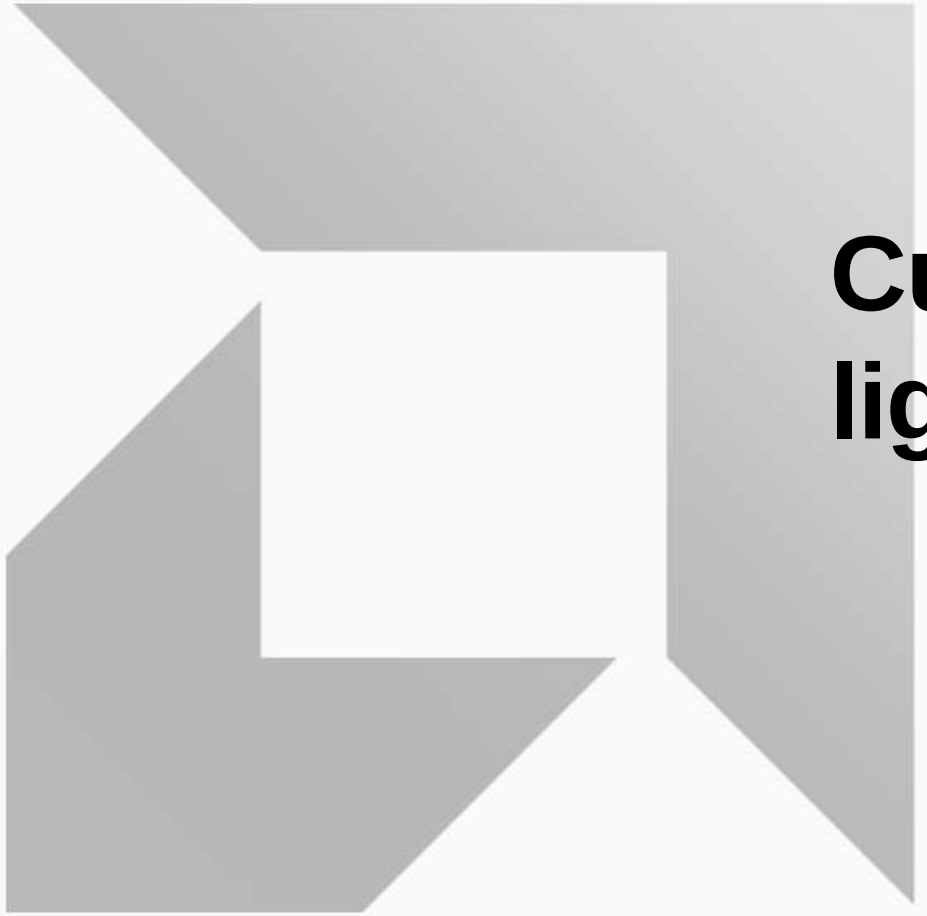


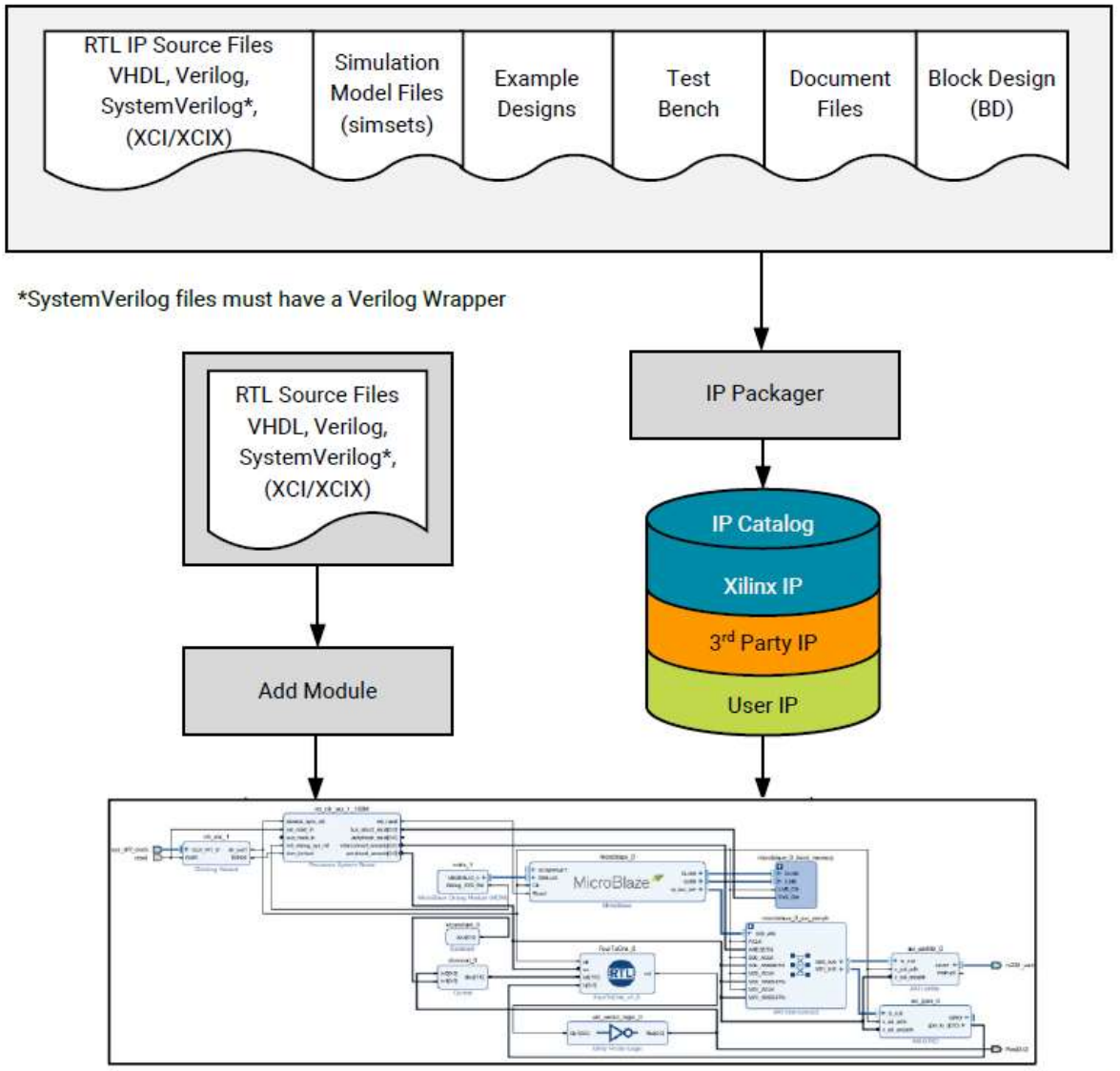
Custom IP LED breathing light

AMD
together we advance_

Agenda

1. Custom IP inturdction
2. Program code design

Figure 1: **IP Packaging and Usage Flow**



- **AXI4:** For memory-mapped interfaces, which allows burst of up to 256 data transfer cycles with a single address phase.
- **AXI4-Lite:** A light-weight, single transaction memory-mapped interface.
- **AXI4-Stream:** For high-speed streaming data.

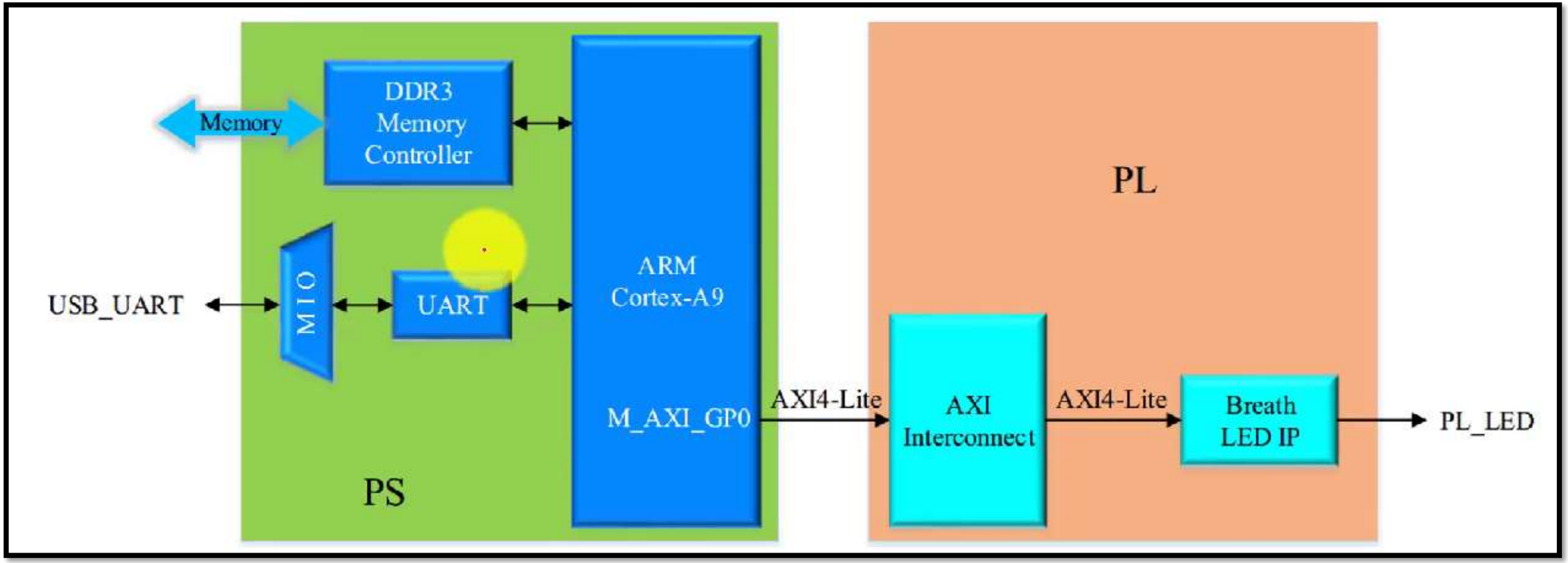
For more information on the Xilinx adoption of AXI, see the *Vivado Design Suite: AXI Reference Guide* (UG1037).

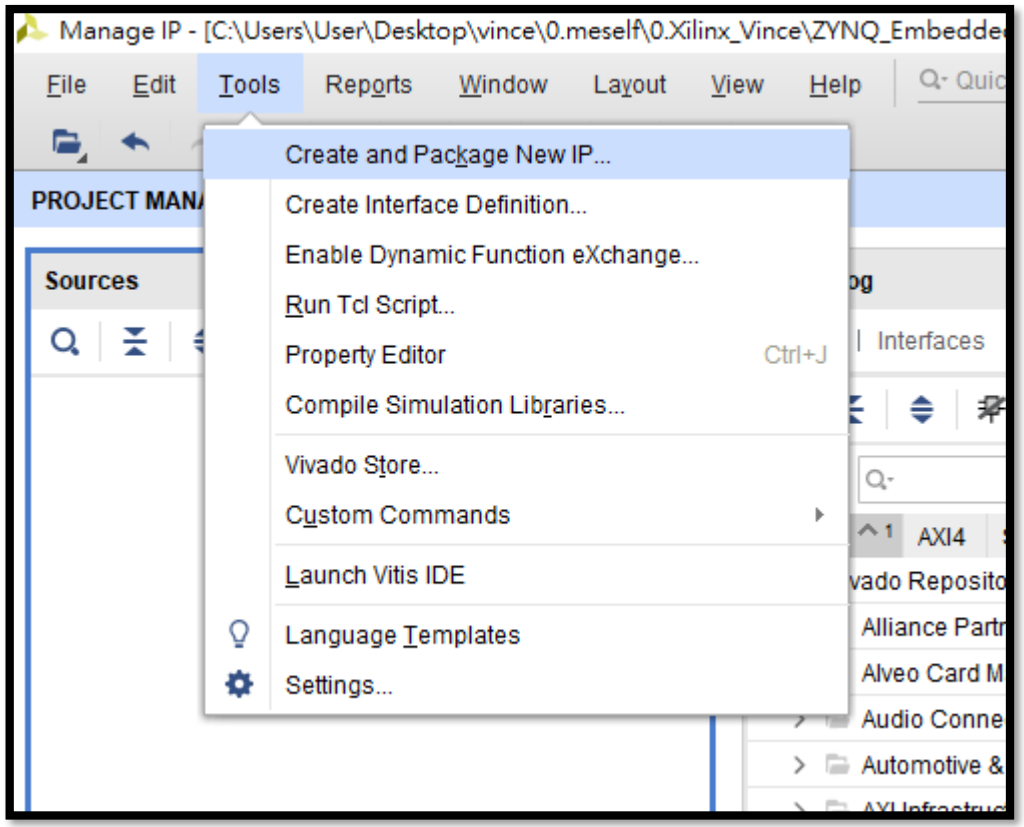
Note: For the simple purpose of adding custom RTL for use in IP integrator, the Module Reference feature is available and described in the *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994).

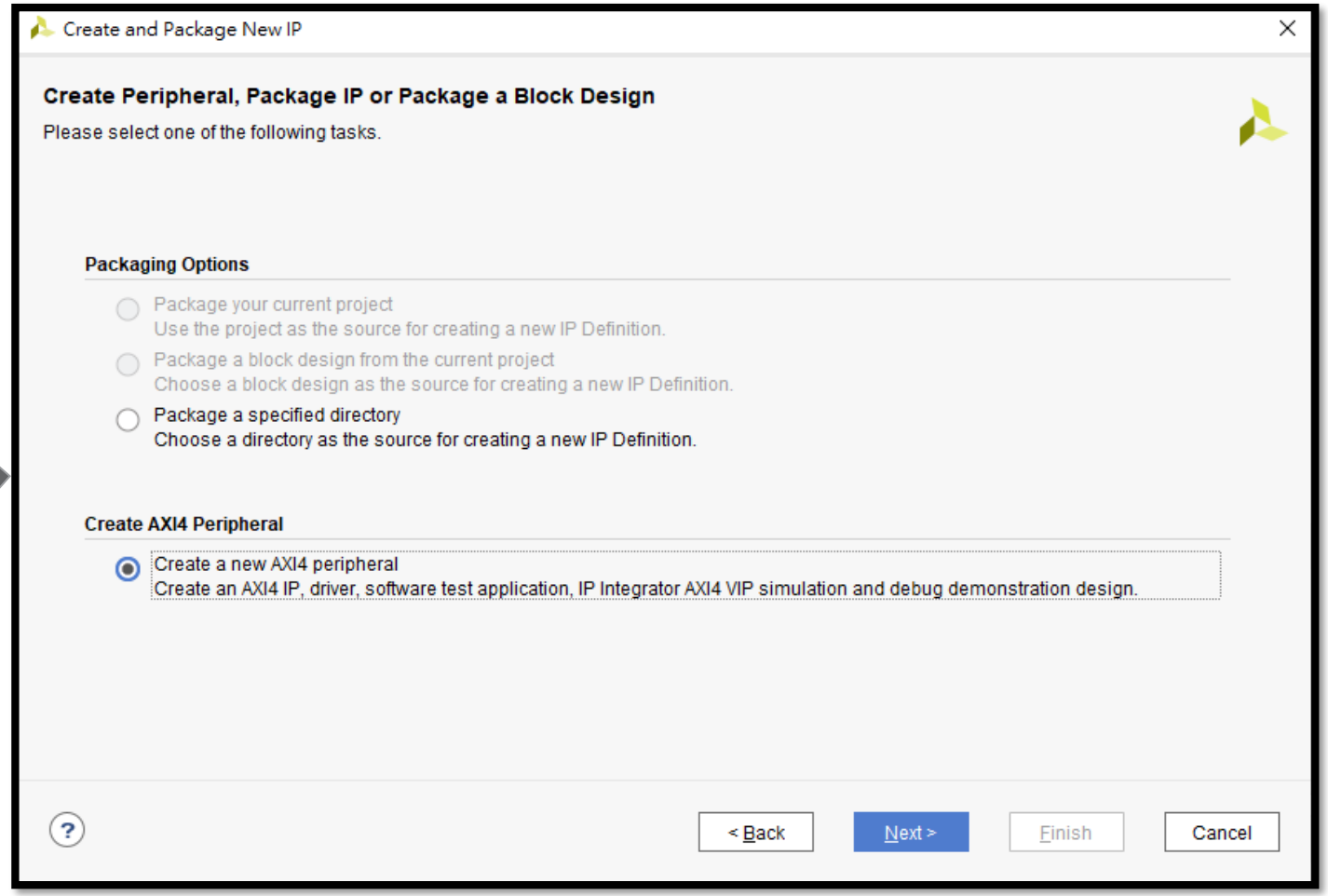
Experiment purpose

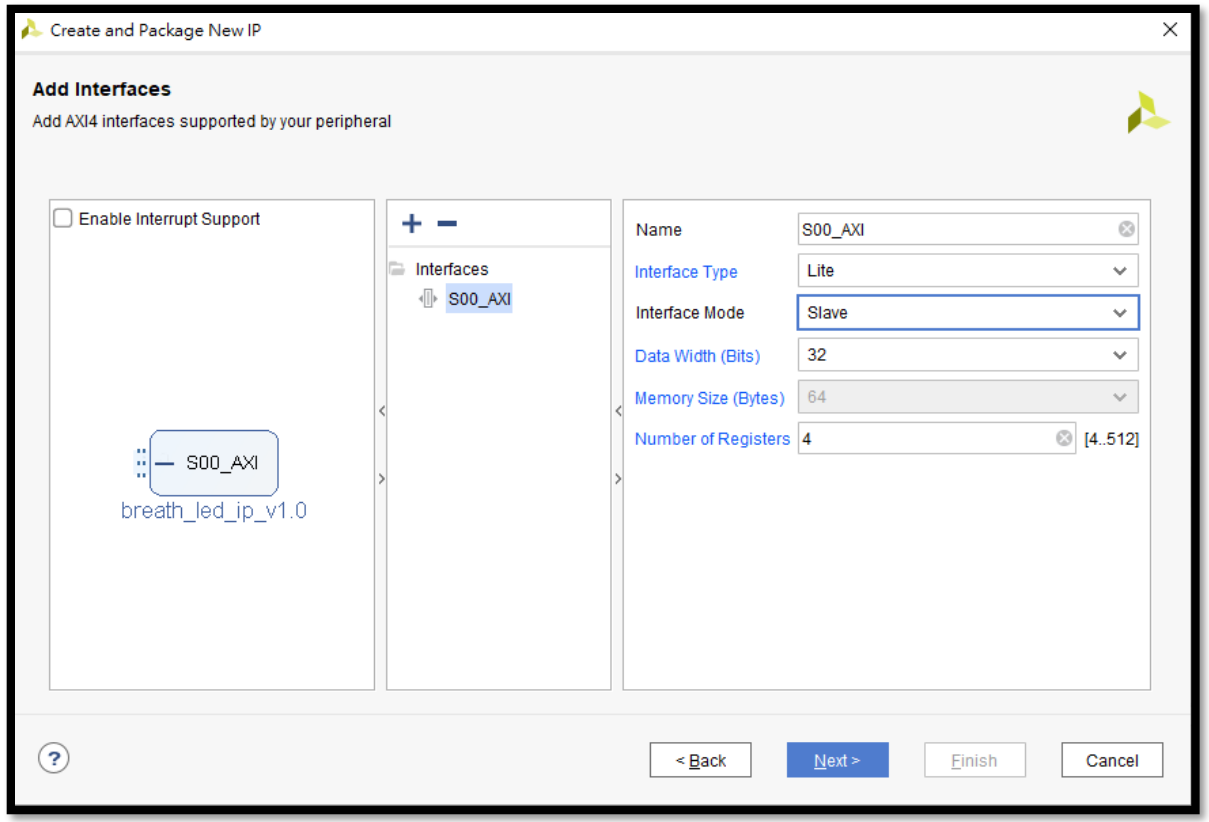
Custom LED IP, control PL LED to present breathing LED, and PS can use AXI interface to control the switch and Frequency.

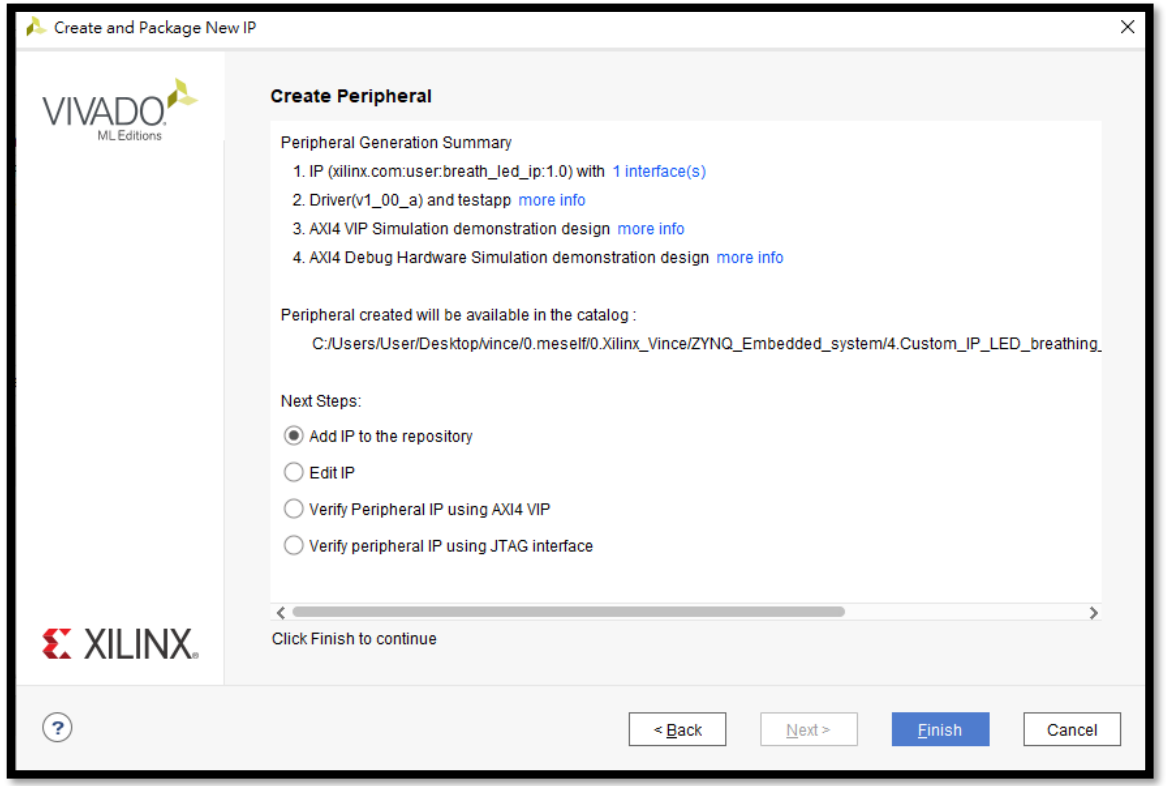
System Block diagram

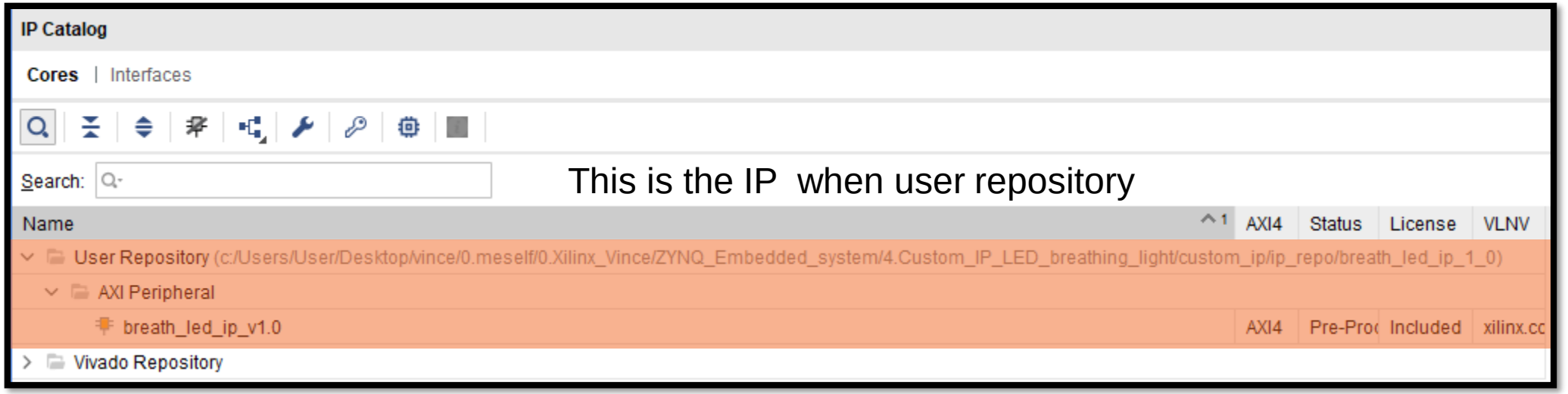


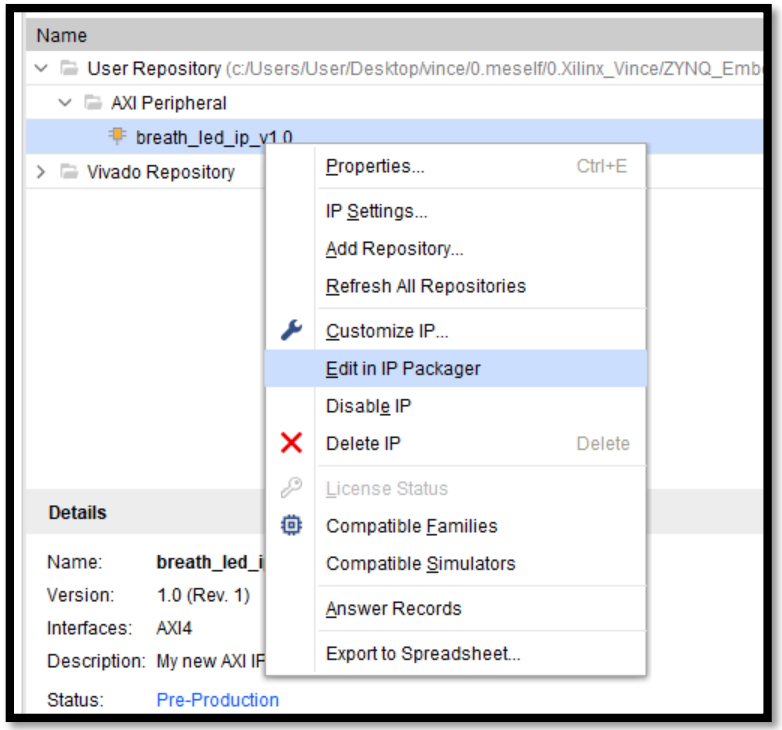


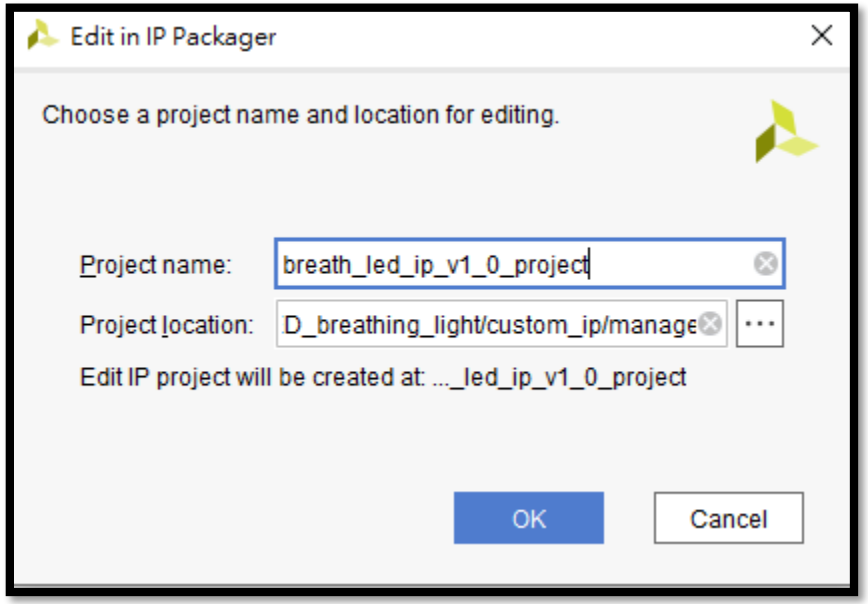




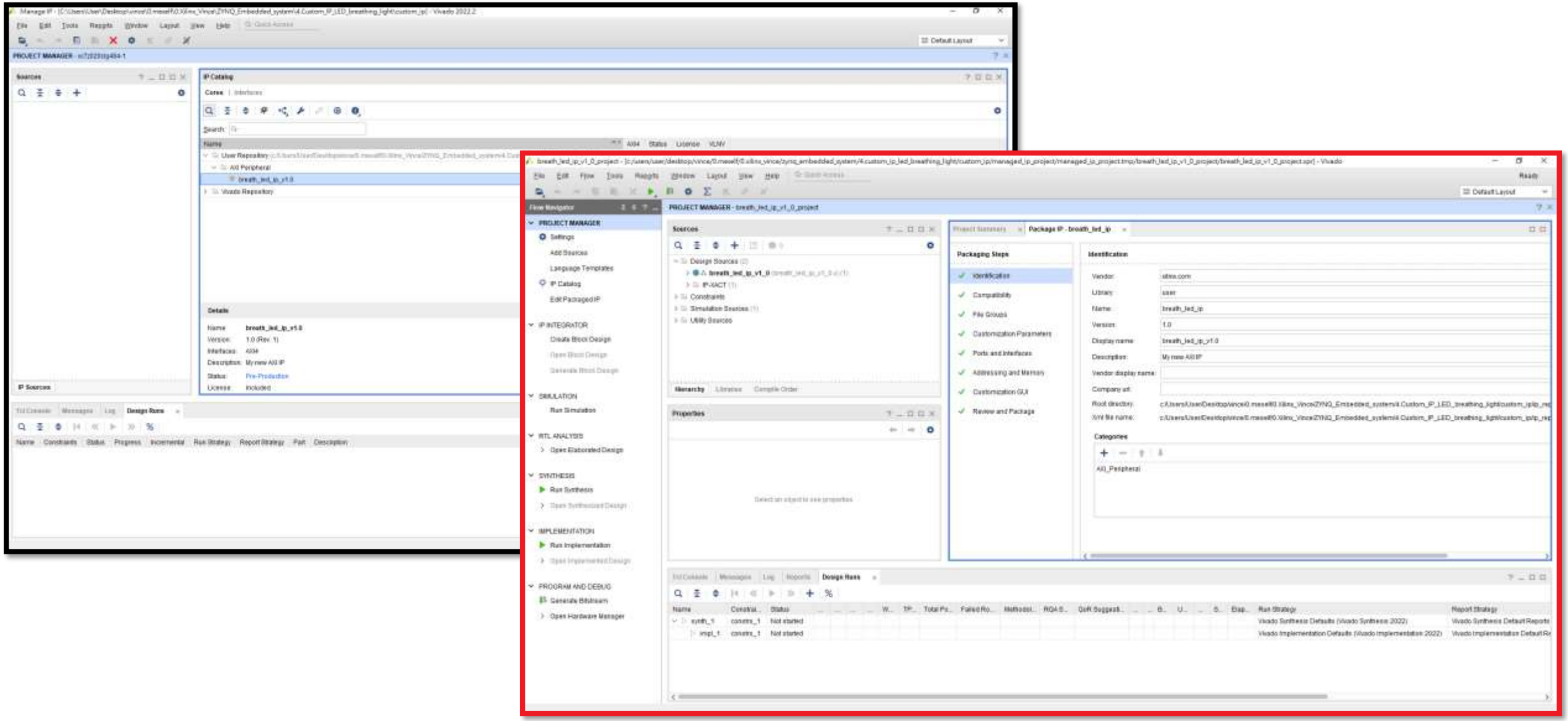


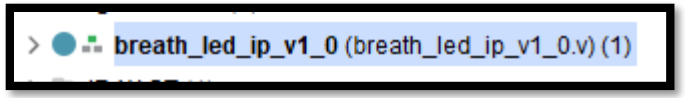


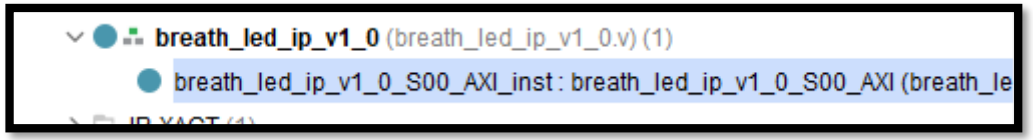




Than mange ip will open other the Vivad project.



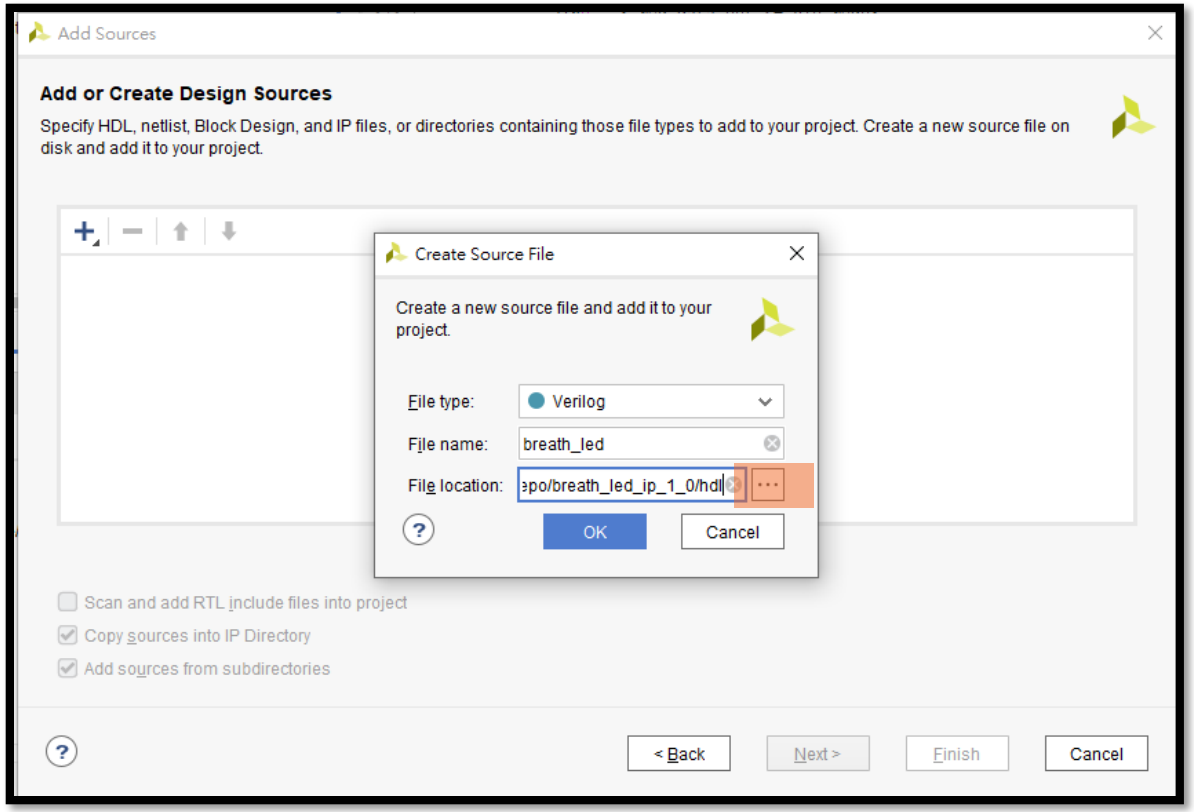




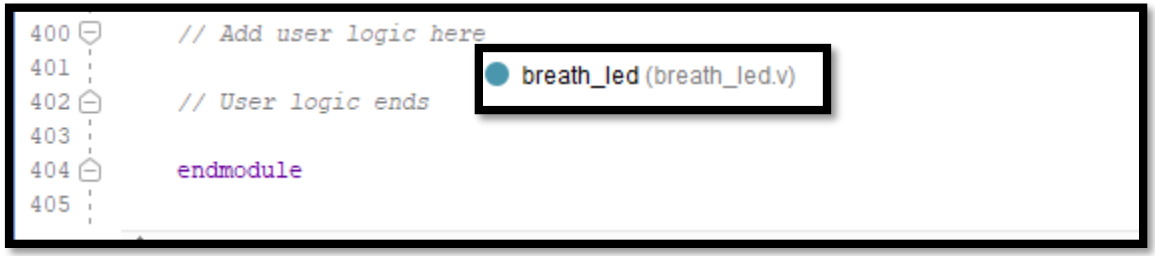

```
// Ports of Axi Slave Bus Interface S00_AXI
input wire  s00_axi_aclk,
input wire  s00_axi_aresetn,
input wire [C_S00_AXI_ADDR_WIDTH-1 : 0] s00_axi_awaddr,
input wire [2 : 0] s00_axi_awprot,
input wire  s00_axi_awvalid,
output wire  s00_axi_awready,
input wire [C_S00_AXI_DATA_WIDTH-1 : 0] s00_axi_wdata,
input wire [(C_S00_AXI_DATA_WIDTH/8)-1 : 0] s00_axi_wstrb,
input wire  s00_axi_wvalid,
output wire  s00_axi_wready,
output wire [1 : 0] s00_axi_bresp,
output wire  s00_axi_bvalid,
input wire  s00_axi_bready,
input wire [C_S00_AXI_ADDR_WIDTH-1 : 0] s00_axi_araddr,
input wire [2 : 0] s00_axi_arprot,
input wire  s00_axi_arvalid,
output wire  s00_axi_arready,
output wire [C_S00_AXI_DATA_WIDTH-1 : 0] s00_axi_rdata,
output wire [1 : 0] s00_axi_rresp,
output wire  s00_axi_rvalid,
input wire  s00_axi_rready
```

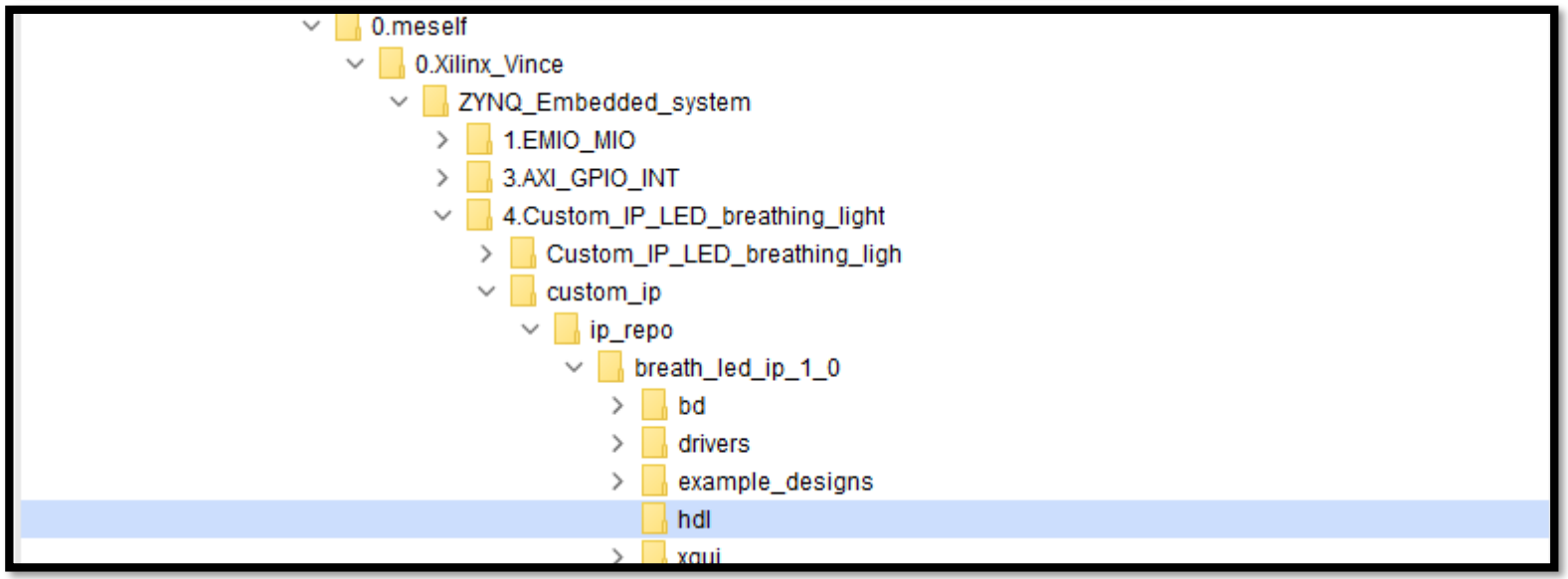
```
400     // Add user logic here
401
402     // User logic ends
403
404     endmodule
405
```

AXI Lite interface port



Create the breath_led, connect under the //add user logic here





breath_led_ip_v1_0.v(Top)

```
1
2   `timescale 1 ns / 1 ps
3
4       module breath_led_ip_v1_0 #
5       (
6           // Users to add parameters here
7           parameter START_FREA_STEP = 10'd100,  // setup the frequency interval initial value
8           // User parameters ends
9           // Do not modify the parameters beyond this line
10
11
12          // Parameters of Axi Slave Bus Interface S00_AXI
13          parameter integer C_S00_AXI_DATA_WIDTH  = 32,
14          parameter integer C_S00_AXI_ADDR_WIDTH  = 4
15      )
16      (
17          // Users to add ports here
18           output  led,
19          // User ports ends
20          // Do not modify the ports beyond this line
```

```
47  // Instantiation of Axi Bus Interface S00_AXI
48      breath_led_ip_v1_0_S00_AXI # (
49          .START_FREA_STEP(START_FREA_STEP),
50          .C_S_AXI_DATA_WIDTH(C_S00_AXI_DATA_WIDTH),
51          .C_S_AXI_ADDR_WIDTH(C_S00_AXI_ADDR_WIDTH)
52      ) breath_led_ip_v1_0_S00_AXI_inst (
53          .led(led),
54          .S_AXI_ACLK(s00_axi_aclk),
55          .S_AXI_ARESETN(s00_axi_aresetn),
56          .S_AXI_AWADDR(s00_axi_awaddr),
57          .S_AXI_AWPROT(s00_axi_awprot),
58          .S_AXI_AWVALID(s00_axi_awvalid),
59          .S_AXI_AWREADY(s00_axi_awready),
60          .S_AXI_WDATA(s00_axi_wdata),
61          .S_AXI_WSTRB(s00_axi_wstrb),
```

breath_led_ip_v1_0_S00_AXI.v

```
`timescale 1 ns / 1 ps

    module breath_led_ip_v1_0_S00_AXI #
    (
        // Users to add parameters here

        // User parameters ends

        parameter START_FREA_STEP = 10'd100,  // setup the frequency interval initial value

        // Do not modify the parameters beyond this line

        // Width of S_AXI data bus
        parameter integer C_S_AXI_DATA_WIDTH    = 32,
        // Width of S_AXI address bus
        parameter integer C_S_AXI_ADDR_WIDTH    = 4
```

Add code ,refer the figure.

In the top need to define parameter, because in the breath_led.v has defined the parameter.

breath_led_ip_v1_0.v & breath_led_ip_v1_0_S00_AXI.v

had defined START_FREA_STEP,

need to write the breath_led #().

Use the Axi clk and reset, in the beginning, had set the AXI interface, data width is 32 bit, number of registers is 4,

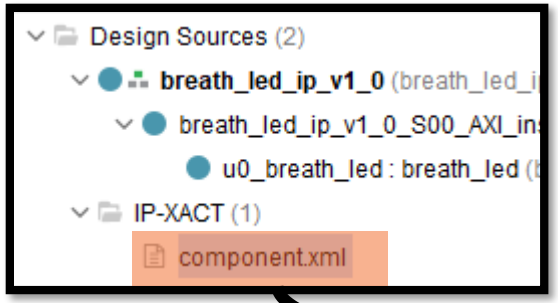
(slv_reg0; slv_reg1; slv_reg2; slv_reg3;)

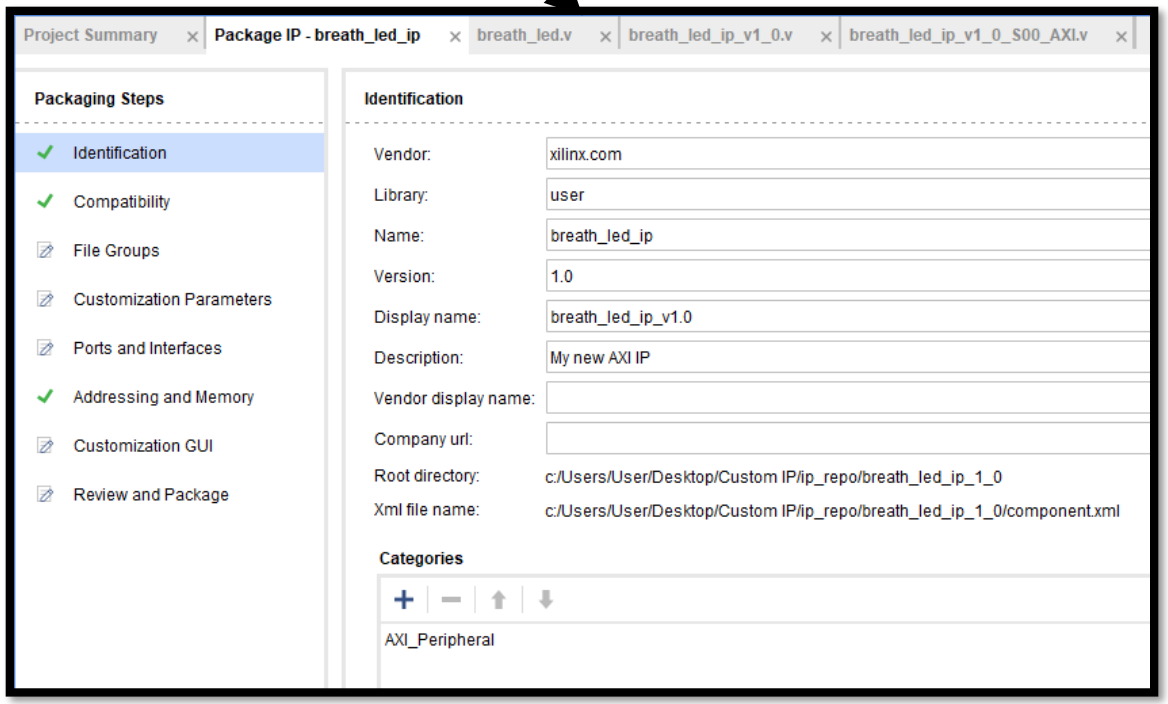
Add port

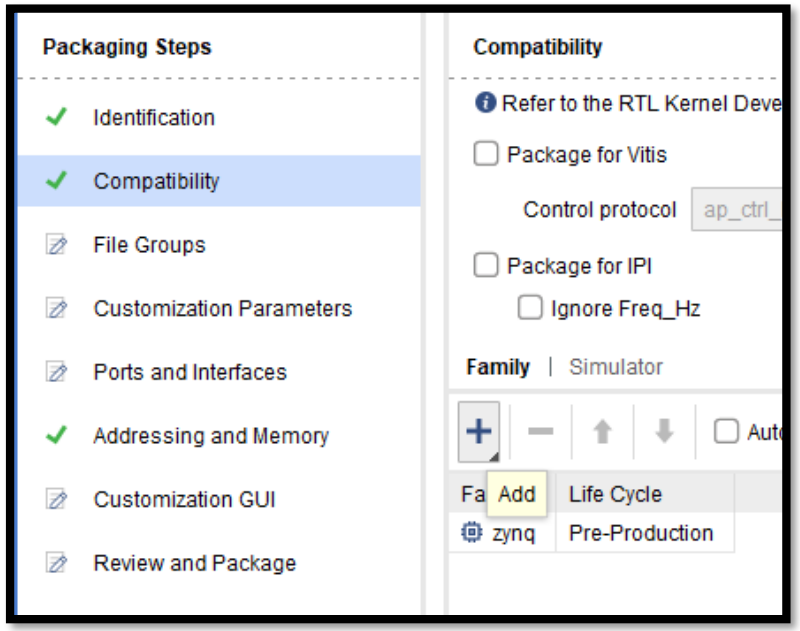
```
    // Add user logic here
    breath_led #(
                .START_FREA_STEP(START_FREA_STEP))

    u0_breath_led(
                    .sys_clk            (S_AXI_ACLK         ),
                    .sys_rst_n          (S_AXI_ARESETN      ),
                    .sw_ctrl            (slv_reg0[0]        ),
                    .set_en             (slv_reg0[1]        ),
                    .set_freq_step      (slv_reg0[11:2]     ),
                    .led                (led                )
    );
    // User logic ends
```

```
20      // Users to add ports here
21      output  led,
22      // User ports ends
23      // Do not modify the ports beyond this line
24
```

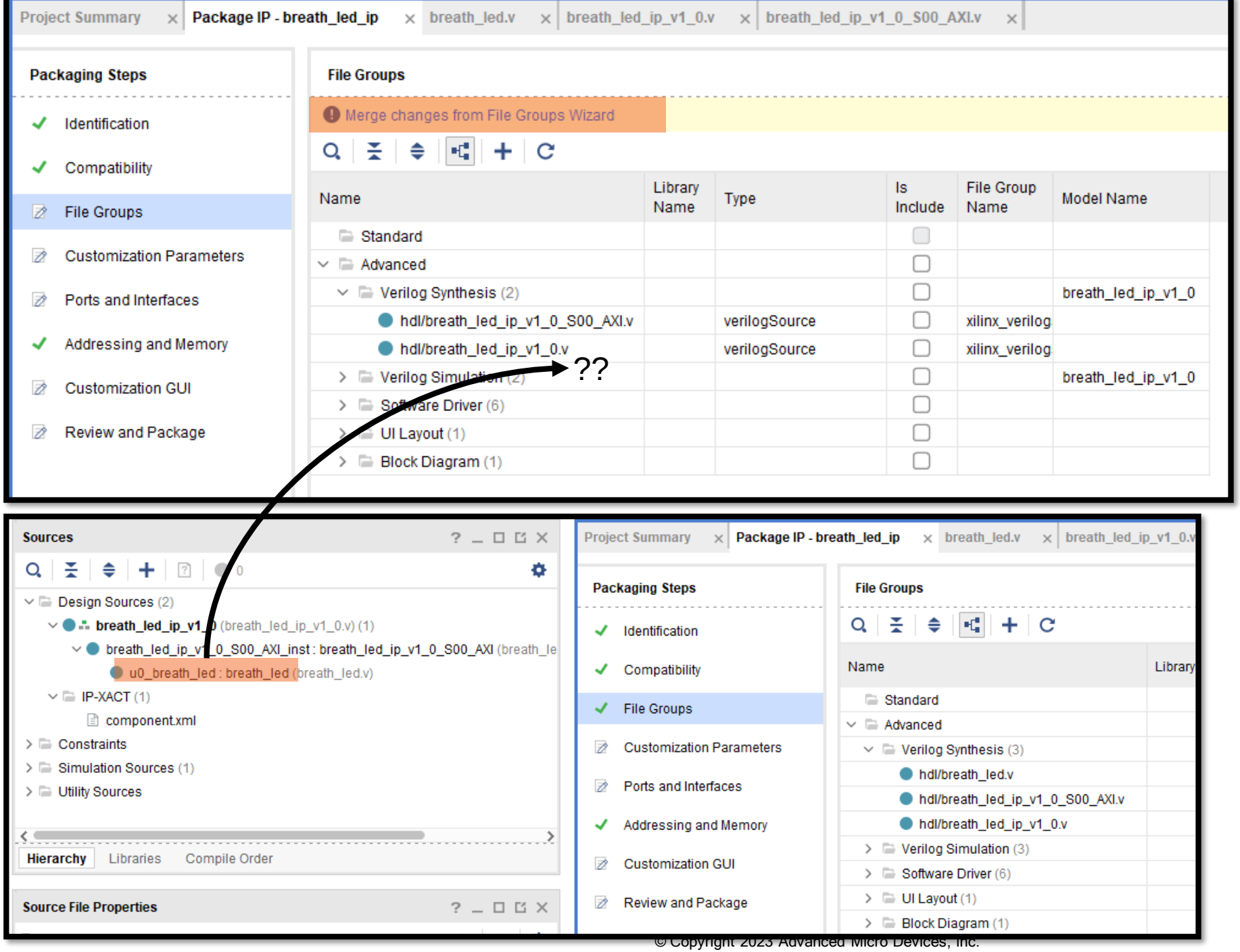

```
//-- Number of Slave Registers 4
reg [C_S_AXI_DATA_WIDTH-1:0]    slv_reg0;
reg [C_S_AXI_DATA_WIDTH-1:0]    slv_reg1;
reg [C_S_AXI_DATA_WIDTH-1:0]    slv_reg2;
reg [C_S_AXI_DATA_WIDTH-1:0]    slv_reg3;
```

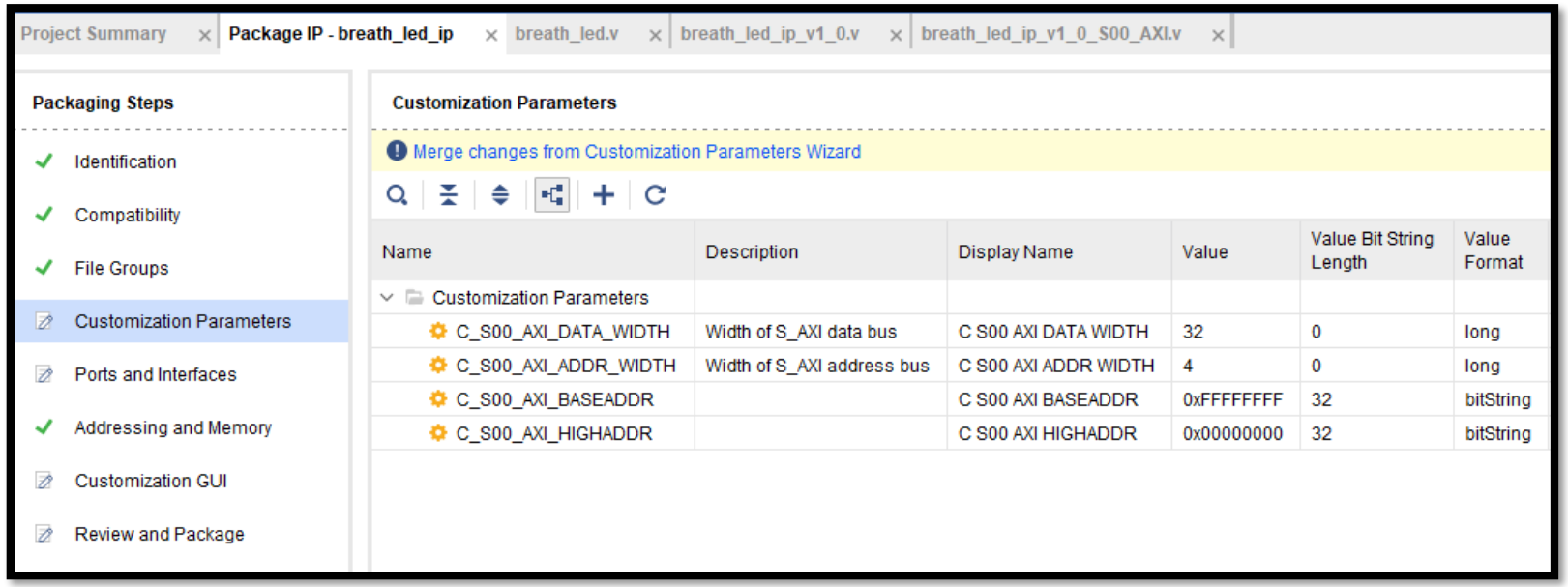


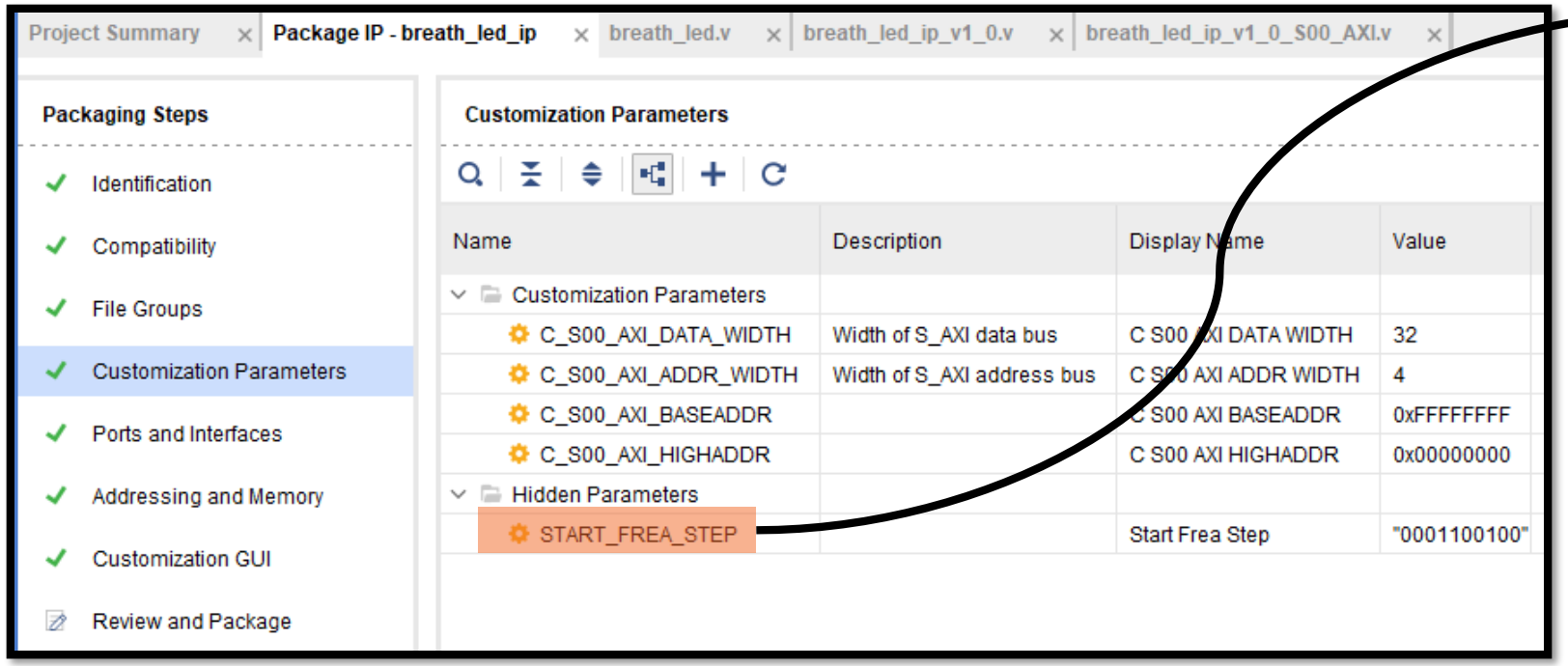


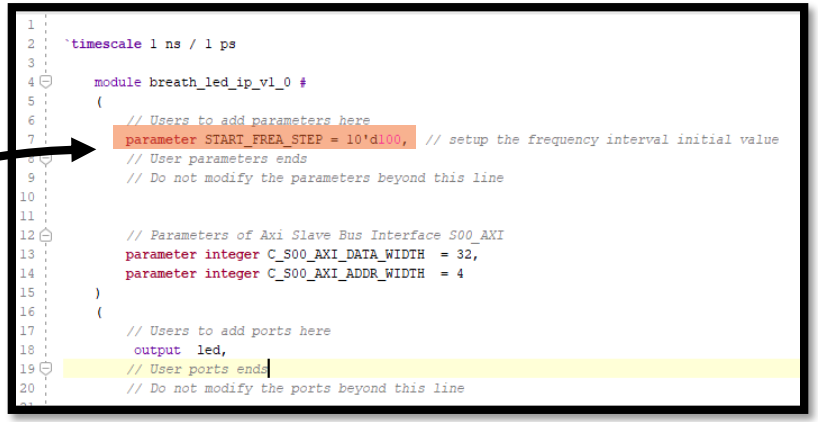


Add the board family

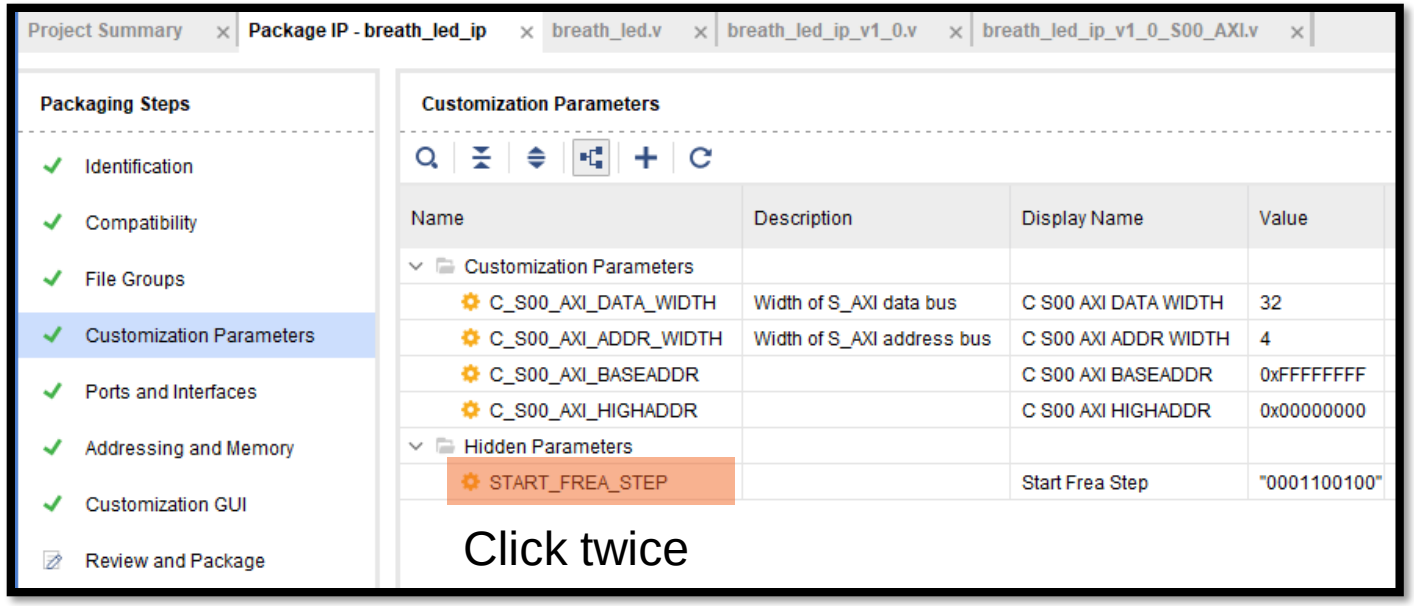


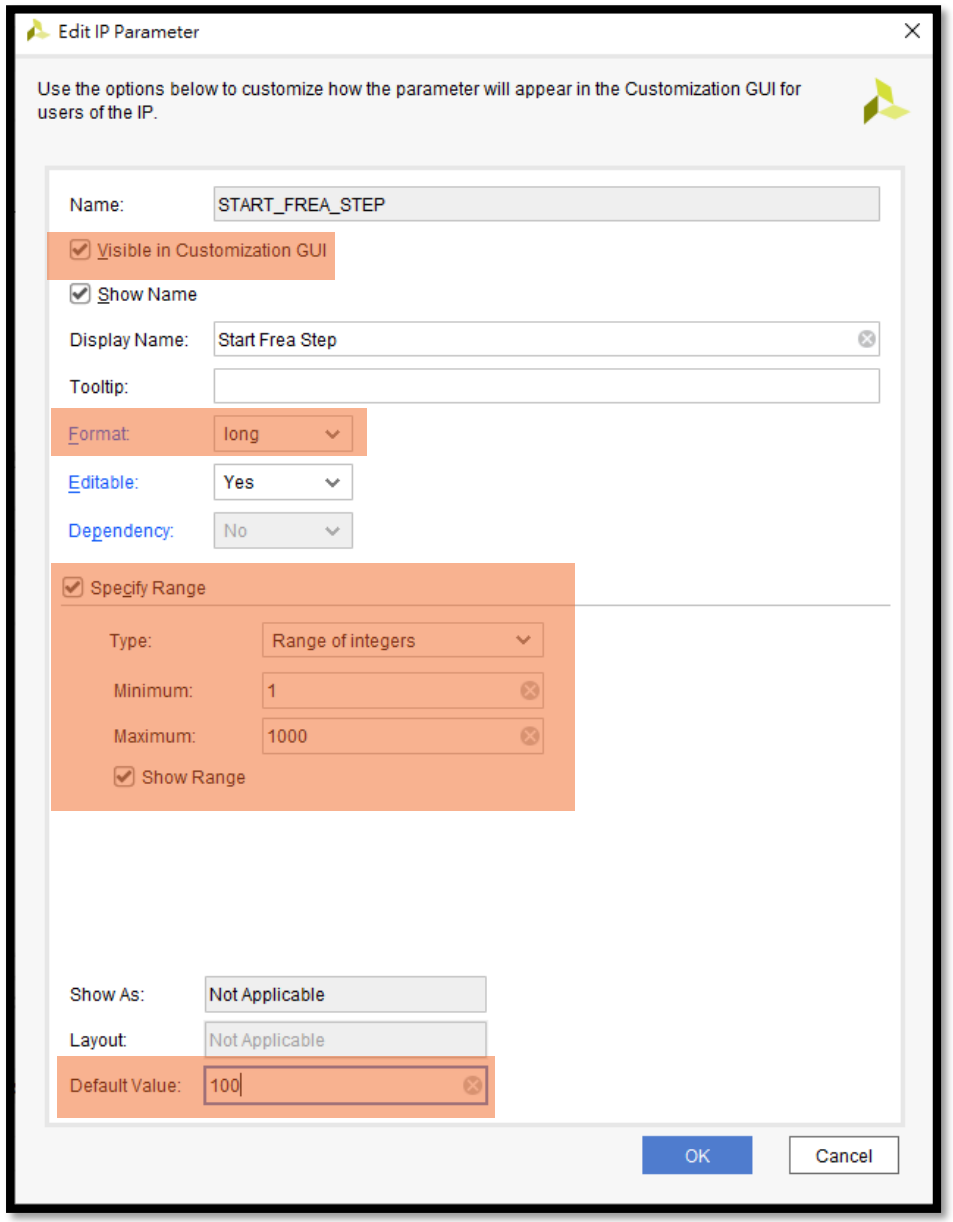


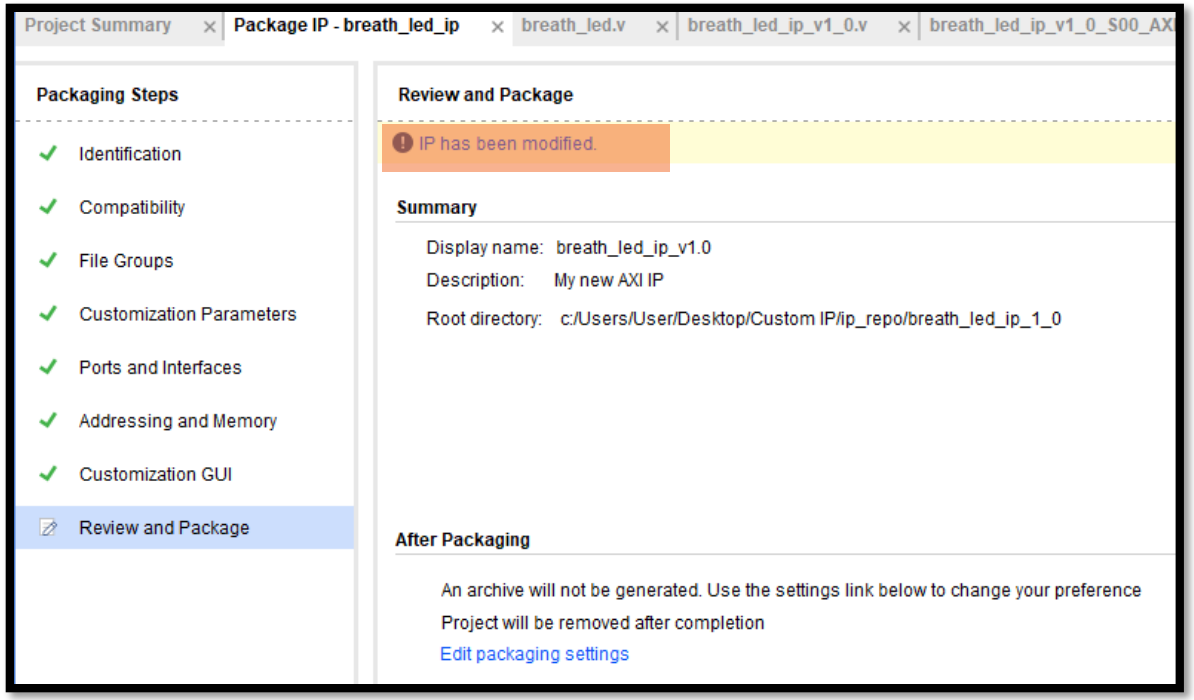


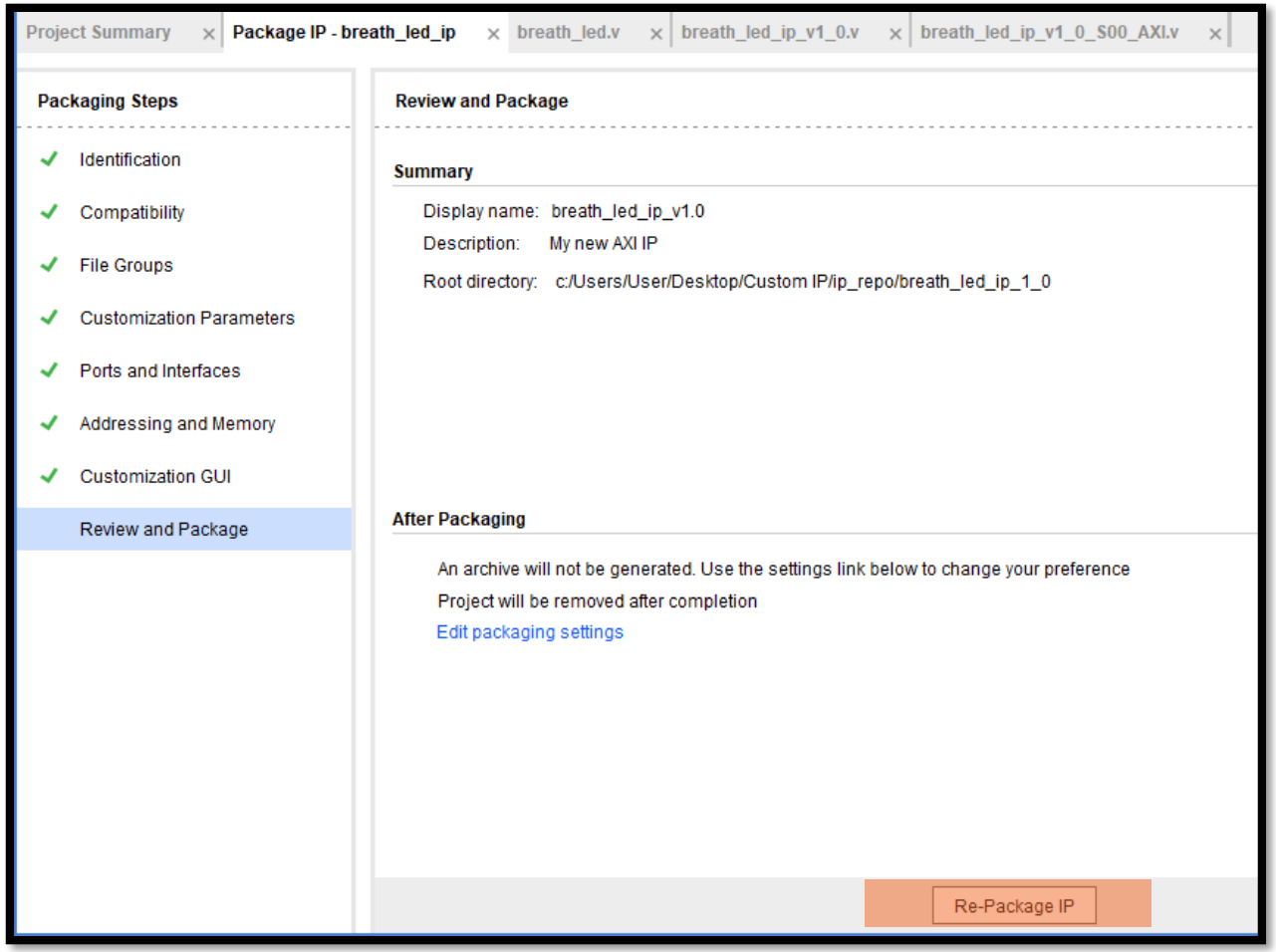


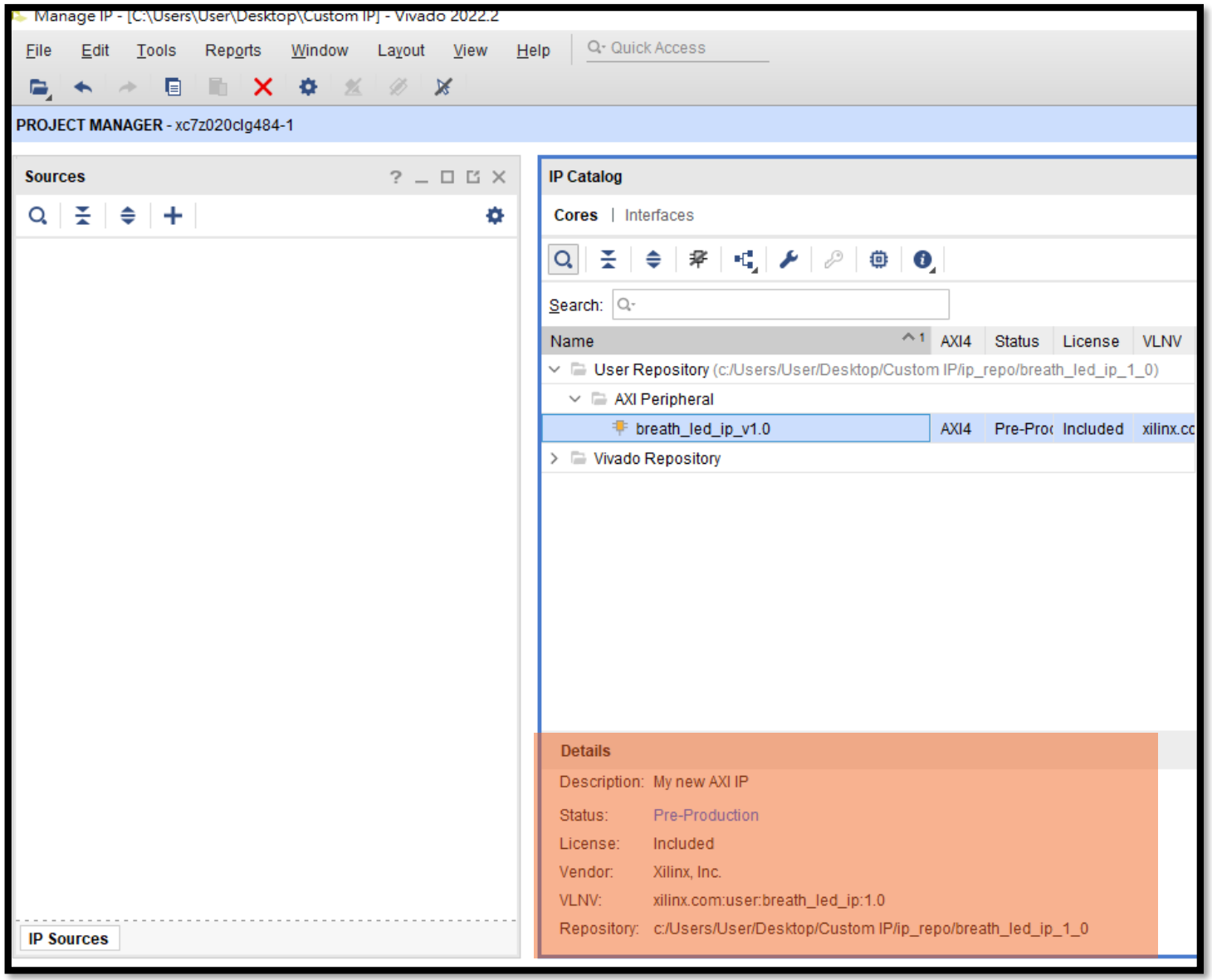
```
1
2   `timescale 1 ns / 1 ps
3
4       module breath_led_ip_v1_0 #
5       (
6           // Users to add parameters here
7           parameter START_FREA_STEP = 10'd100,  // setup the frequency interval initial value
8           // User parameters ends
9           // Do not modify the parameters beyond this line
10
11
12          // Parameters of Axi Slave Bus Interface S00_AXI
13          parameter integer C_S00_AXI_DATA_WIDTH  = 32,
14          parameter integer C_S00_AXI_ADDR_WIDTH  = 4
15      )
16      (
17          // Users to add ports here
18           output  led,
19          // User ports ends
20          // Do not modify the ports beyond this line
```

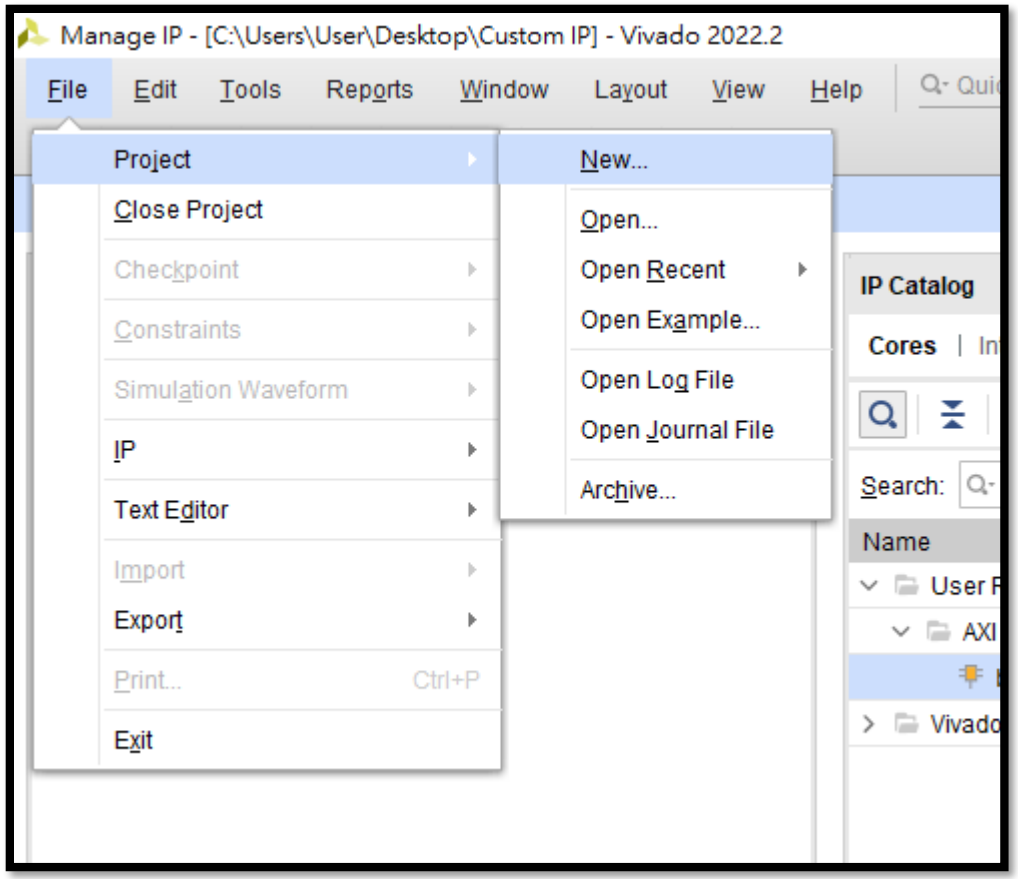




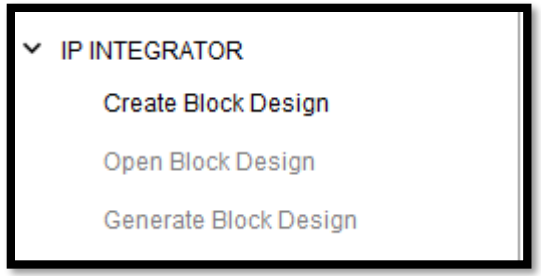


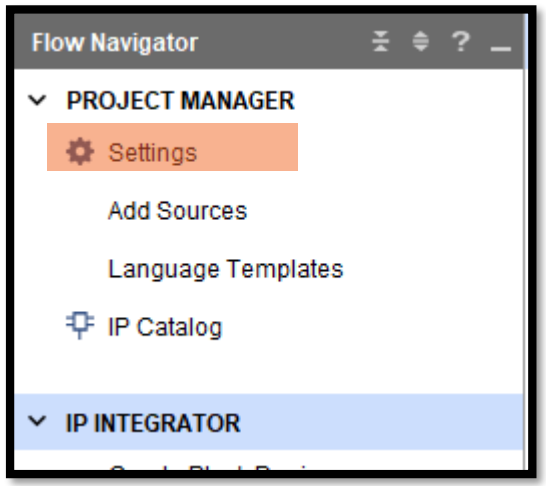


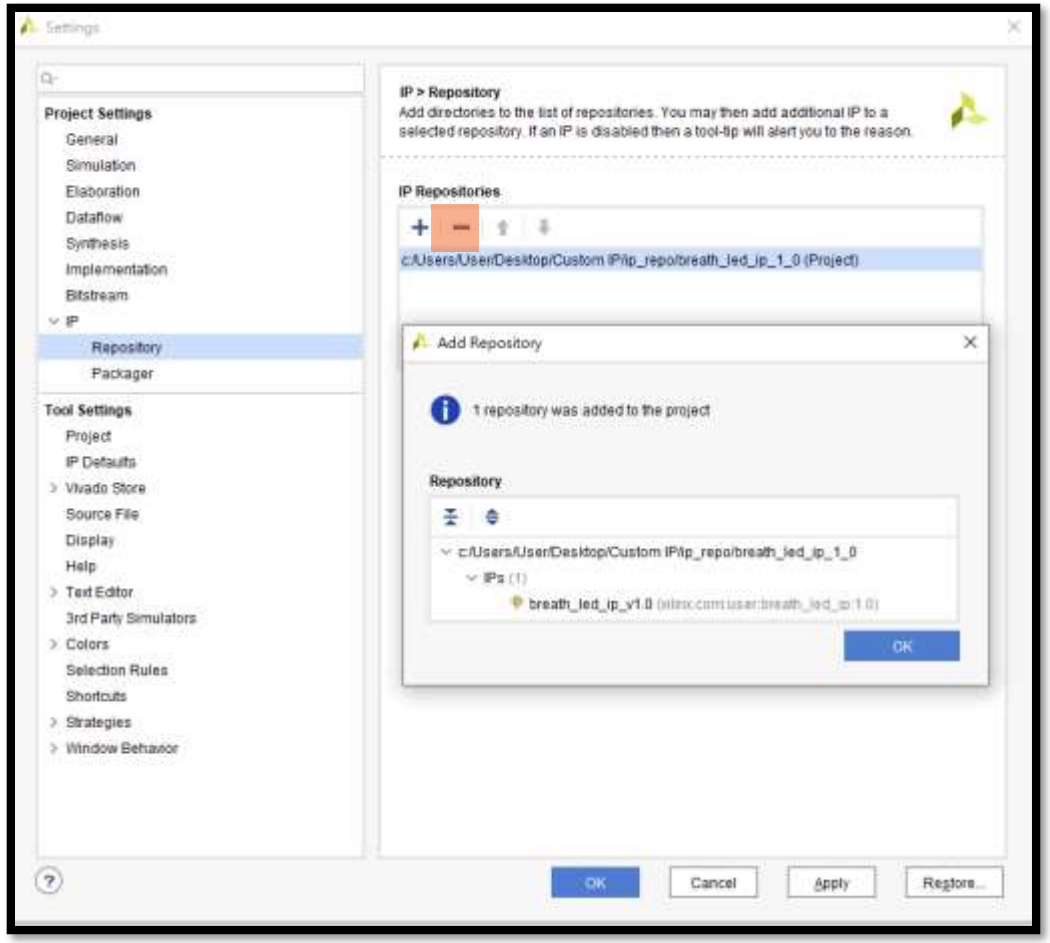


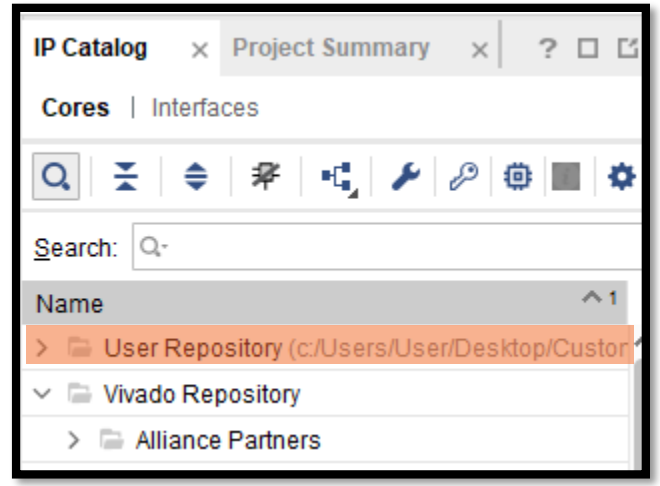


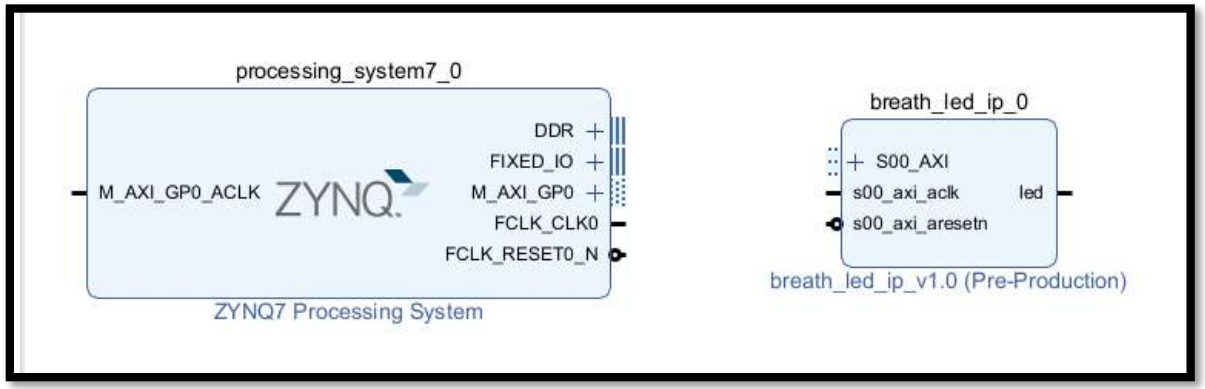
New project

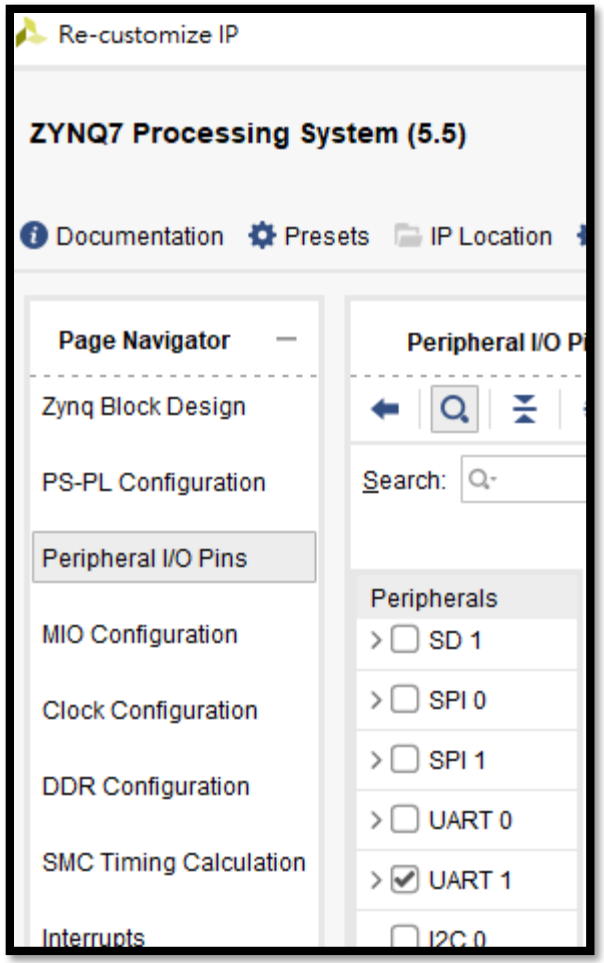


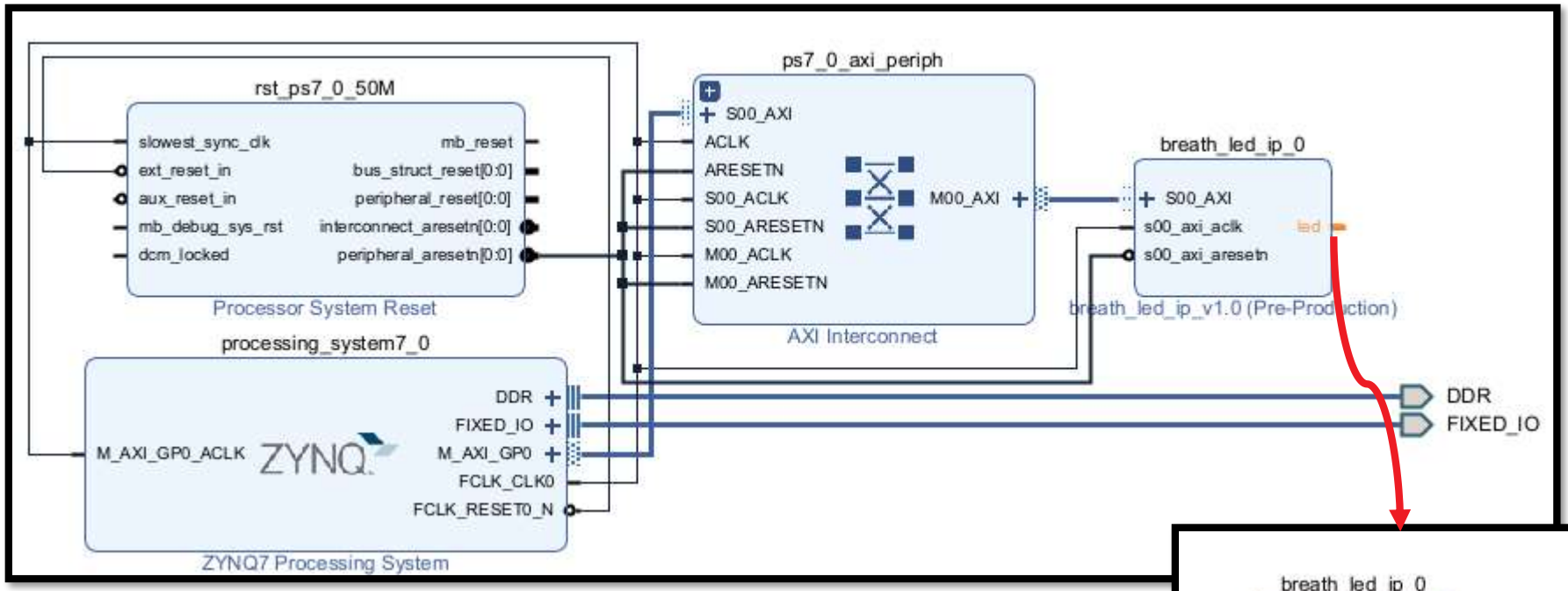


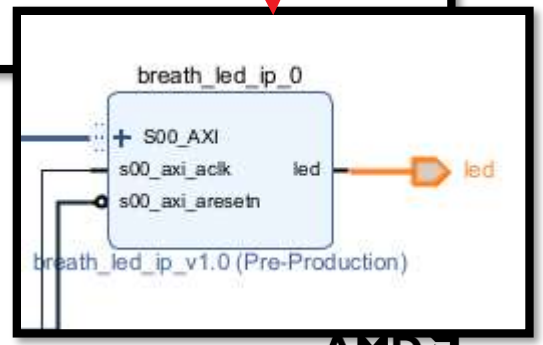


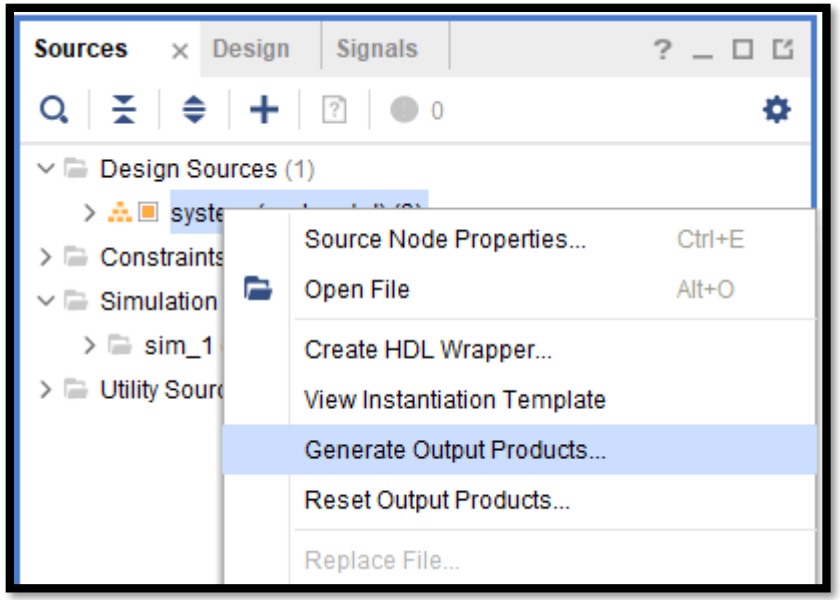


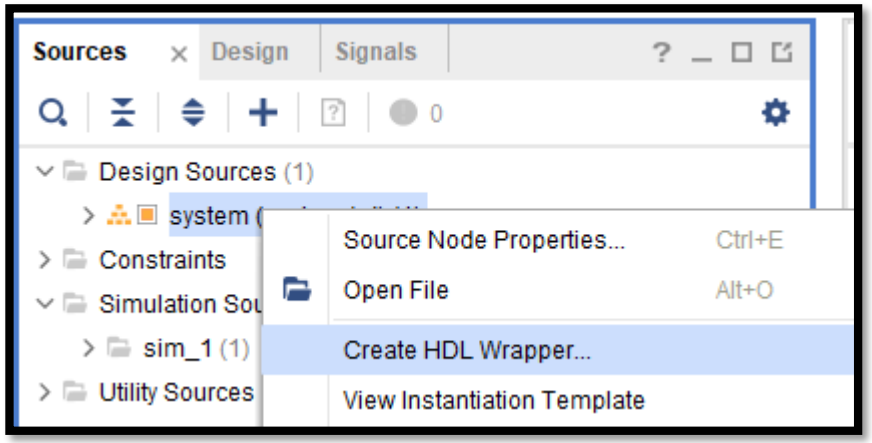


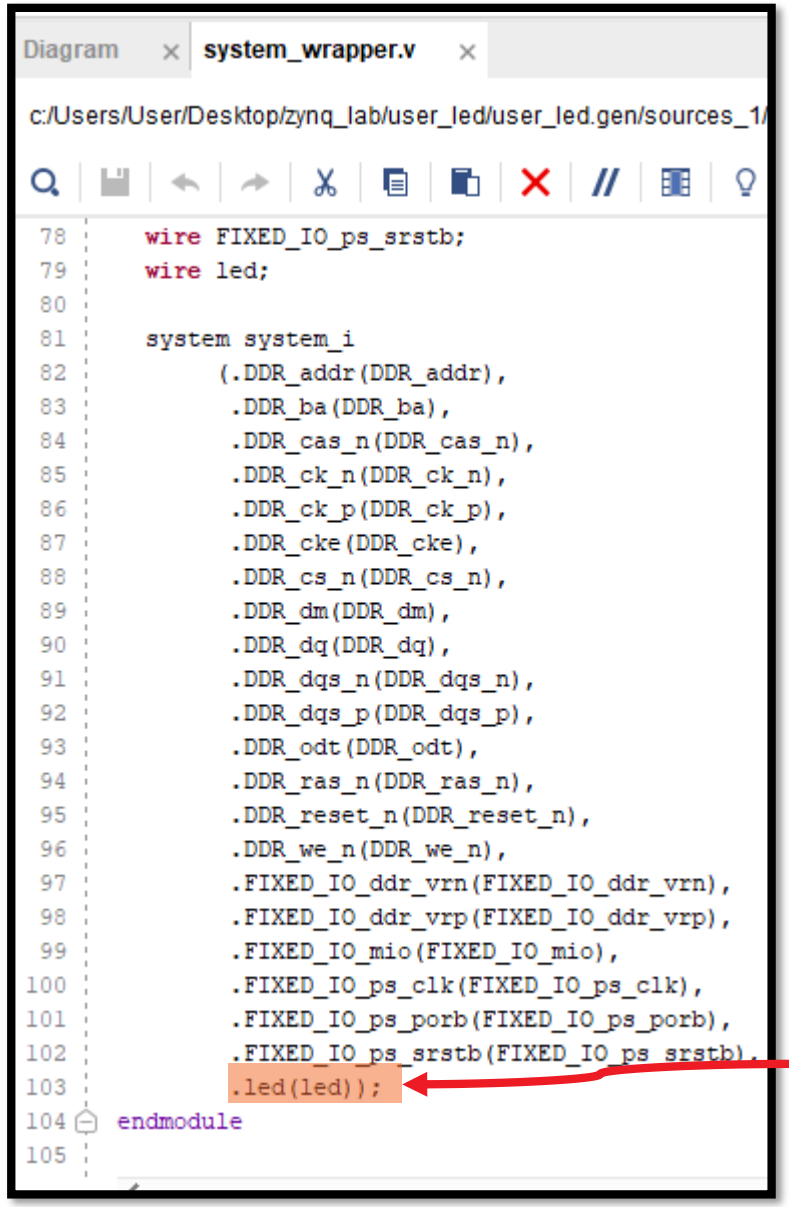




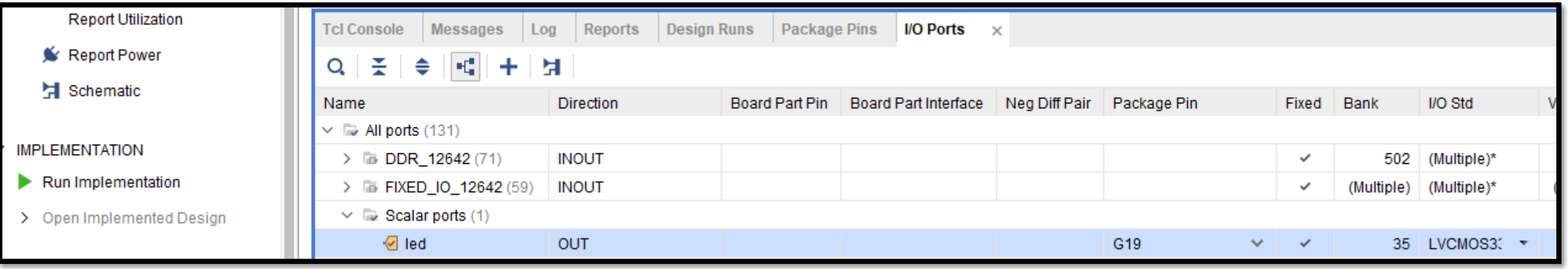


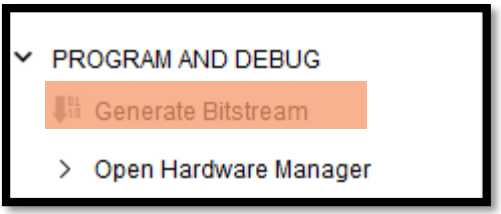






PL port need constrains






```
 */
#define BREATH_LED_IP_mWriteReg(BaseAddress, RegOffset, Data) \
    Xil_Out32((BaseAddress) + (RegOffset), (u32)(Data))
```

```
 */
#define BREATH_LED_IP_mReadReg(BaseAddress, RegOffset) \
    Xil_In32((BaseAddress) + (RegOffset))
```

breath_led_ip.h

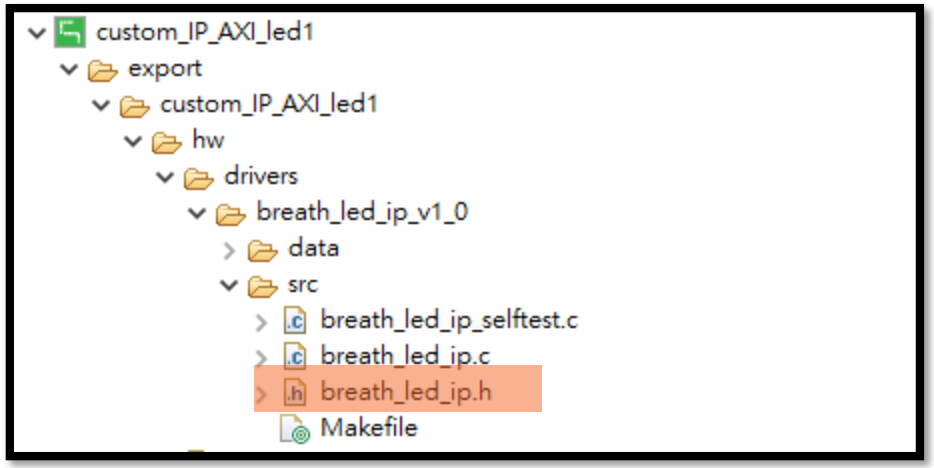
```
#ifndef BREATH_LED_IP_H
#define BREATH_LED_IP_H


/****************** Include Files ********************/
#include "xil_types.h"
#include "xstatus.h"

#define BREATH_LED_IP_S00_AXI_SLV_REG0_OFFSET 0
#define BREATH_LED_IP_S00_AXI_SLV_REG1_OFFSET 4
#define BREATH_LED_IP_S00_AXI_SLV_REG2_OFFSET 8
#define BREATH_LED_IP_S00_AXI_SLV_REG3_OFFSET 12
```

Define the AXI_SLV_REG0~REG4
We can use the function to Read/Write the reg

INCLUDEFILES=*.h
LIBSOURCES=*.c
OUTS = *.o

INCLUDEFILES=$(wildcard *.h)
LIBSOURCES=$(wildcard *.c)
OUTS = $(wildcard *.o)

```
COMPILER=
ARCHIVER=
CP=cp
COMPILER_FLAGS=
EXTRA_COMPILER_FLAGS=
LIB=libxil.a

RELEASEDIR=../../../lib
INCLUDEDIR=../../../include
INCLUDES=-I./. -I${INCLUDEDIR}

INCLUDEFILES=$(wildcard *.h)
LIBSOURCES=$(wildcard *.c)
OUTS = $(wildcard *.o)

libs:
	echo "Compiling axiled..."
	$(COMPILER) $(COMPILER_FLAGS) $(EXTRA_COMPILER_FLAGS) $(INCLUDES) $(LIBSOURCES)
	$(ARCHIVER) -r ${RELEASEDIR}/${LIB} ${OUTS}
	make clean

include:
	${CP} $(INCLUDEFILES) $(INCLUDEDIR)

clean:
	rm -rf ${OUTS}
```




```
#define BREATH_LED_IP_mWriteReg(BaseAddress, RegOffset, Data)
```

xparameters.h

```
/* Definitions for driver BREATH_LED_IP */
#define XPAR_BREATH_LED_IP_NUM_INSTANCES 1

/* Definitions for peripheral BREATH_LED_IP_0 */
#define XPAR_BREATH_LED_IP_0_DEVICE_ID 0
#define XPAR_BREATH_LED_IP_0_S00_AXI_BASEADDR 0x43C00000
#define XPAR_BREATH_LED_IP_0_S00_AXI_HIGHADDR 0x43C0FFFF
```

breath_led_ip.h

```
#define BREATH_LED_IP_S00_AXI_SLV_REG0_OFFSET 0
#define BREATH_LED_IP_S00_AXI_SLV_REG1_OFFSET 4
#define BREATH_LED_IP_S00_AXI_SLV_REG2_OFFSET 8
#define BREATH_LED_IP_S00_AXI_SLV_REG3_OFFSET 12
```

References

UG1118


AMD