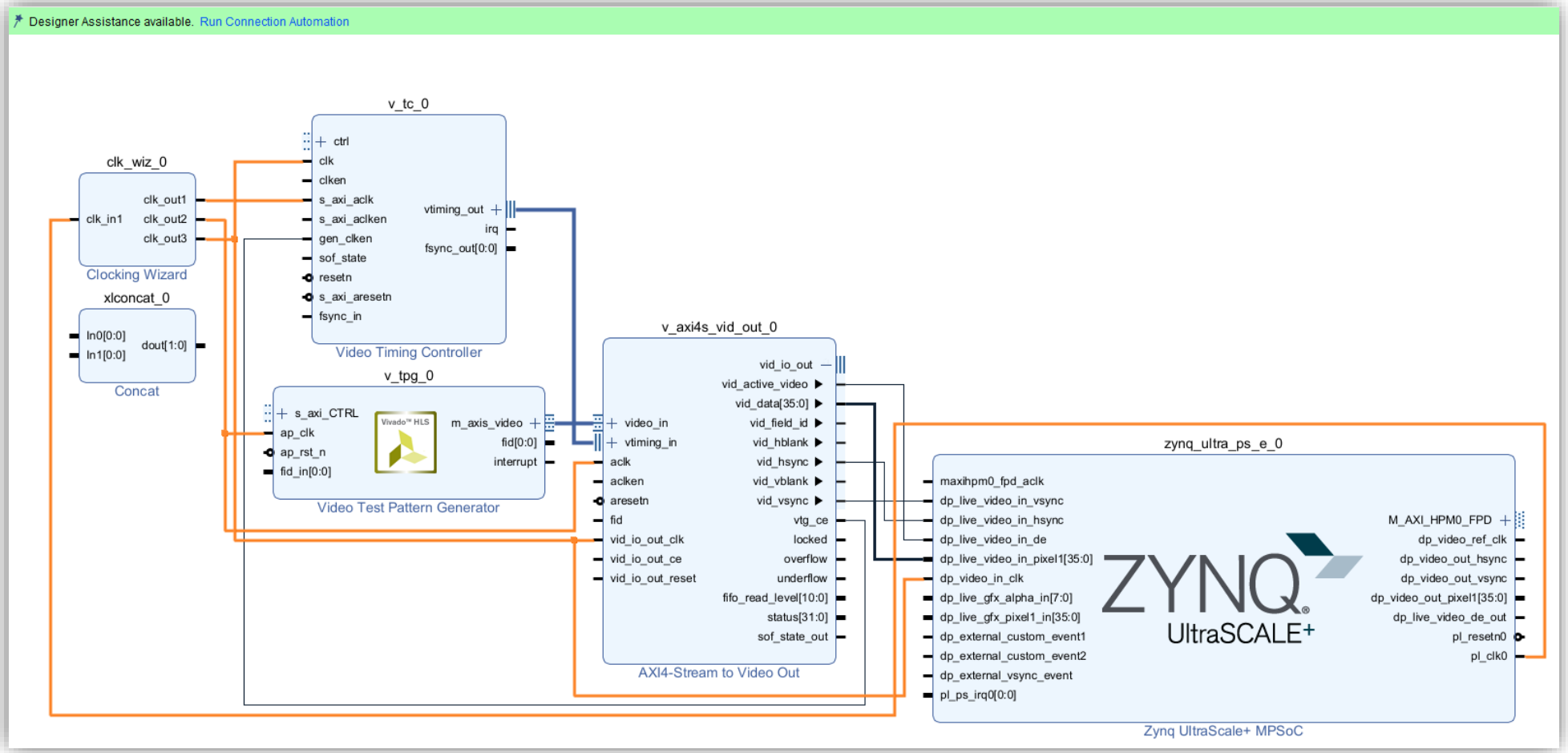
AXI Stream and Clock Connections

As we are generating three clock outputs from the clock wizard, we need to connect these in the following manner.

clk_wiz clock	IP clock pin	IP
clk_out1	s_axi_aclk	VTC
clk_300	ap_clk	VTPG
	aclk	Axis2Video
clk_297	clk	VTC
	vid_io_out_clk	Axis2Video
	dp_video_in_clk	Zynq U+ MPSoC

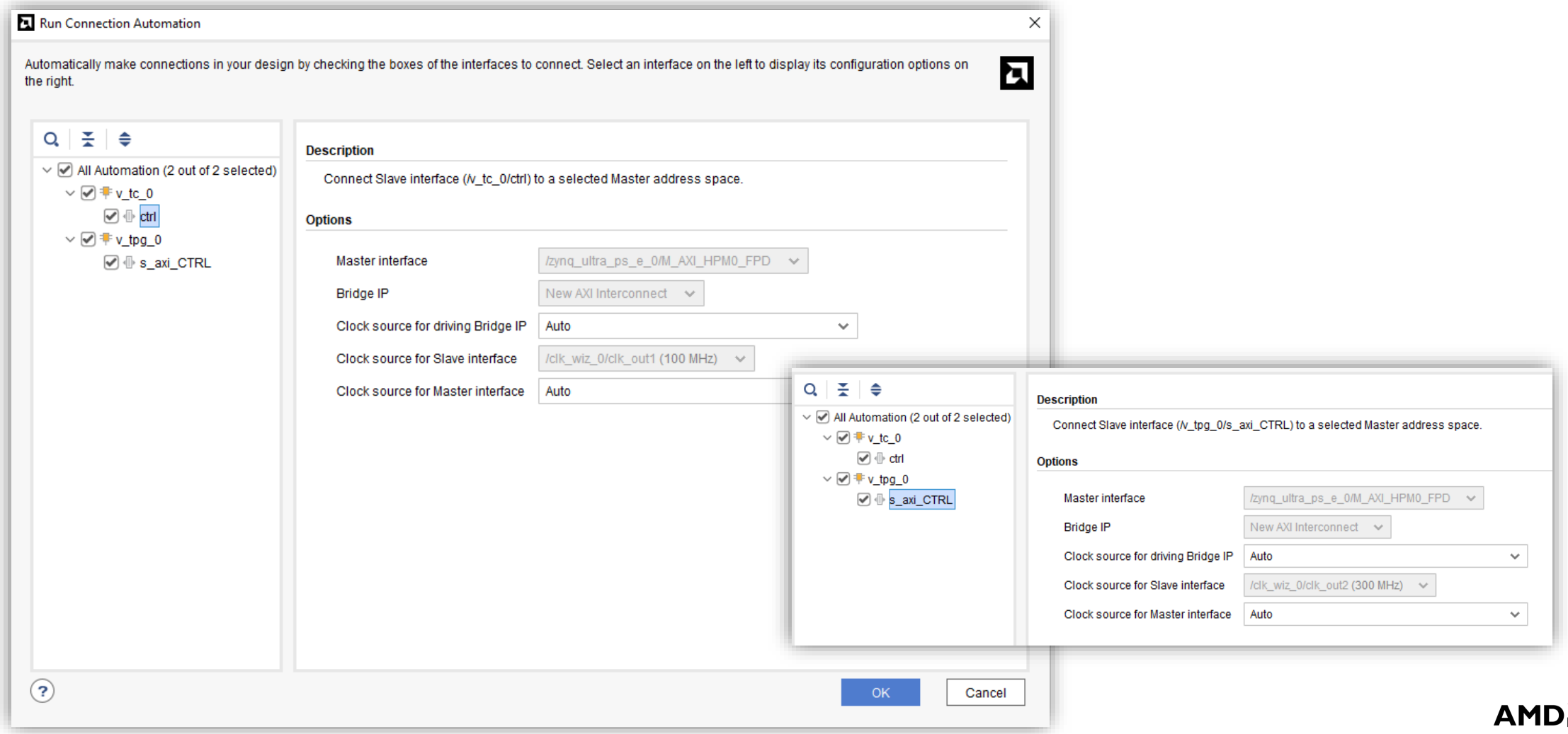
AXI Stream and Clock Connections

As we are generating three clock outputs from the clock wizard, we need to connect these in the following manner.



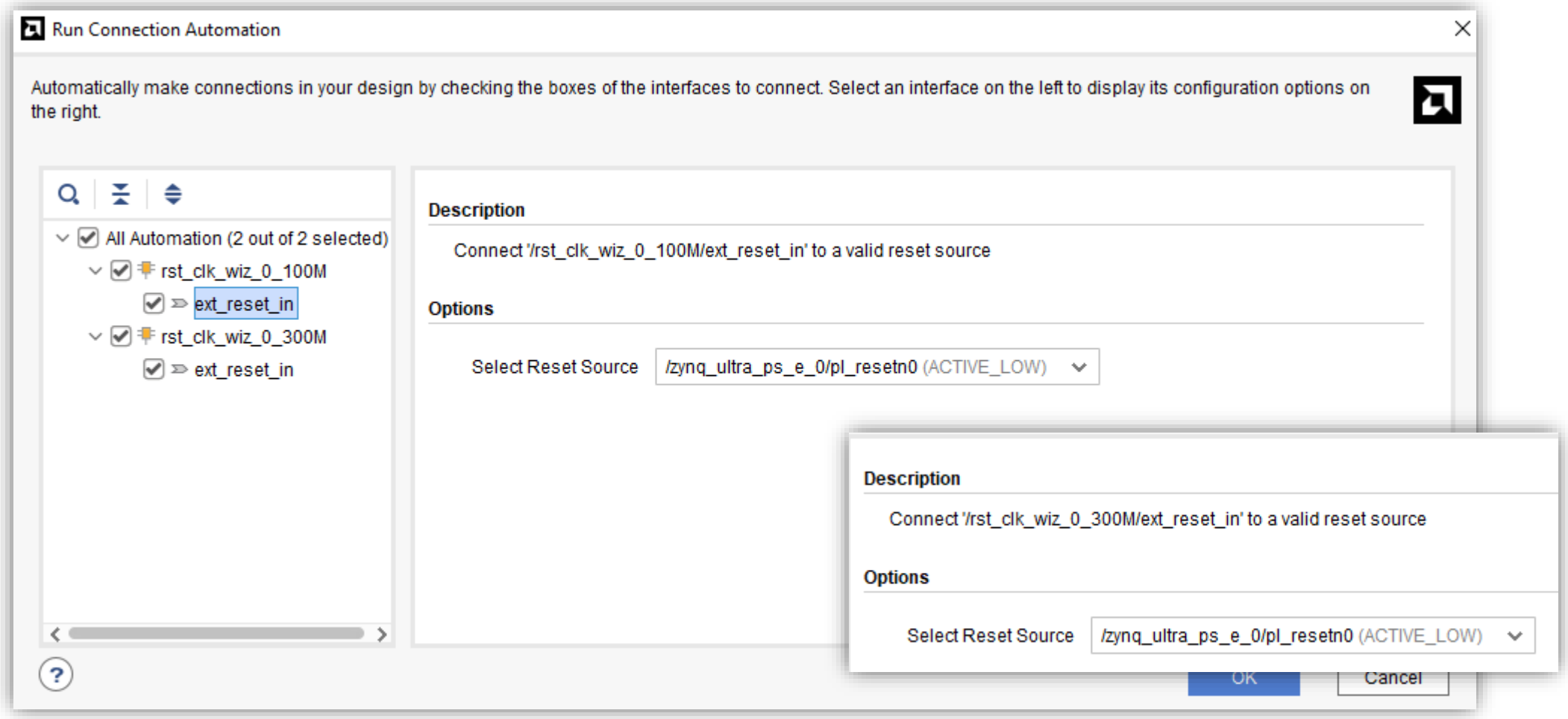
Run Connection Automation

Finally, let's run connection automation to connect all AXI Lite interfaces of IPs to Master AXI Port in the Zynq PS block. Vivado will automatically add AXI Interconnect IP and do the necessary clock and reset pin connection for it.

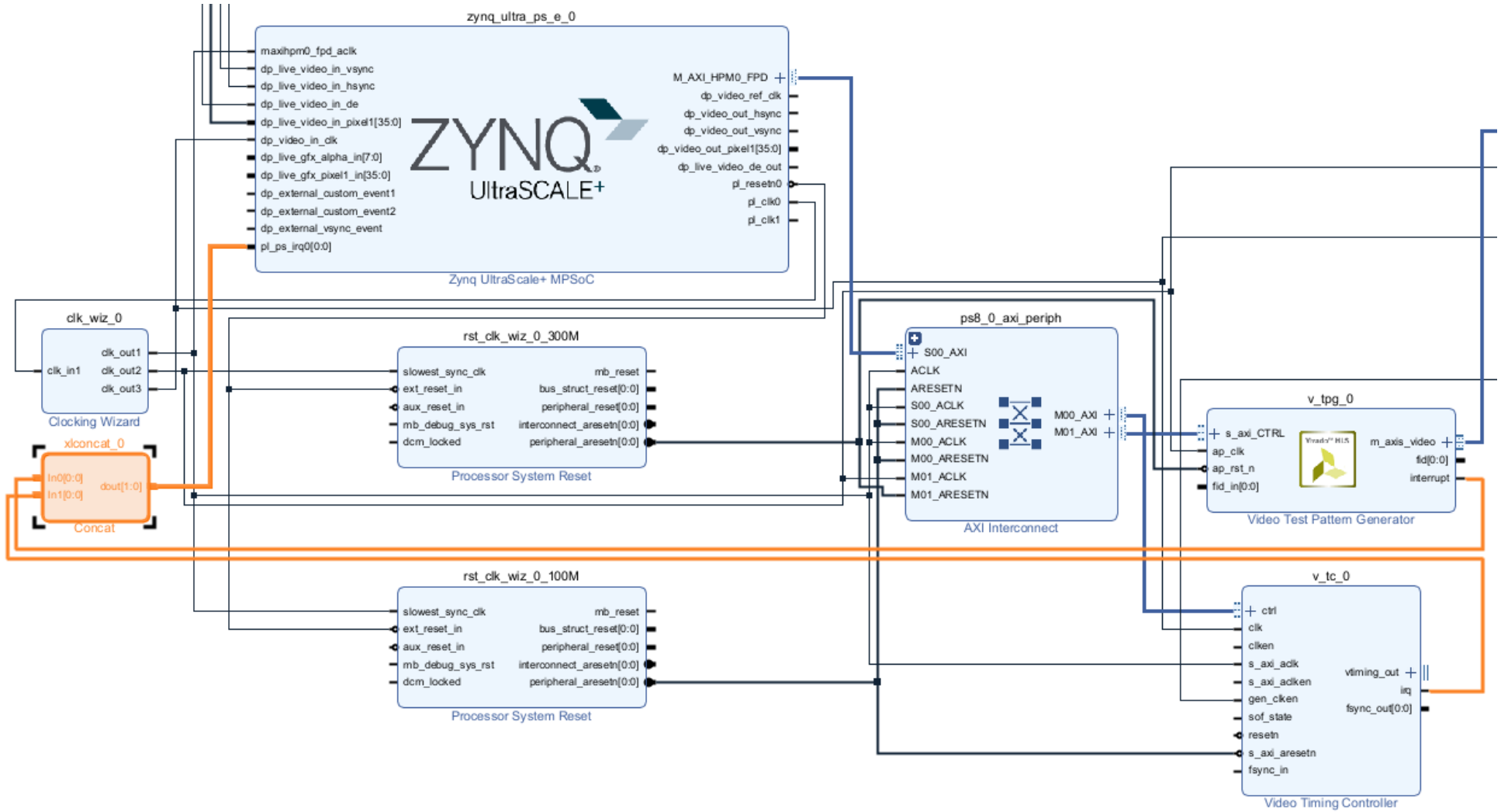


Run Connection Automation

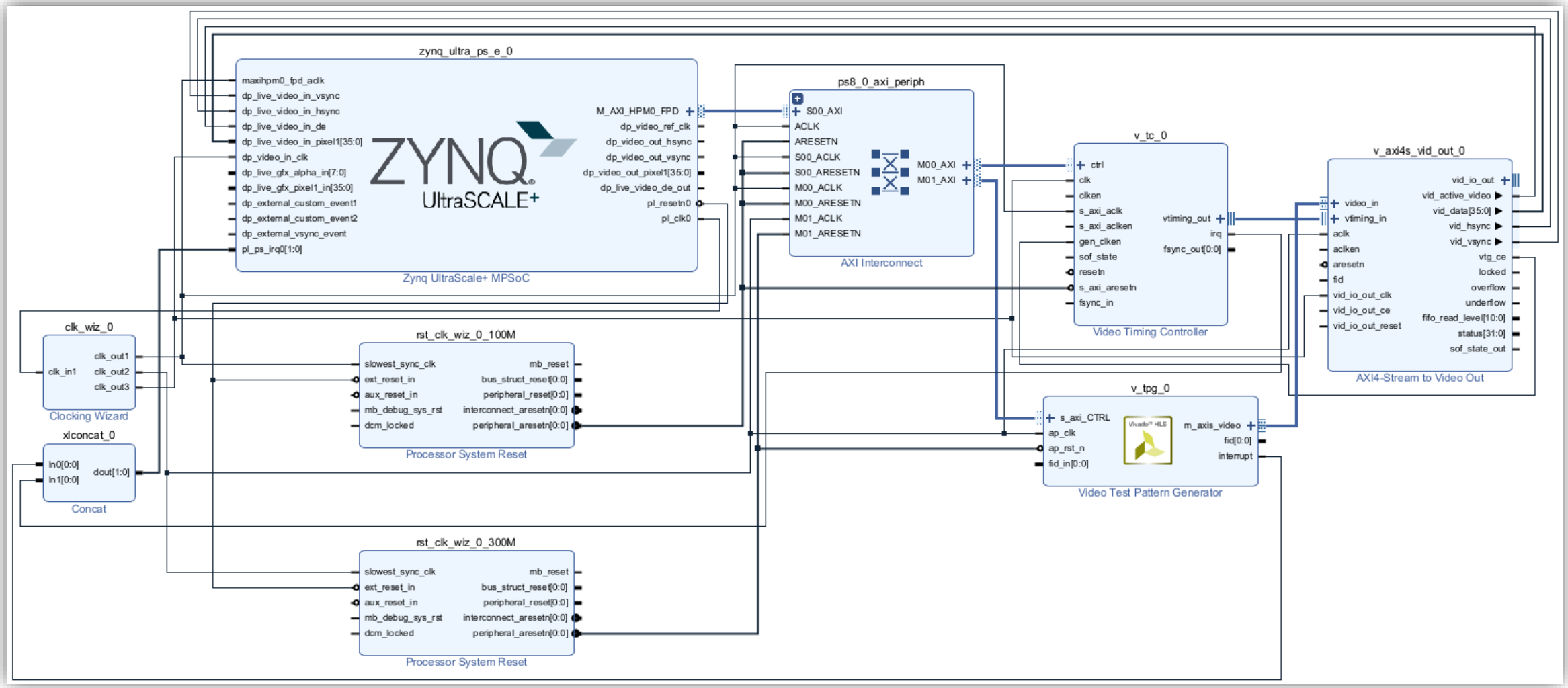
As you might have noticed the processor System Reset IPs are also automatically added. We need to connect the ext_reset_in pin to the pl_resetn pin of the Zynq PS block.



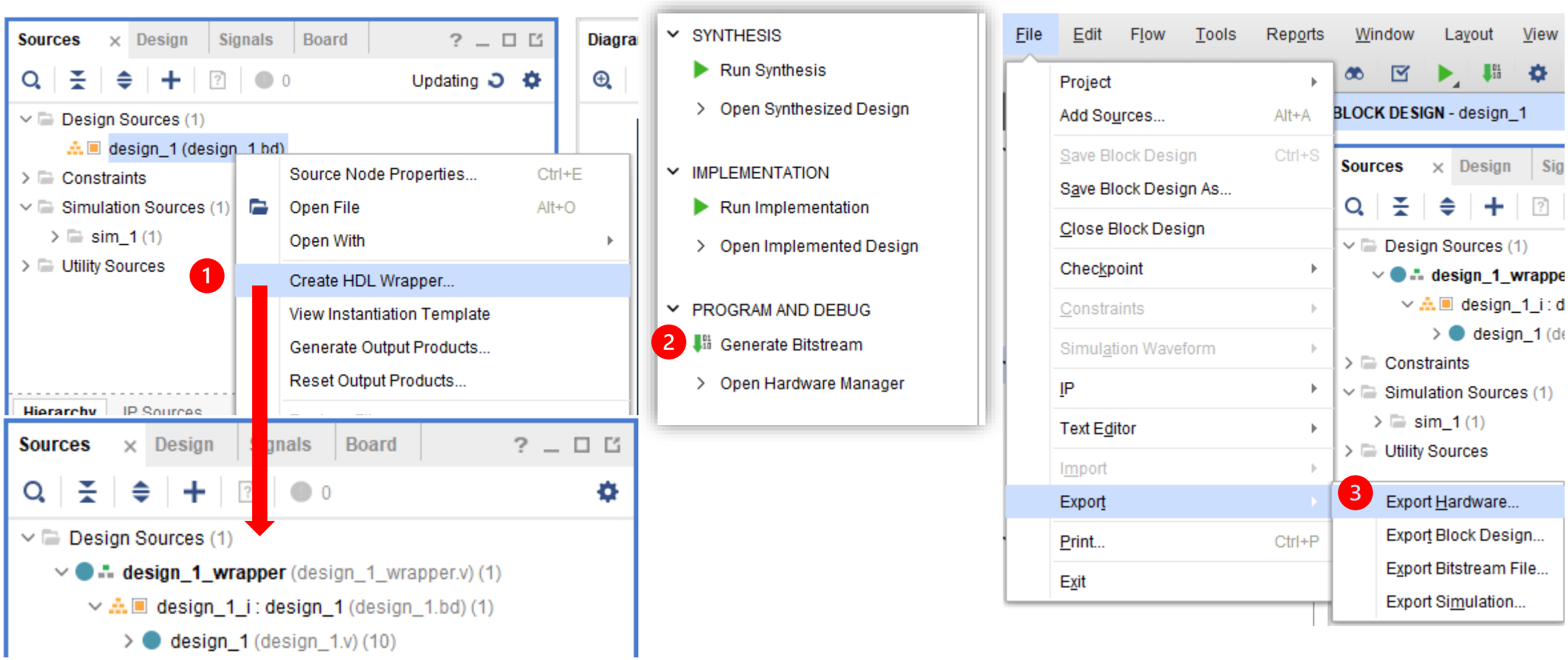
PL_PS Interrupt



Final Block Design



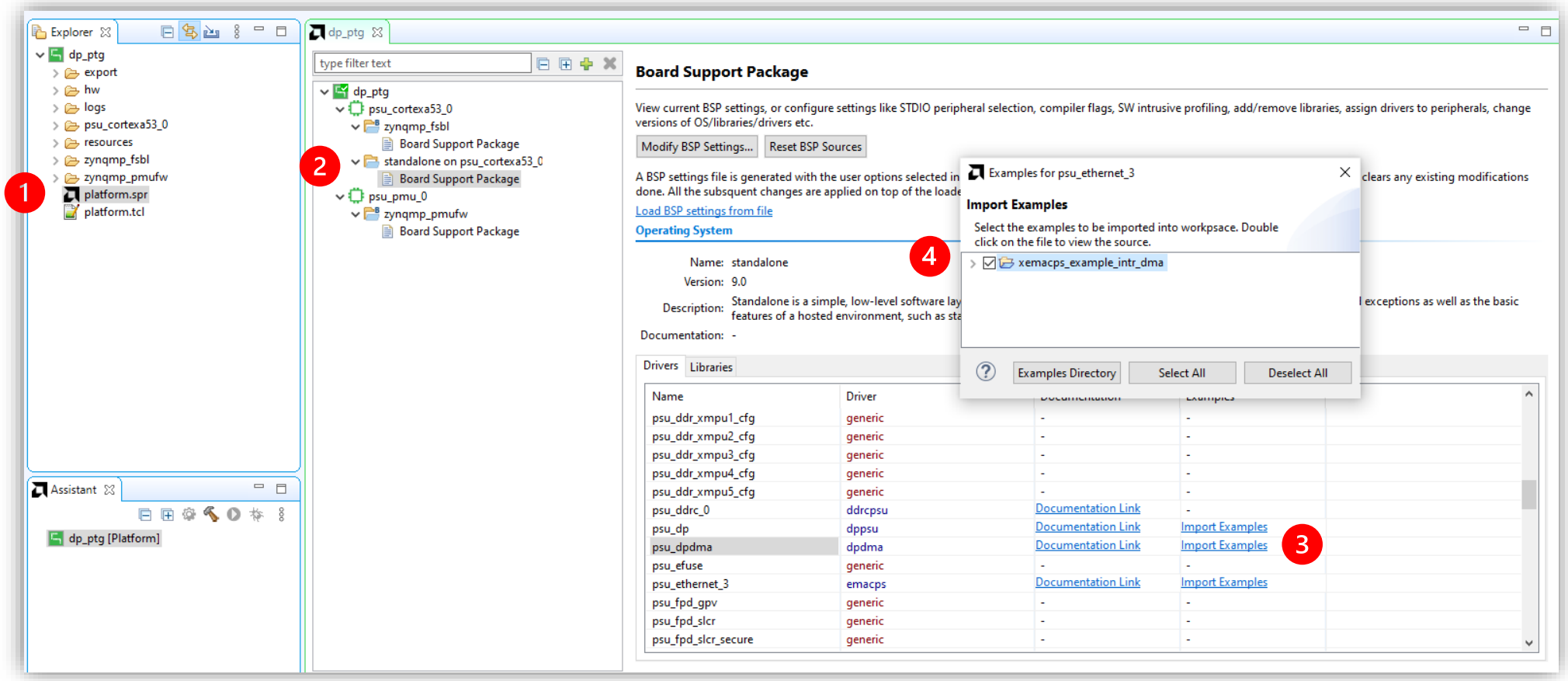
Export Xsa to Vitis



Vitis Part

Create Platform & Import Examples

Xilinx has provided software API for DP interface. Users can quickly implement the DP interface in their design. You can find the API code and import it by following the steps shown in the pictures below.



Example Design

xdpdma_video_example.c

```
38  /************************** Constant Definitions *****************************/
39  #ifndef SDT
40  #define DPPSU_DEVICE_ID     XPAR_PSU_DP_DEVICE_ID
41  #define AVBUF_DEVICE_ID     XPAR_PSU_DP_DEVICE_ID
42  #define DPDMA_DEVICE_ID     XPAR_XDPDMA_0_DEVICE_ID
43  #define INTC_DEVICE_ID      XPAR_SCUGIC_0_DEVICE_ID
44  #define DPPSU_INTR_ID       151
45  #define DPDMA_INTR_ID       154
46
47  #define DPPSU_BASEADDR      XPAR_PSU_DP_BASEADDR
48  #define AVBUF_BASEADDR      XPAR_PSU_DP_BASEADDR
49  #define DPDMA_BASEADDR      XPAR_PSU_DPDMA_BASEADDR
50  #else
51  #define DPPSU_BASEADDR      XPAR_XDPPSU_0_BASEADDR
52  #define AVBUF_BASEADDR      XPAR_XDPPSU_0_BASEADDR
53  #define DPDMA_BASEADDR      XPAR_XDPDMA_0_BASEADDR
54  #define INTC_BASEADDR       XPAR_XSCUGIC_0_BASEADDR
55  #endif
56
57  #define BUFFERSIZE          3840 * 2160 * 4     /* HTotal * VTotal * BPP */
58  #define LINESIZE            3840 * 4            /* HTotal * BPP */
59  #define STRIDE              LINESIZE            /* The stride value should
60                                                      be aligned to 256*/
```

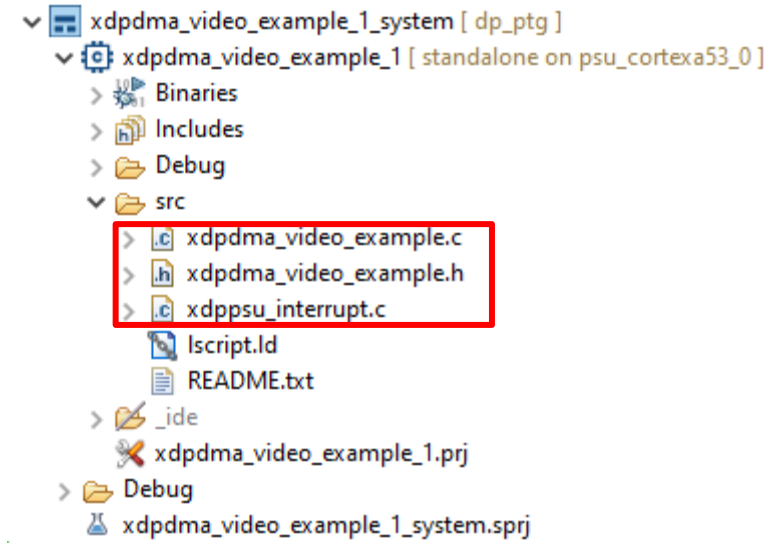
```
143 /*****************************************************************************/
144 /**
145 *
146 * The purpose of this function is to initialize the application configuration.
147 *
148 * @param    RunCfgPtr is a pointer to the application configuration structure.
149 *
150 * @return   None.
151 *
152 * @note     None.
153 *
154 ******************************************************************************/
155 void InitRunConfig(Run_Config *RunCfgPtr)
156 {
157     /* Initial configuration parameters. */
158         RunCfgPtr->DpPsuPtr    = &DpPsu;
159         RunCfgPtr->IntrPtr     = &Intr;
160         RunCfgPtr->AVBufPtr    = &AVBuf;
161         RunCfgPtr->DpDmaPtr    = &DpDma;
162         RunCfgPtr->VideoMode = XVIDC_VM_3840x2160_30_P;
163         RunCfgPtr->Bpc       = XVIDC_BPC_8;
164         RunCfgPtr->ColorEncode          = XDPPSU_CENC_RGB;
165         RunCfgPtr->UseMaxCfgCaps        = 1;
166         RunCfgPtr->LaneCount            = LANE_COUNT_2;
167         RunCfgPtr->LinkRate             = LINK_RATE_540GBPS;
168         RunCfgPtr->EnSynchClkMode       = 0;
169         RunCfgPtr->UseMaxLaneCount      = 1;
170         RunCfgPtr->UseMaxLinkRate       = 1;
171 }
172
173 /*****************************************************************************/
```

```
240     /* Select the Input Video Sources.
241      * Here in this example we are going to demonstrate
242      * graphics overlay over the TPG video.
243      */
244     XAVBuf_InputVideoSelect(AVBufPtr, XAVBUF_VIDSTREAM1_NONE,
245                             XAVBUF_VIDSTREAM2_NONLIVE_GFX);
```

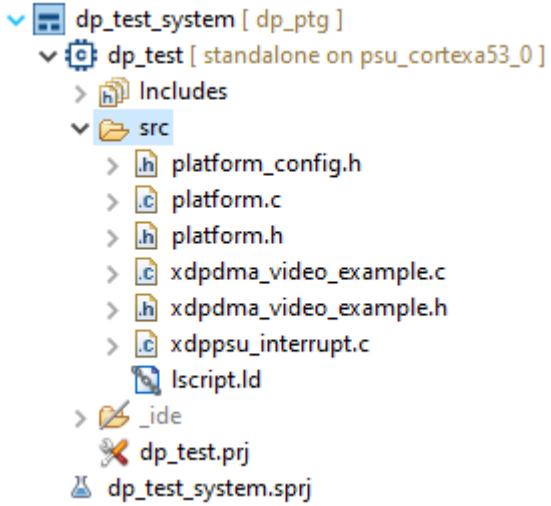
Source Import

Copy these files and paste them into your application project src directory.

Example design



User design



Main.c

```c
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include "xv_tpg.h"
#include "xvtc.h"
#include "xvidc.h"
#include "xdpdma_video_example.h" //change according to your file name

XV_tpg tpg;
XVtc vtc;
XVtc_Config *vtc_config;

void driverInit() {

    XV_tpg_Initialize(&tpg, 0);

    vtc_config = XVtc_LookupConfig(XPAR_VTC_0_DEVICE_ID);

    XVtc_CfgInitialize(&vtc, vtc_config, vtc_config->BaseAddress);

}
void ConfigVtc(XVidC_VideoStream *StreamPtr) {
    XVtc_Timing vtc_timing = { 0 };
    u16 PixelsPerClock = 1;

    vtc_timing.HActiveVideo = StreamPtr->Timing.HActive / PixelsPerClock;
    vtc_timing.HFrontPorch = StreamPtr->Timing.HFrontPorch / PixelsPerClock;
    vtc_timing.HSyncWidth = StreamPtr->Timing.HSyncWidth / PixelsPerClock;
    vtc_timing.HBackPorch = StreamPtr->Timing.HBackPorch / PixelsPerClock;
    vtc_timing.HSyncPolarity = StreamPtr->Timing.HSyncPolarity;
    vtc_timing.VActiveVideo = StreamPtr->Timing.VActive;
    vtc_timing.V0FrontPorch = StreamPtr->Timing.F0PVFrontPorch;
    vtc_timing.V0SyncWidth = StreamPtr->Timing.F0PVSyncWidth;
    vtc_timing.V0BackPorch = StreamPtr->Timing.F0PVBackPorch;
    vtc_timing.VSyncPolarity = StreamPtr->Timing.VSyncPolarity;
    XVtc_SetGeneratorTiming(&vtc, &vtc_timing);
    XVtc_Enable(&vtc);
    XVtc_EnableGenerator(&vtc);
    XVtc_RegUpdateEnable(&vtc);

}
void ConfigTpg(XVidC_VideoStream *StreamPtr) {
    XV_tpg_DisableAutoRestart(&tpg);
    XV_tpg_Set_height(&tpg, StreamPtr->Timing.VActive);
    XV_tpg_Set_width(&tpg, StreamPtr->Timing.HActive);
    XV_tpg_Set_colorFormat(&tpg, XVIDC_CSF_RGB);
    XV_tpg_Set_bckgndId(&tpg, XTPG_BKGND_COLOR_BARS);
    XV_tpg_Set_ovrlayId(&tpg, 1);
    XV_tpg_Set_boxSize(&tpg, 100);
    XV_tpg_Set_motionSpeed(&tpg, 10);
    XV_tpg_EnableAutoRestart(&tpg);
    XV_tpg_Start(&tpg);
}
int main() {
    init_platform();
    print("-----------------------------------\n\r");
    print("----------ZCU104 4K TPG-----------\n\r");
    print("-----------------------------------\n\r");

    XVidC_VideoTiming const *TimingPtr;
    XVidC_VideoMode TestModes[2] = { XVIDC_VM_3840x2160_30_P, XVIDC_VM_UHD_30_P };
    XVidC_VideoStream VidStream;

    /*Set stream parameters*/
    VidStream.PixPerClk = tpg.Config.PixPerClk;
    VidStream.ColorFormatId = XVIDC_CSF_RGB;
    VidStream.ColorDepth = tpg.Config.MaxDataWidth;
    VidStream.VmId = TestModes[0];
    TimingPtr = XVidC_GetTimingInfo(VidStream.VmId);
    VidStream.Timing = *TimingPtr;
    VidStream.FrameRate = XVidC_GetFrameRate(VidStream.VmId);

    xil_printf("\r\n*****************************************\r\n");
    xil_printf("Test Input Stream: %s (%s)\r\n",
            XVidC_GetVideoModeStr(VidStream.VmId),
            XVidC_GetColorFormatStr(VidStream.ColorFormatId));
    xil_printf("*****************************************\r\n");

    driverInit();

    ConfigTpg(&VidStream);

    ConfigVtc(&VidStream);

    run_dppsu();

    cleanup_platform();
    return 0;
}
```

ZCU104 Hardware Setting

ZCU104(MPSoC) DP TPG Output Example

```
Zynq MP First Stage Boot Loader
Release 2023.2   Feb 22 2024  -  11:05:58
PMU-FW is not running, certain applications may not be supported.
--------------------------------
----------KV260 4K TPG-----------
--------------------------------

*****************************************
Test Input Stream: 3840x2160@30Hz (RGB)
*****************************************
DPDMA Generic Video Example Test
Generating Overlay.....
HPD event .......... ! Connected.
Lane count =    2
Link rate =     20

Starting Training...
        ! Training succeeded.
AVBuf Input Ref Clk = 3333333333 HzAVBuf Input Ref Clk = 3333333333 HzDONE!
.......... HPD event
Successfully ran DPDMA Video Example Test
```

ZCU104(MPSoC) DP TPG Output Example