# MIO/EMIO Controller LED

AMD
together we advance_

# GPIO MIO CONTROLL led

STEP

1. GPIO INRTODUCTION
2. HARDWAVE DESIGN
3. PROGRAMMING DESIGN

**GPIO 可以獨立且動態編成，作為輸入輸出以及中斷。**

**GPIO分成四組BANK (PS端引角 54個)**

BANK 0 ,1 MIO

0: 32B

1:22B

BANK 2,3 EMIO

2:32B

3:32B

軟件通過一組寄存器映射(MEMORYMAP)的暫存器來控制GPIO

**暫存器組:**

DATA_RO 用來反映PIN的狀態

DATA 在GPIO被配置乘輸出的時候 可以控制輸出的數值。

MASK_DATA_LSW 用於屏蔽DATA的低16位元。

MASK-DATA_MSW 用於屏蔽DATA的高16位元。

DIRM 用於控制IO引角作為輸入或輸出。0 :關閉輸出驅動 1: 使能輸出驅動

OEN output enable,

**GPIO 可以獨立且動態編成，作為輸入輸出以及中斷。**

**GPIO分成四組BANK (PS端引角 54個)**

BANK 0 ,1 MIO

0: 32B

1:22B

BANK 2,3 EMIO

2:32B

3:32B

軟件通過一組寄存器映射(MEMORYMAP)的暫存器來控制GPIO

**暫存器組:**

DATA_RO 用來反映PIN的狀態

DATA 在GPIO被配置乘輸出的時候 可以控制輸出的數值。

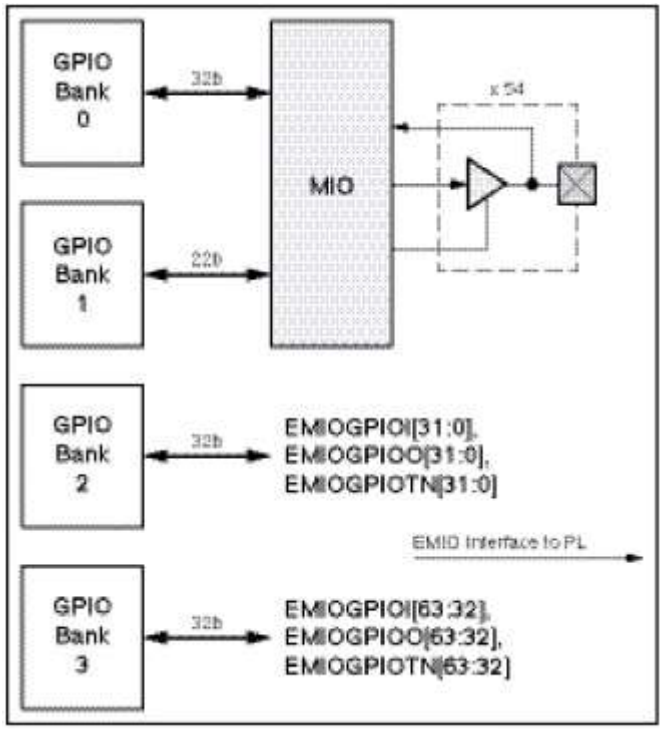
MASK_DATA_LSW 用於屏蔽DATA的低16位元。

MASK-DATA_MSW 用於屏蔽DATA的高16位元。

DIRM 用於控制IO引角作為輸入或輸出。0 :關閉輸出驅動 1: 使能輸出驅動

OEN output enable,

MIO[8:7] 在係土復位過程中作為VMOOE PIN (作為輸入)，用於配置MIO BANK 的電壓，復位結束後，MIO[8:7]只能作為輸出信號。



## Start-up Sequence

**Main Example: Start-up Sequence**

1. **Resets:** The reset options are described in section Resets.
2. **Clocks:** The clocks are described in section Clocks.
3. **GPIO Pin Configurations:** Configure pin as input/output is described in section GPIO Pin Configurations.
4. **Write Data to GPIO Output pin**: Refer to example in section Writing Data to GPIO Output Pins.
5. **Read Data from GPIO Input pin:** Refer to example in section Reading Data from GPIO Input Pins .
6. **Set GPIO pin as wake-up event**: Refer to example in section GPIO as Wake-up Event.

## GPIO Pin Configurations

Each individual GPIO pin can be configured as input/output. However, bank0 [8:7] pins must be configured as outputs. Refer to section Bank0, Bits[8:7] are Outputs for further details.

**Example: Configure MIO pin 10 as an output**

1. **Set the direction as output:** Write `0x0000_0400` to the gpio.DIRM_0 register.
2. **Set the output enable**: Write `0x0000_0400` to the gpio.OEN_0 register.

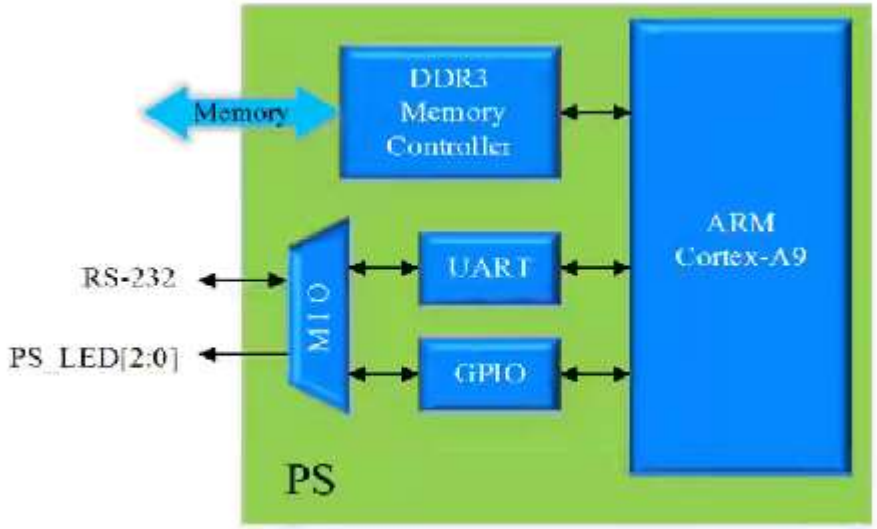
***Note:*** The output enable has significance only when the GPIO pin is configured as an output.

**Example: Configure MIO pin 10 as an input**

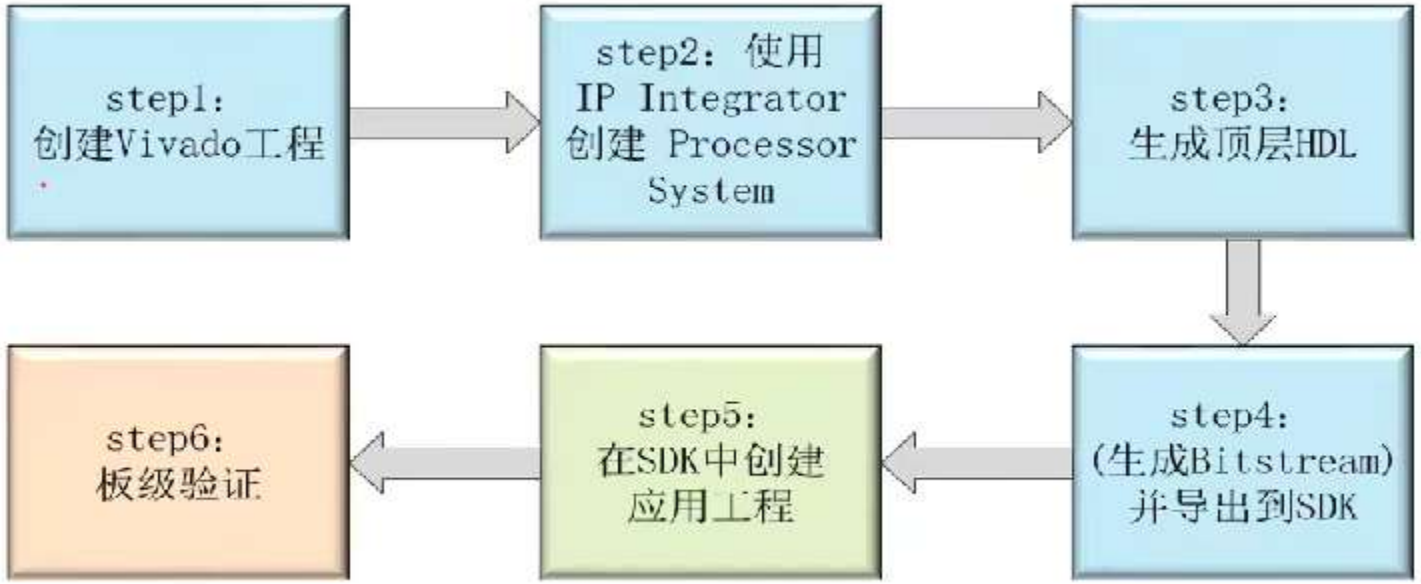
1. **Set the direction as input**: Write `0x0` to the gpio.DIRM_0 register. This sets gpio.DIRM_0[10] = `0`.

# Purpose

- Use MIO to light the LED.




System Block diagram



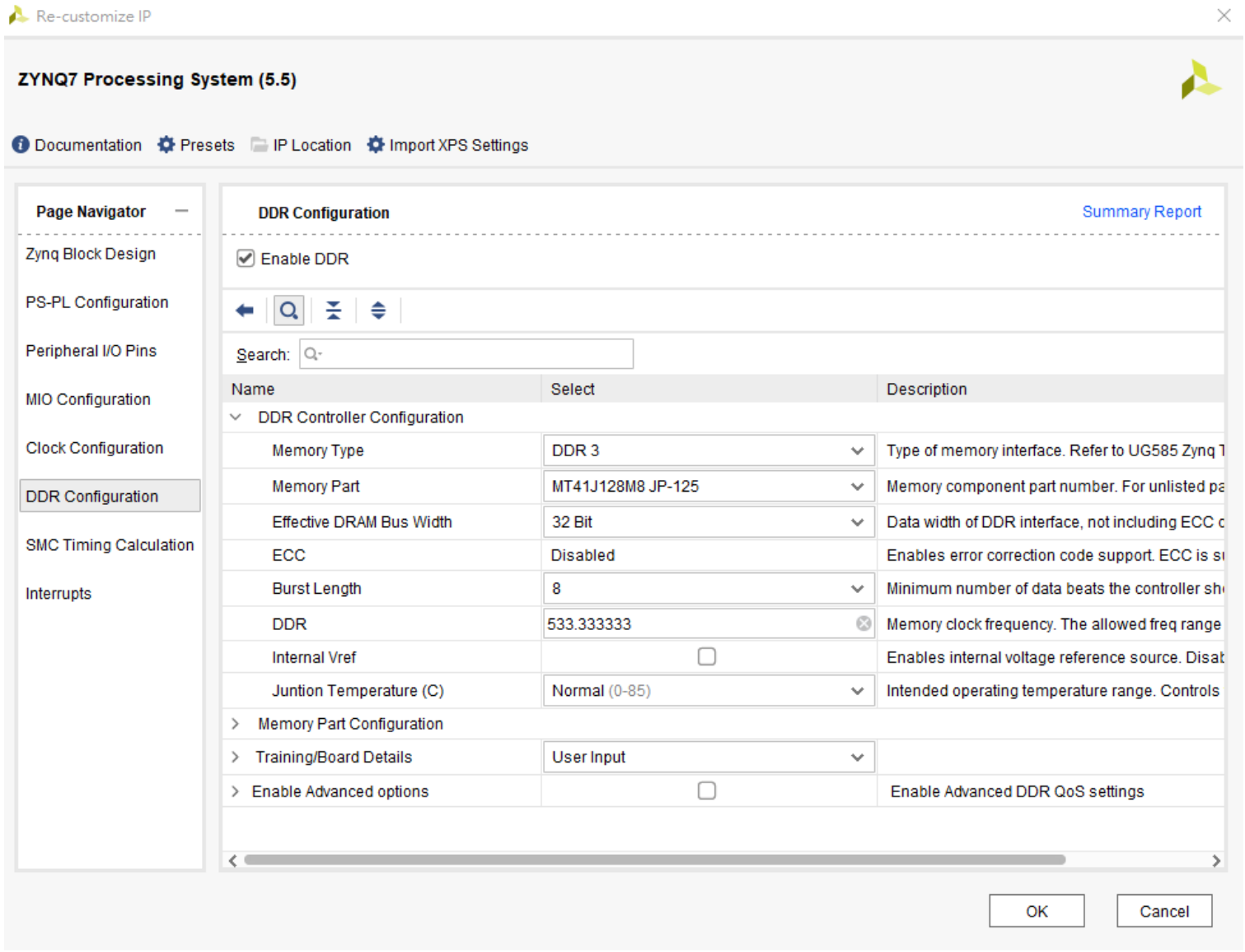

Design Flow

# 1.Create Vivado project – create Block Design – Zynq

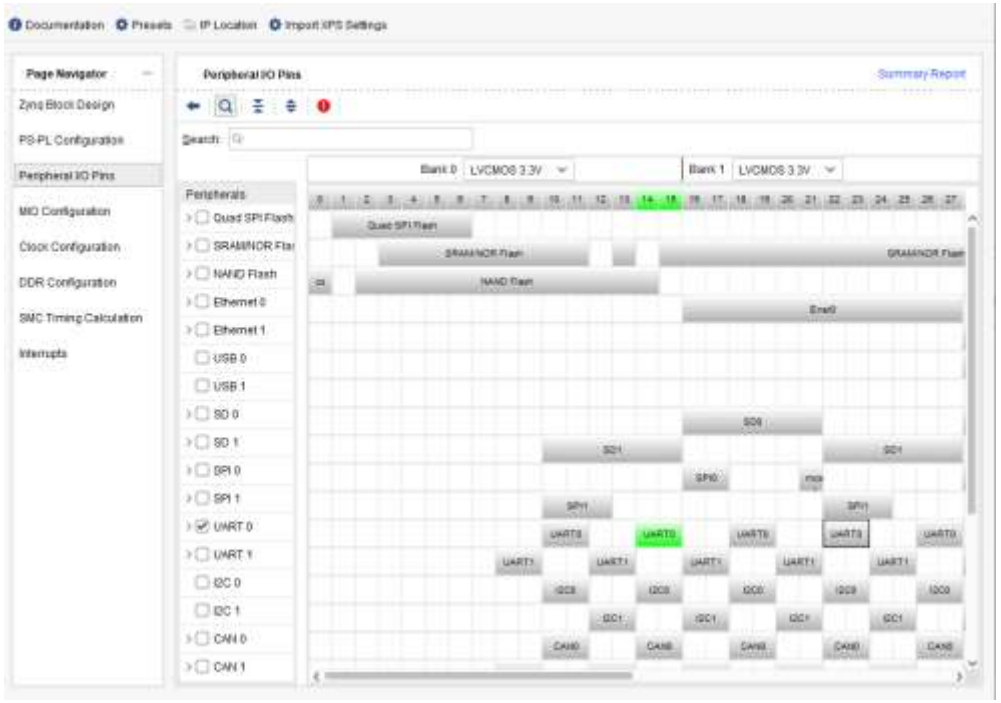


We just using MIO , so delete redundant ports ，M_AXI_GP0_ACLK,M_AXI_GP0,FCLK_CLK,FCLK_RESET0_N

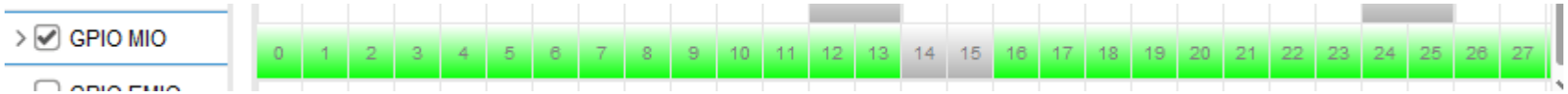
# DDR port

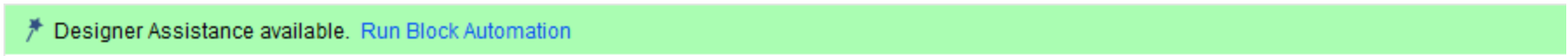


# Fixed_IO

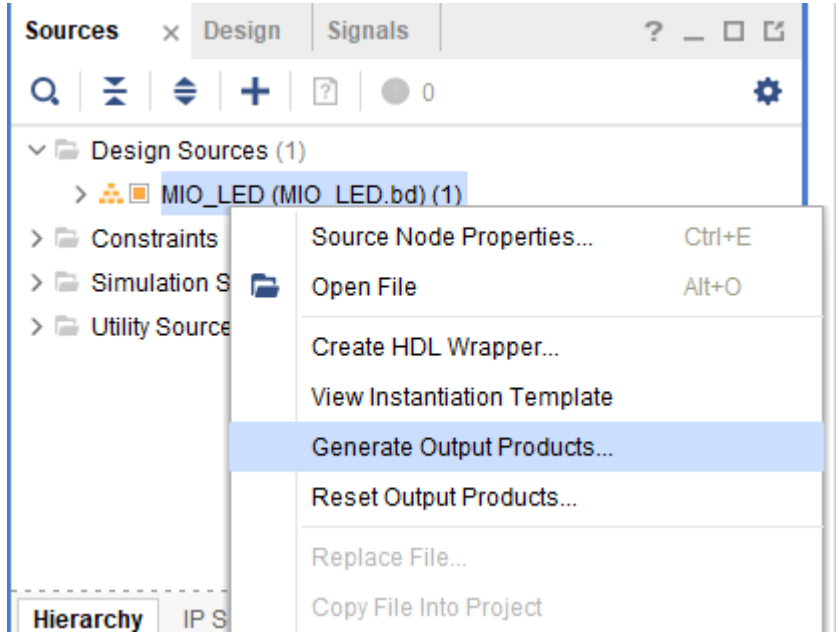


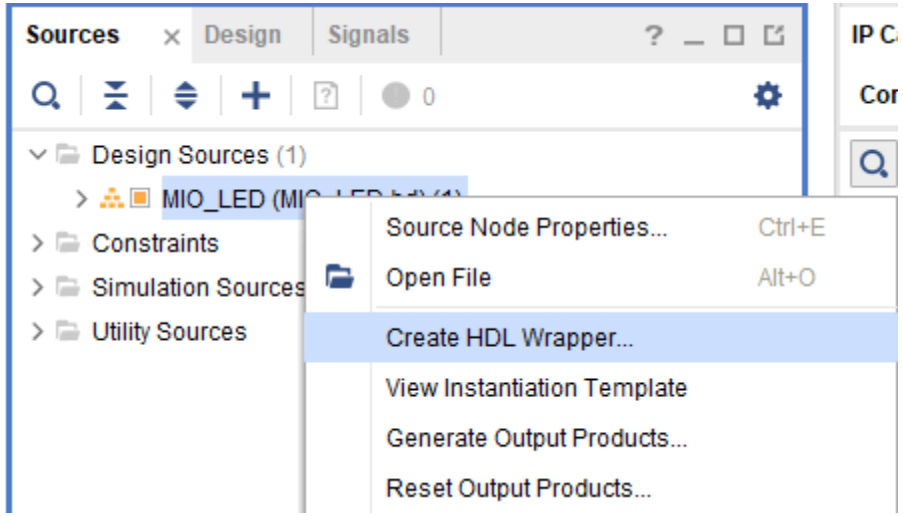

Configurate the port, using UART0, **click the GPIO MIO,** than click OK, next step click the " Run Block Automation"



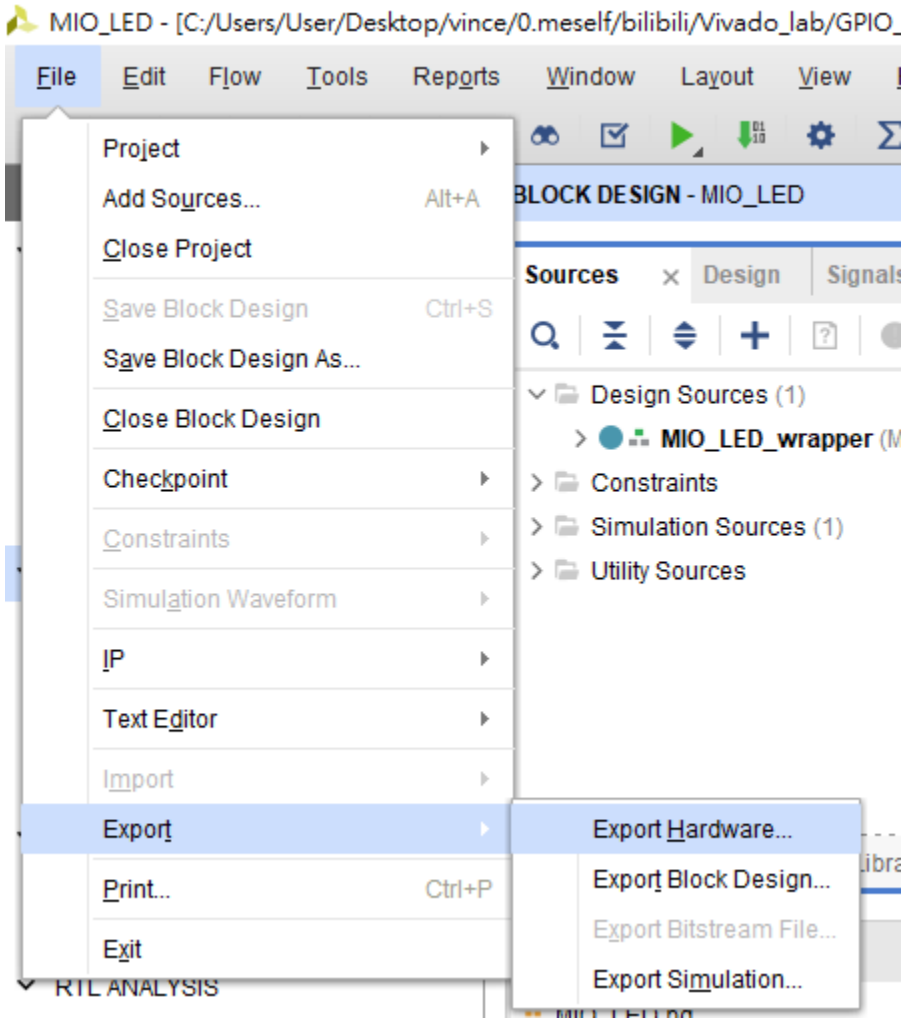



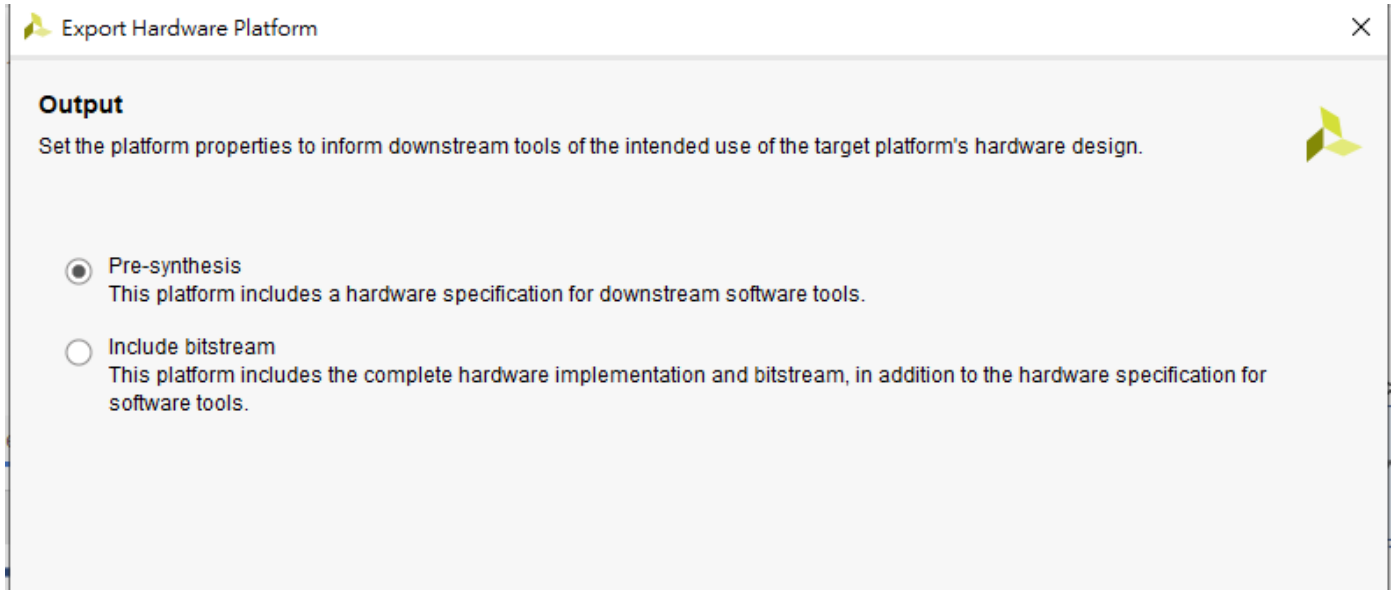
# Generate Output Product & Create HDL Wrapper



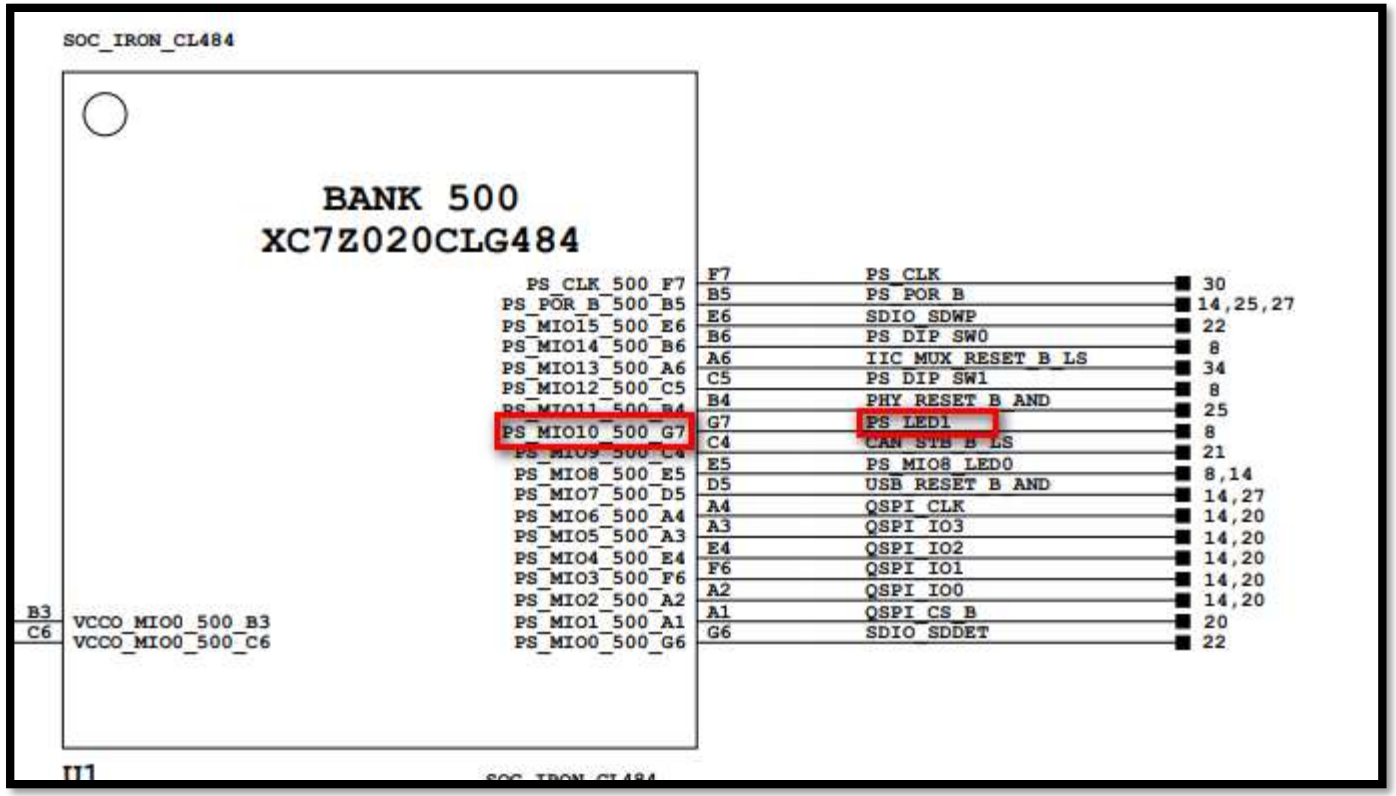



- We just using PS , doesn't have PL, so doesn't need to Generate Bitstream



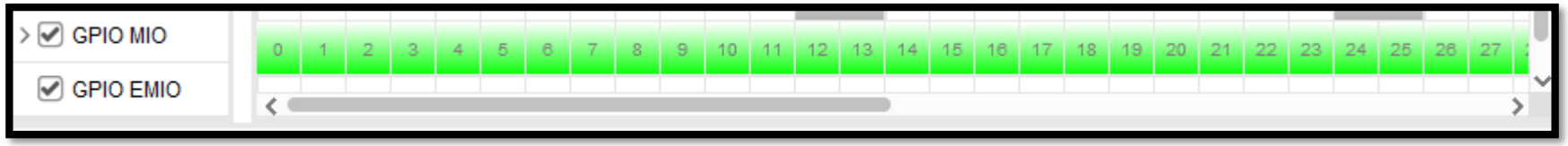
```
XGpioPs_SetDirectionPin(&Gpio, Output_Pin, gpio_output);
```

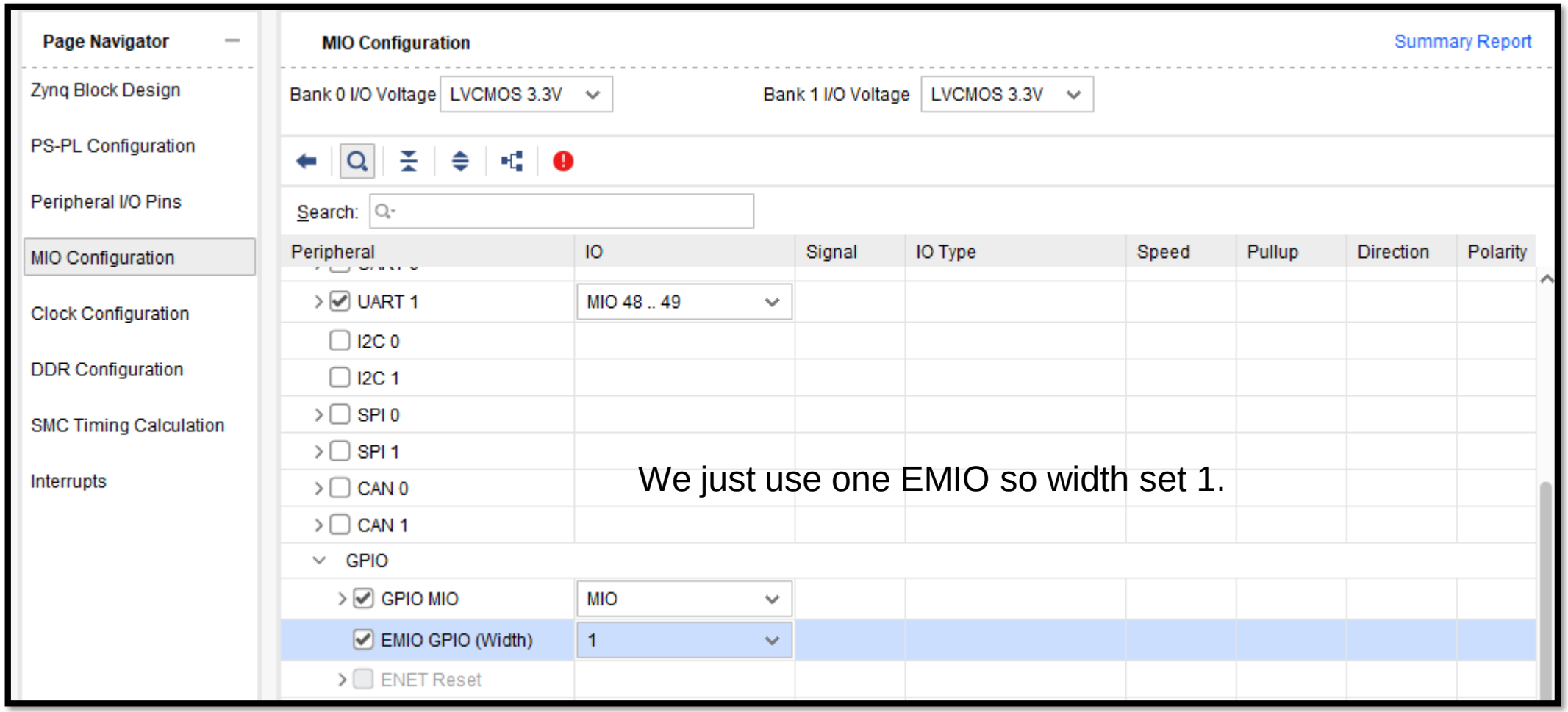


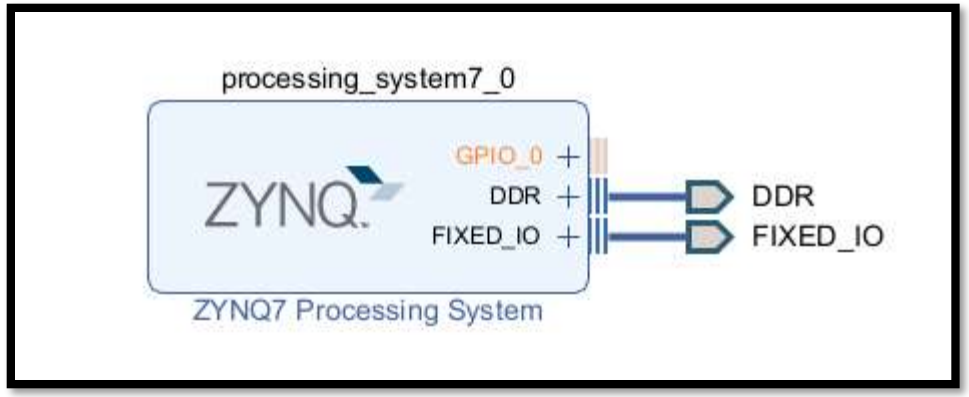

```
#define PS_LED1         10
```

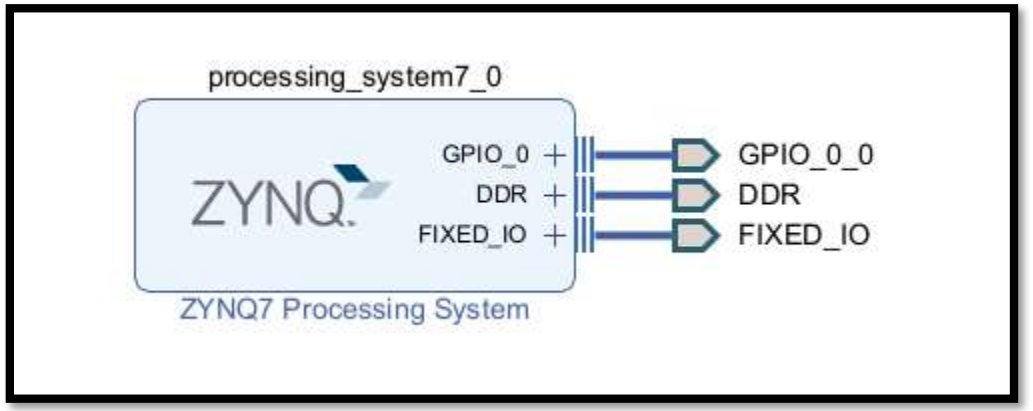
```
XGpioPs_SetDirectionPin(&Gpio, PS_LED1, gpio_output);
```

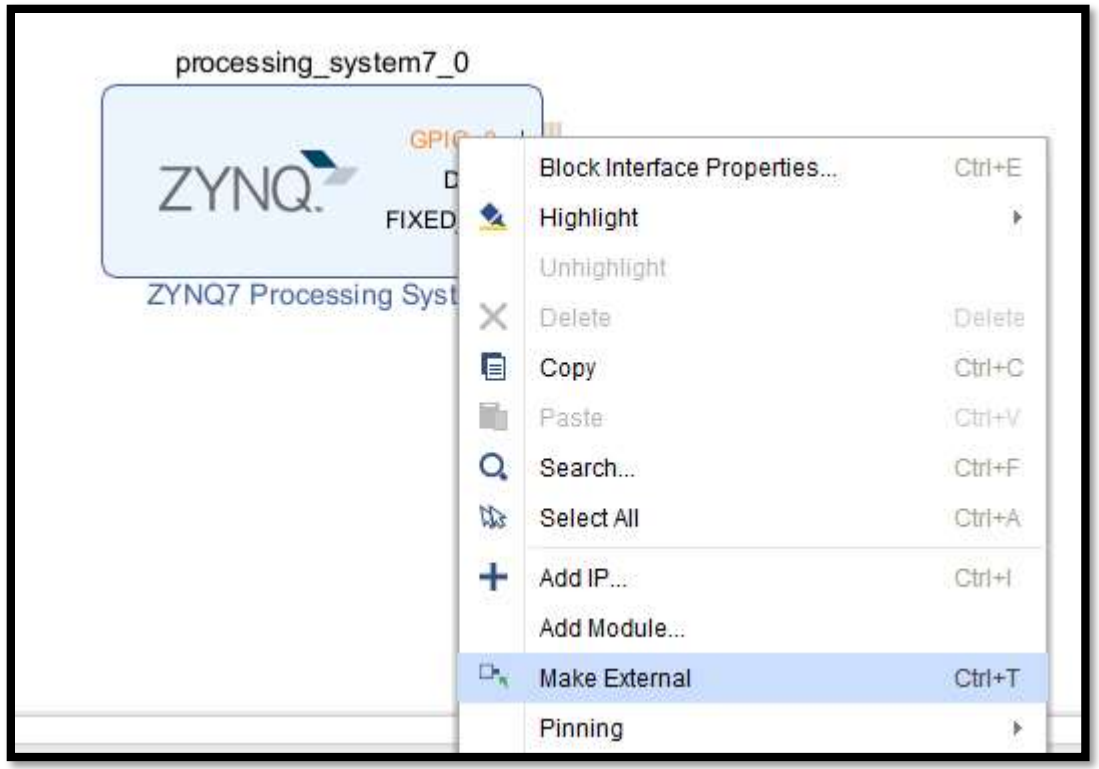
# EMIO

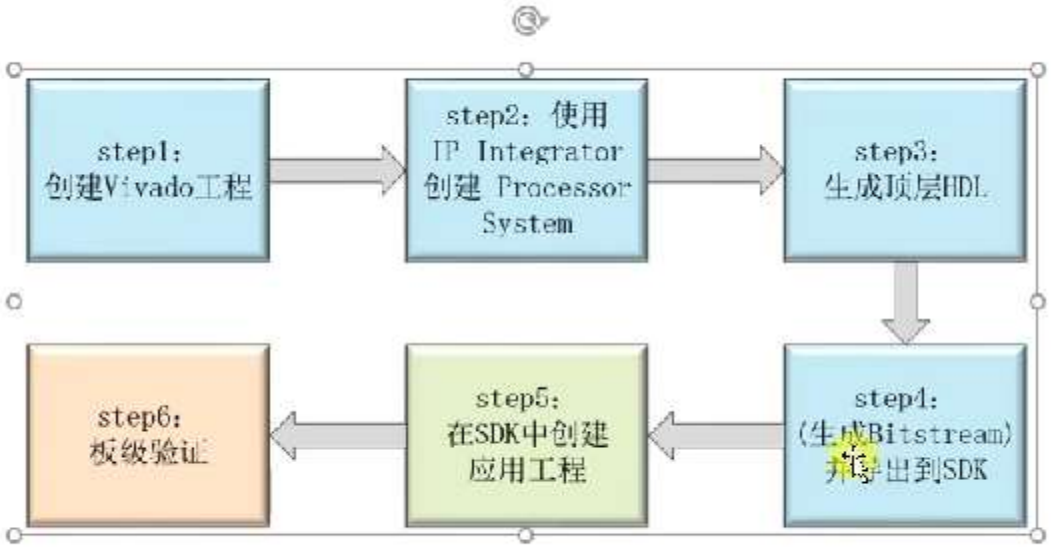


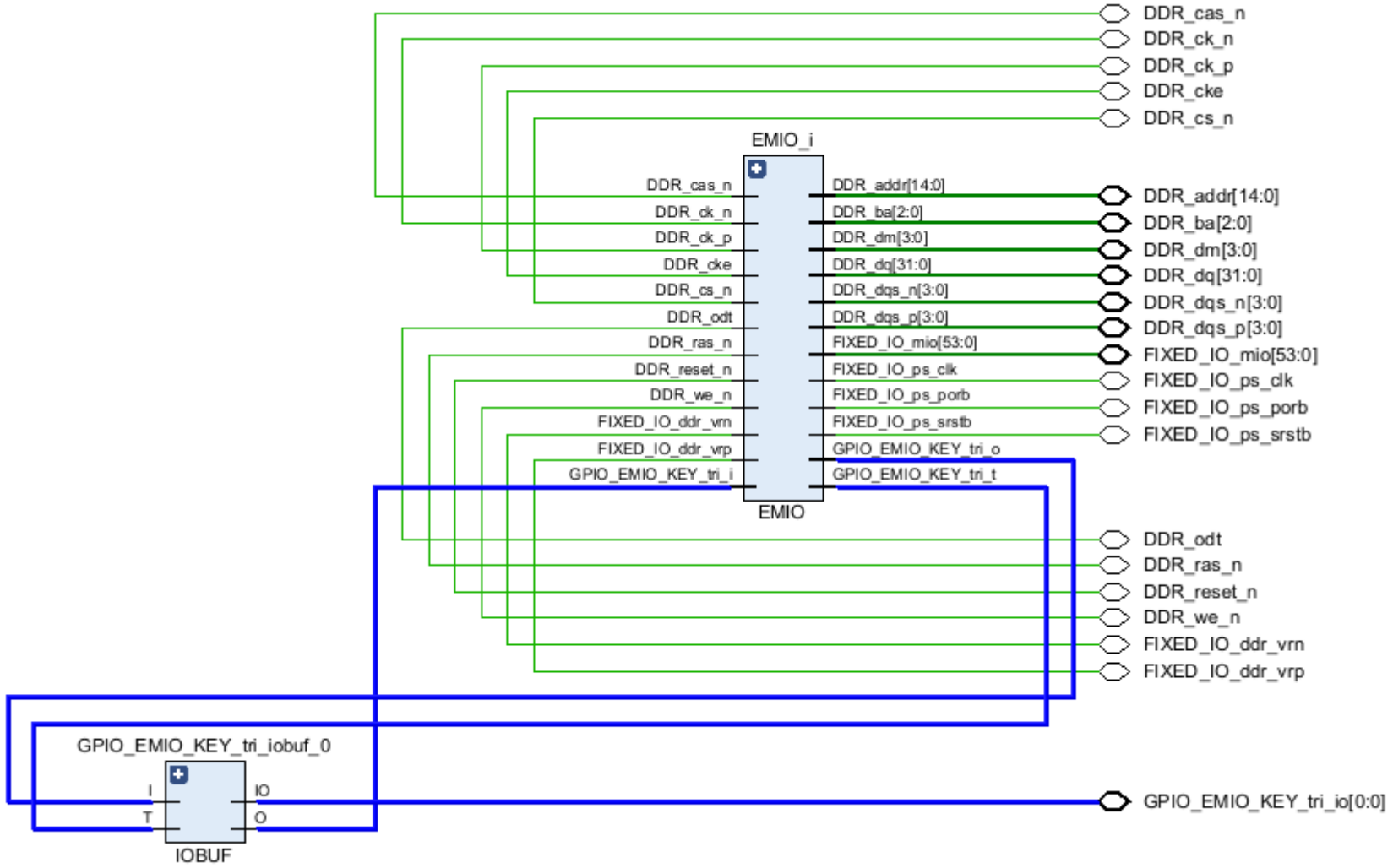





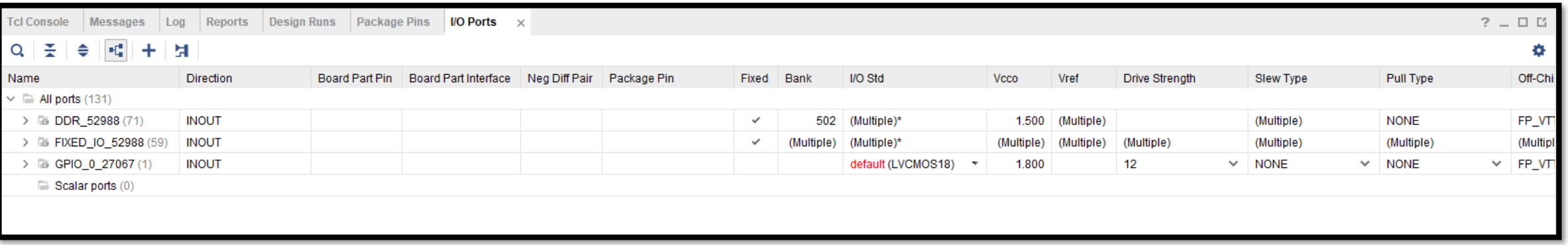



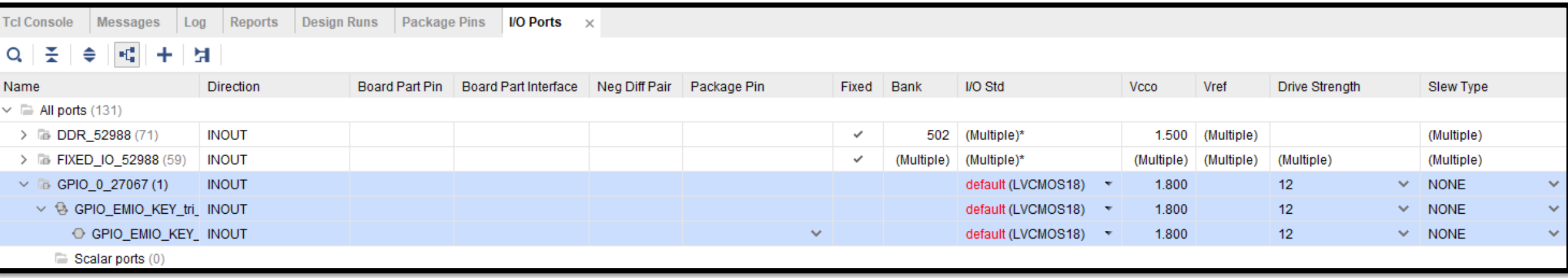


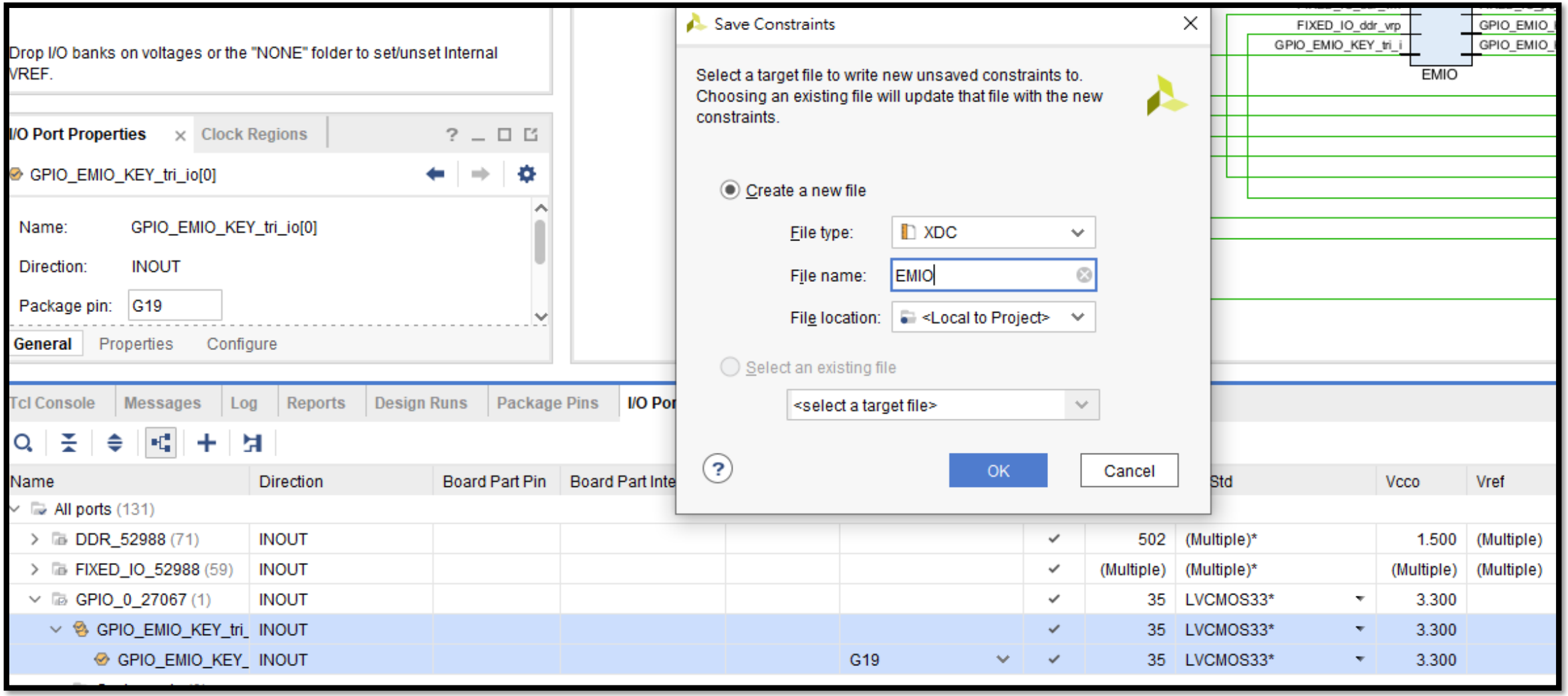


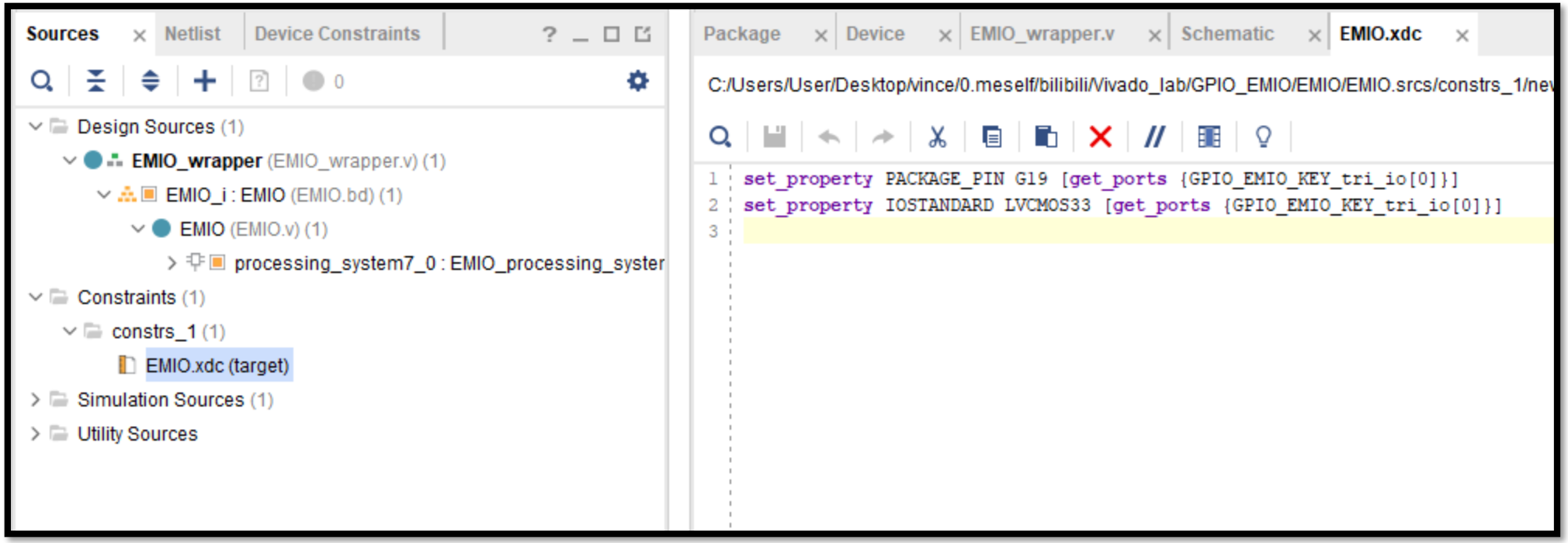

Because we use EMIO need to constraints the port.

# Layout > IO planning







```
#define PL_EMIO         54
```

```
    XGpioPs_SetDirectionPin(&Gpio, PS_LED1, gpio_output);
    XGpioPs_SetDirectionPin(&Gpio, PL_EMIO, gpio_input);
     /*
      * Enable the Output enable for the LED Pin.
      */
    XGpioPs_SetOutputEnablePin(&Gpio, PS_LED1, Enable);

    /* Set the GPIO output */

    print("Set the GPIO output Successfully \n\r");



    while(1){
        Readdata =  XGpioPs_ReadPin(&Gpio, PL_EMIO);
        XGpioPs_WritePin(&Gpio, PS_LED1, Readdata);
        printf("LED, statue is %d \n\r",Readdata);
        sleep(1);
    }
    return 0;
    }
```


AMD