



MIO/EMIO Controller LED

GPIO MIO CONTROLL led

STEP

1. GPIO INTRODUCTION
2. HARDWARE 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被配置為輸出的時候 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。

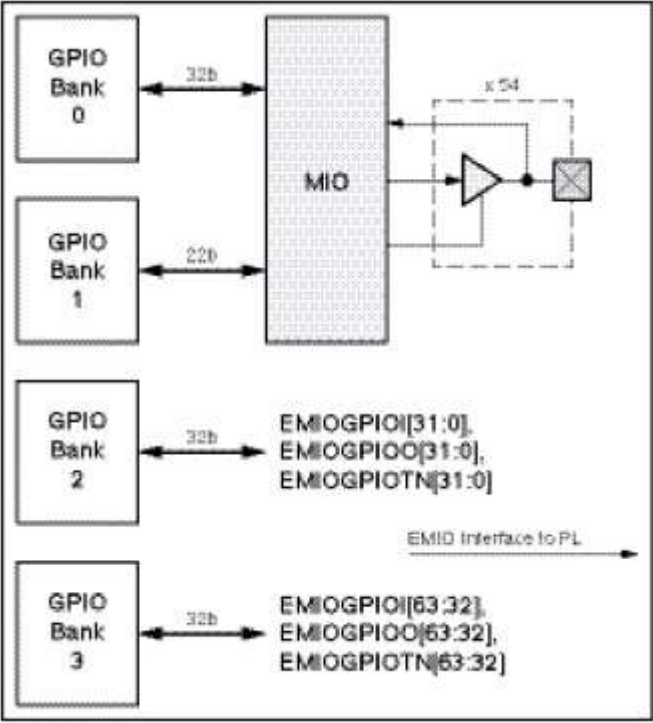
MASK_DATA_LSW 用於屏蔽DATA的低16位元。

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.

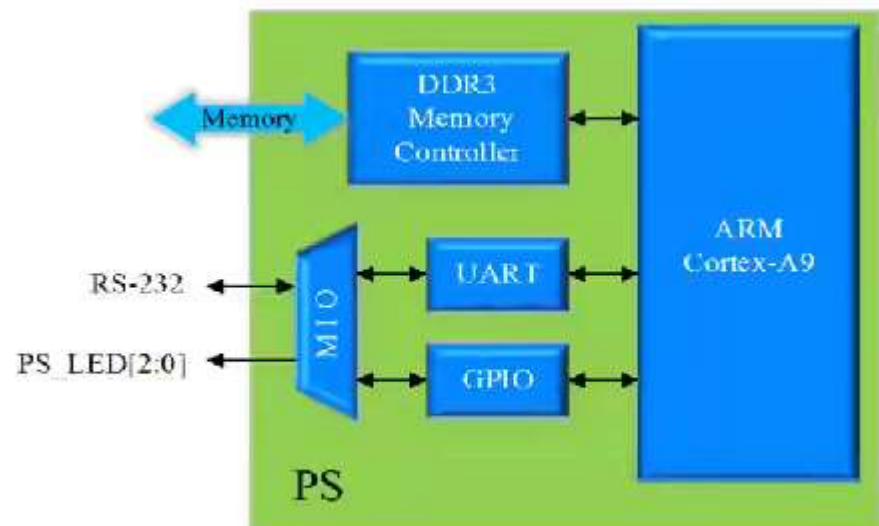
Note: The output enable has significance only when the GPIO pin is configured as an output.

Example: Configure MIO pin 10 as an input

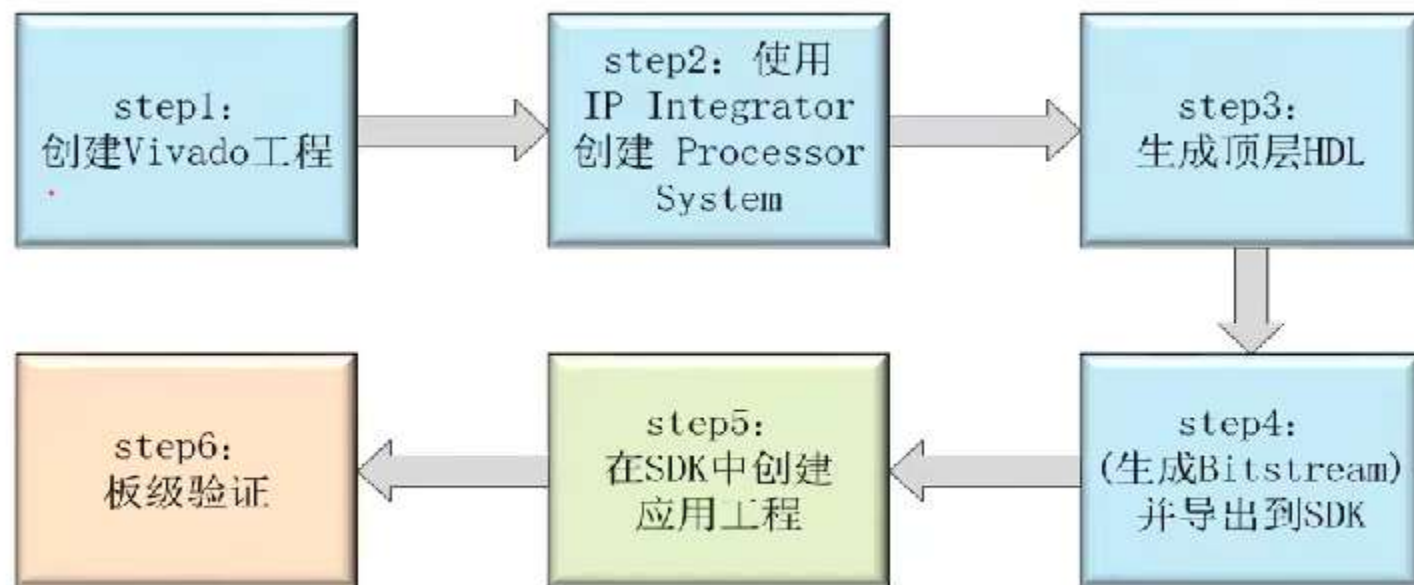
1. **Set the direction as input:** Write 0x0 to the gpio.DIRM_0 register. This sets gpio.DIRM_0[10] = 0.

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_RESE
T0_N



DDR port

Re-customize IP

ZYNQ7 Processing System (5.5)

Documentation

Presets

IP Location

Import XPS Settings

Page Navigator

Zynq Block Design

PS-PL Configuration

Peripheral I/O Pins

MIO Configuration

Clock Configuration

DDR Configuration

SMC Timing Calculation

Interrupts

DDR Configuration

Summary Report

Enable DDR

←

🔍

⌵

⌶

Search: 🔍

Name	Select	Description
▼ DDR Controller Configuration		
Memory Type	DDR 3	Type of memory interface. Refer to UG585 Zynq 1
Memory Part	MT41J128M8 JP-125	Memory component part number. For unlisted pa
Effective DRAM Bus Width	32 Bit	Data width of DDR interface, not including ECC c
ECC	Disabled	Enables error correction code support. ECC is s
Burst Length	8	Minimum number of data beats the controller sho
DDR	533.333333	Memory clock frequency. The allowed freq range
Internal Vref	<input type="checkbox"/>	Enables internal voltage reference source. Disab
Junction Temperature (C)	Normal (0-85)	Intended operating temperature range. Controls
> Memory Part Configuration		
> Training/Board Details	User Input	
> Enable Advanced options	<input type="checkbox"/>	Enable Advanced DDR QoS settings

OK

Cancel

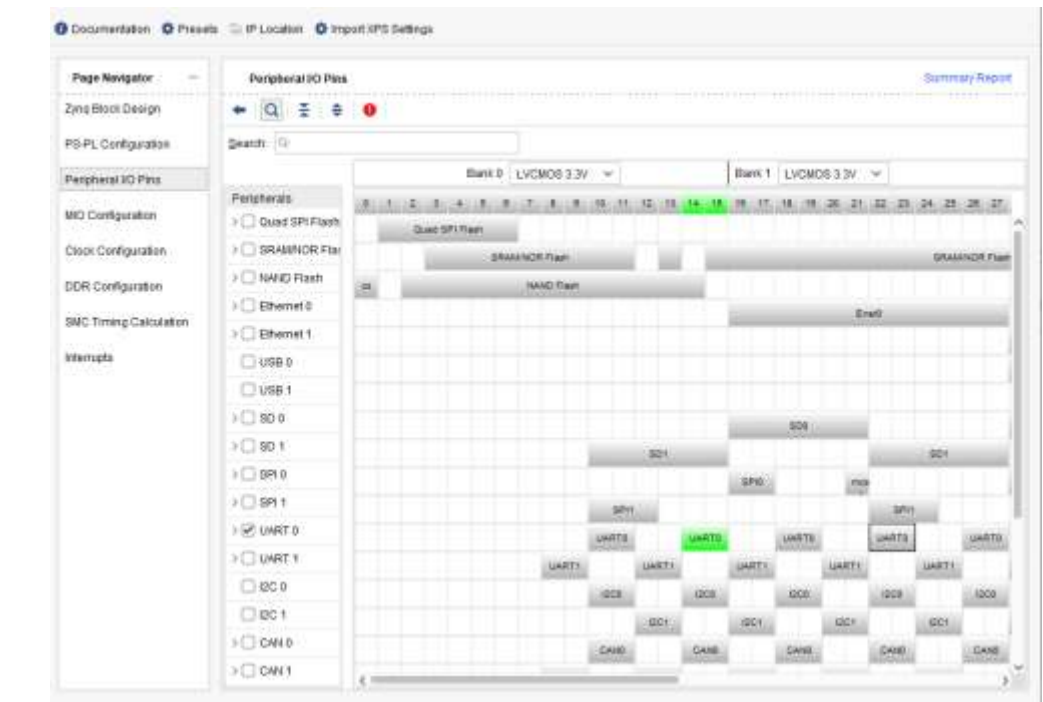
8

© Copyright 2023 Advanced Micro Devices, Inc.

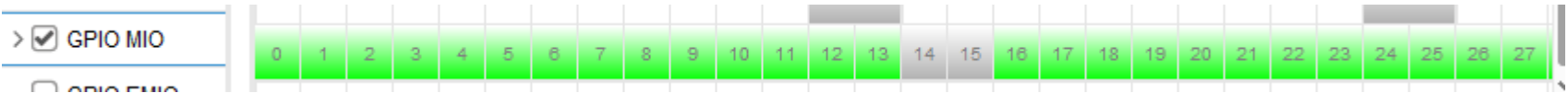
AMD

together we advance_

Fixed_IO

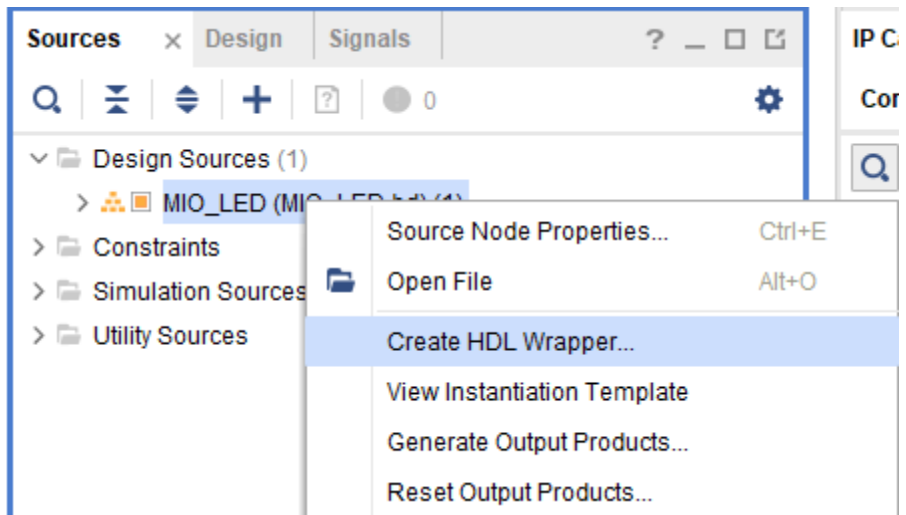
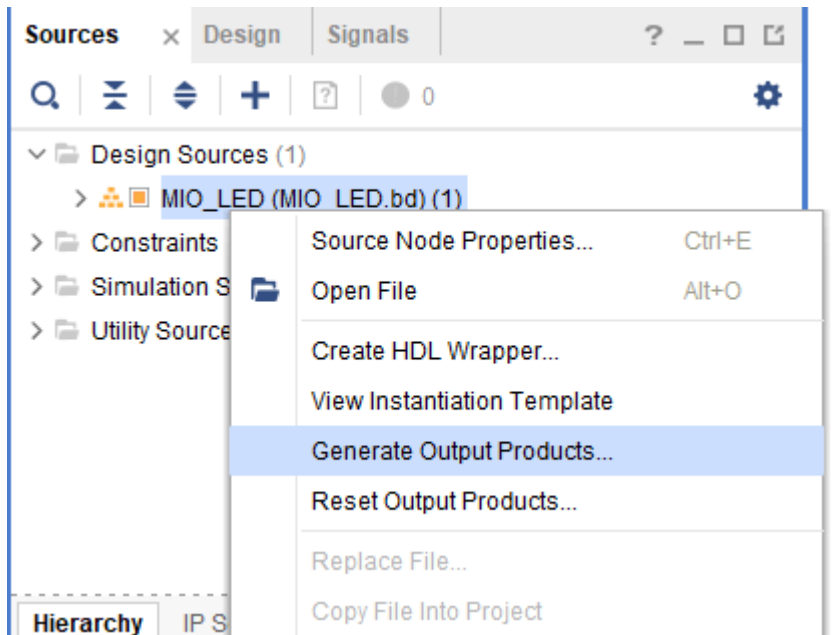


Configure the port, using UART0, **click the GPIO MIO**, than click OK, next step click the “ Run Block Automation”

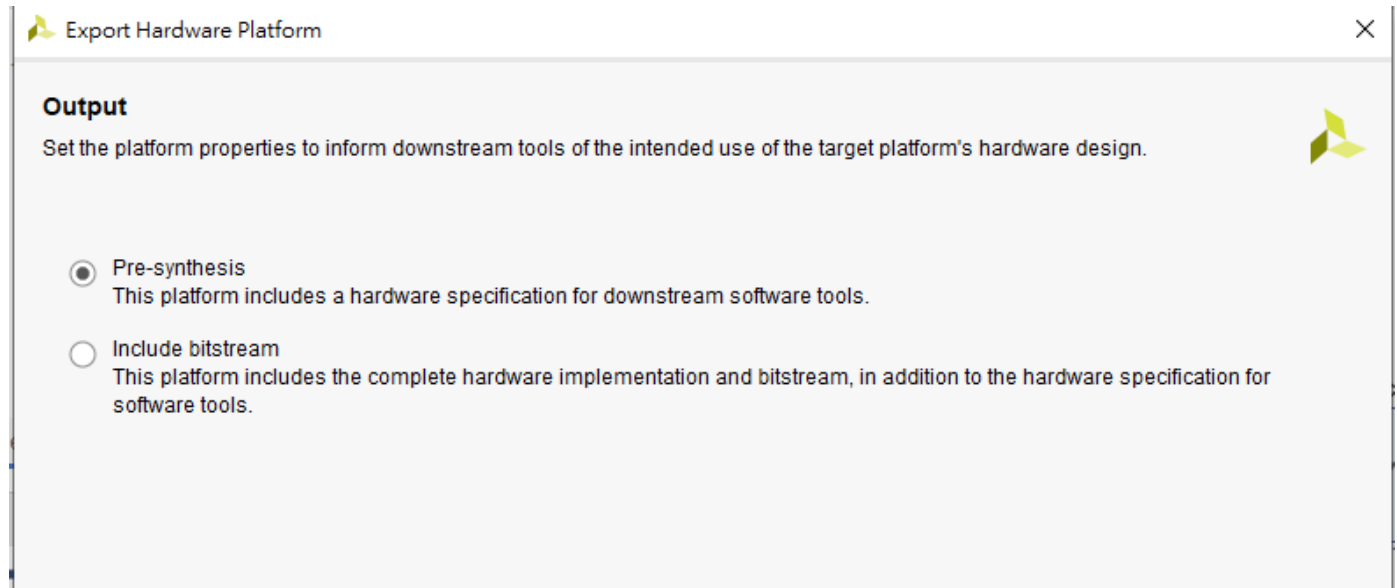
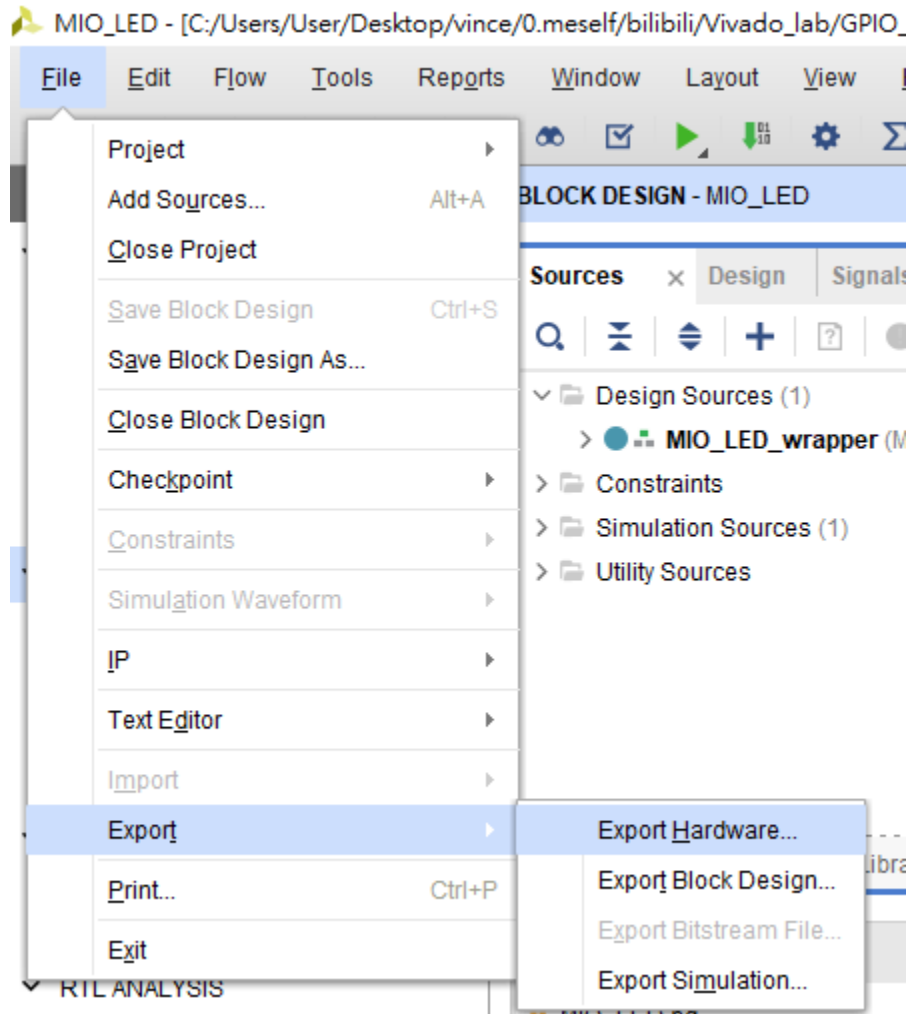


★ Designer Assistance available. [Run Block Automation](#)

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);
```

SOC_IRON_CL484

BANK 500
XC7Z020CLG484

PS_CLK_500_F7	F7	PS_CLK	30
PS_POR_B_500_B5	B5	PS POR B	14, 25, 27
PS_MIO15_500_E6	E6	SDIO SDWP	22
PS_MIO14_500_B6	B6	PS DIP SW0	8
PS_MIO13_500_A6	A6	IIC MUX RESET B_LS	34
PS_MIO12_500_C5	C5	PS DIP SW1	8
PS_MIO11_500_B4	B4	PHY RESET B AND	25
PS_MIO10_500_G7	G7	PS_LED1	8
PS_MIO9_500_C4	C4	CAN STB B_LS	21
PS_MIO8_500_E5	E5	PS MIO8 LED0	8, 14
PS_MIO7_500_D5	D5	USB RESET B AND	14, 27
PS_MIO6_500_A4	A4	QSPI CLK	14, 20
PS_MIO5_500_A3	A3	QSPI IO3	14, 20
PS_MIO4_500_E4	E4	QSPI IO2	14, 20
PS_MIO3_500_F6	F6	QSPI IO1	14, 20
PS_MIO2_500_A2	A2	QSPI IO0	14, 20
PS_MIO1_500_A1	A1	QSPI CS B	20
PS_MIO0_500_G6	G6	SDIO SDDET	22

B3 VCCO_MIO0_500_B3
C6 VCCO_MIO0_500_C6

U1

SOC_IRON_CL484

```
#define PS_LED1 10
```

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

EMIO

> ☒ GPIO MIO

☒ GPIO EMIO

0123456789101112131415161718192021222324252627

Page Navigator

Zynq Block Design

PS-PL Configuration

Peripheral I/O Pins

MIO Configuration

Clock Configuration

DDR Configuration

SMC Timing Calculation

Interrupts

MIO Configuration

Summary Report

Bank 0 I/O Voltage

LVC MOS 3.3V

Bank 1 I/O Voltage

LVC MOS 3.3V

←

🔍

↕

🔄

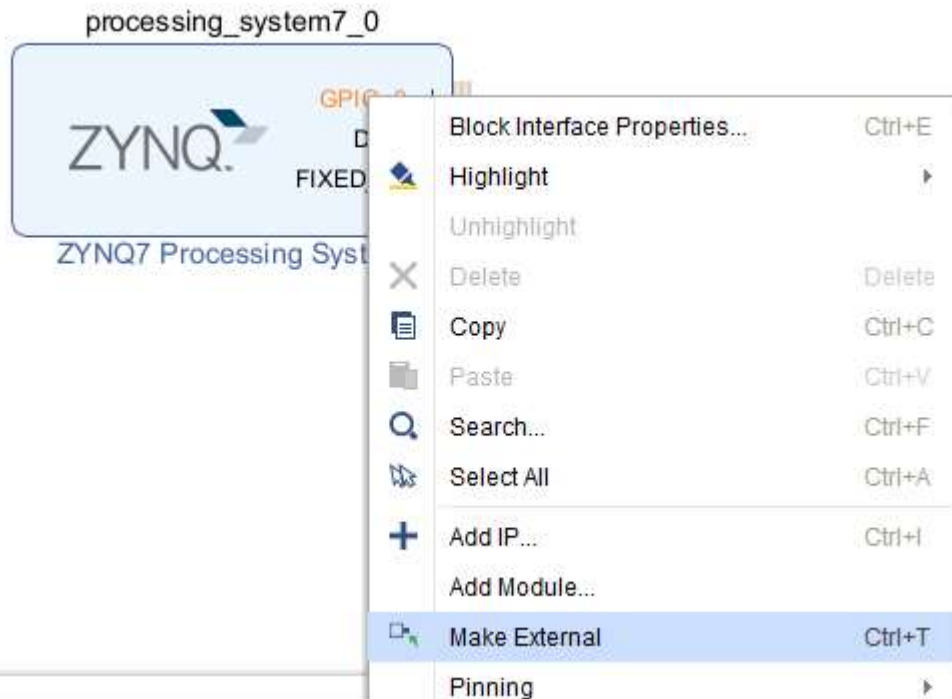
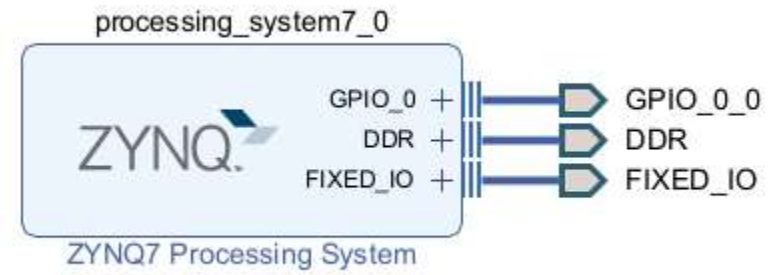
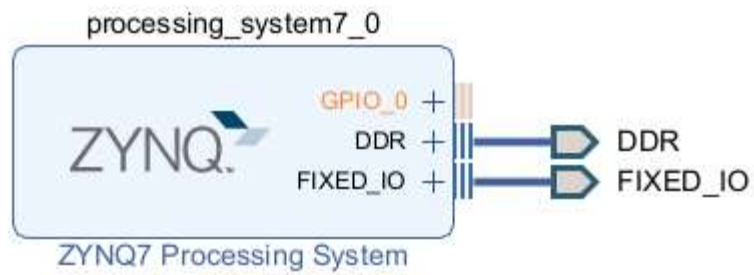
🔧

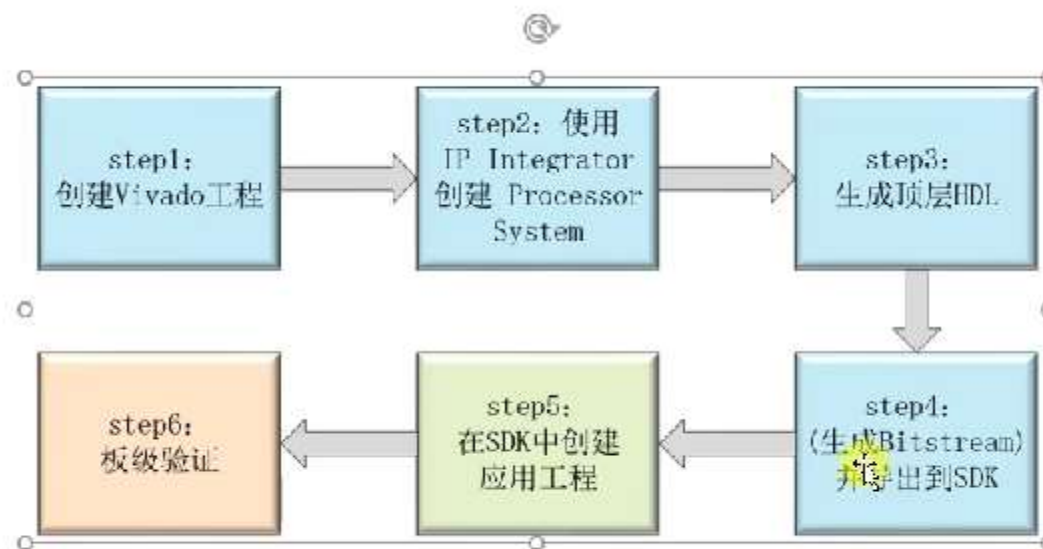
!

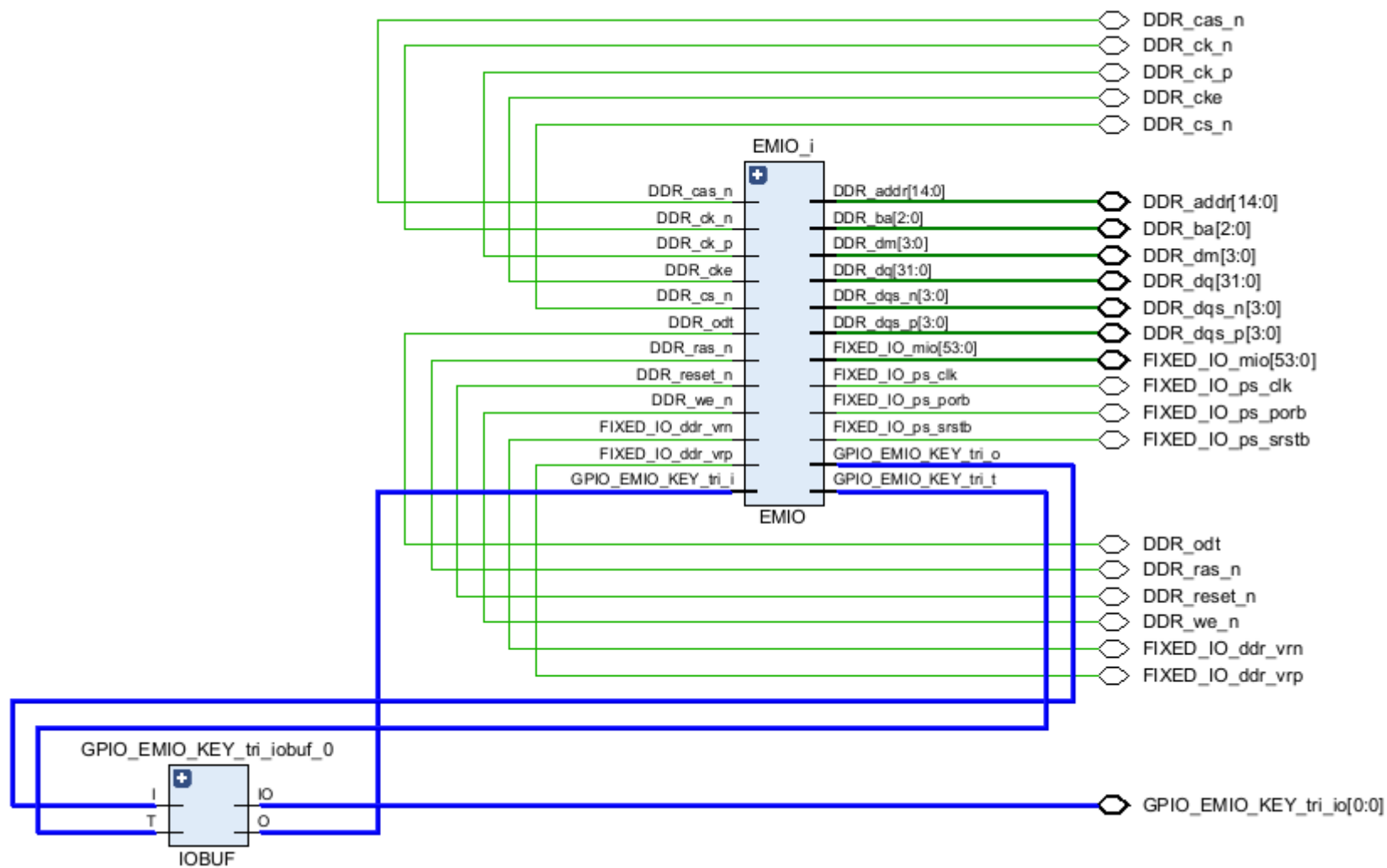
Search:

Peripheral	IO	Signal	IO Type	Speed	Pullup	Direction	Polarity
> <input checked="" type="checkbox"/> UART 1	MIO 48 .. 49						
<input type="checkbox"/> I2C 0							
<input type="checkbox"/> I2C 1							
> <input type="checkbox"/> SPI 0							
> <input type="checkbox"/> SPI 1							
> <input type="checkbox"/> CAN 0							
> <input type="checkbox"/> CAN 1							
▼ GPIO							
> <input checked="" type="checkbox"/> GPIO MIO	MIO						
<input checked="" type="checkbox"/> EMIO GPIO (Width)	1						
> <input type="checkbox"/> ENET Reset							

We just use one EMIO so width set 1.



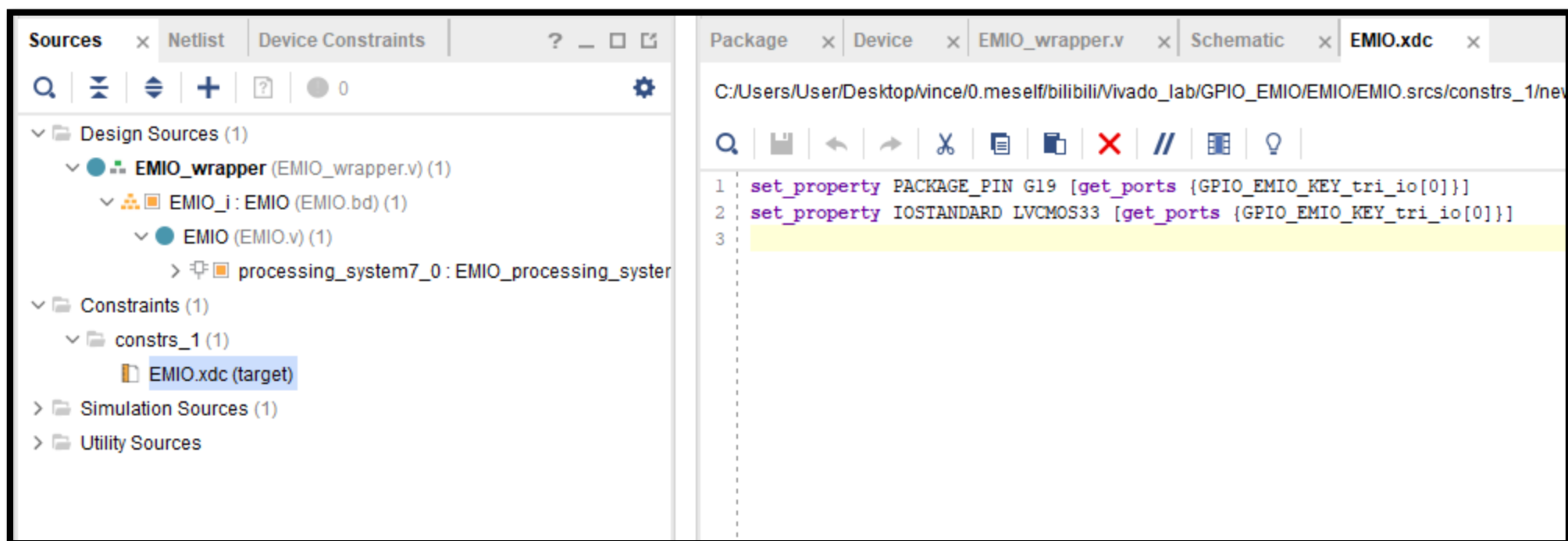




Because we use EMIO need to constraints the port.

Layout > IO planning

[illegible][illegible]



```
#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;
}
```

