芯动力——硬件加速设计方法

第七章 基于平头哥E902处理器的SoC设计

——(5) 基于NexysVideo板卡的FPGA实现和I/O LAB实验

邸志雄@西南交通大学

zxdi@home.swjtu.edu.cn

slides与源代码网址 http://www.dizhixiong.cn/class5/

本课程SOC教学中所有实验内容都使用Nexys Video板卡完成。

只要FPGA板卡的资源能够承载无剑SoC，都可以用来学习与实践本课程的内容。

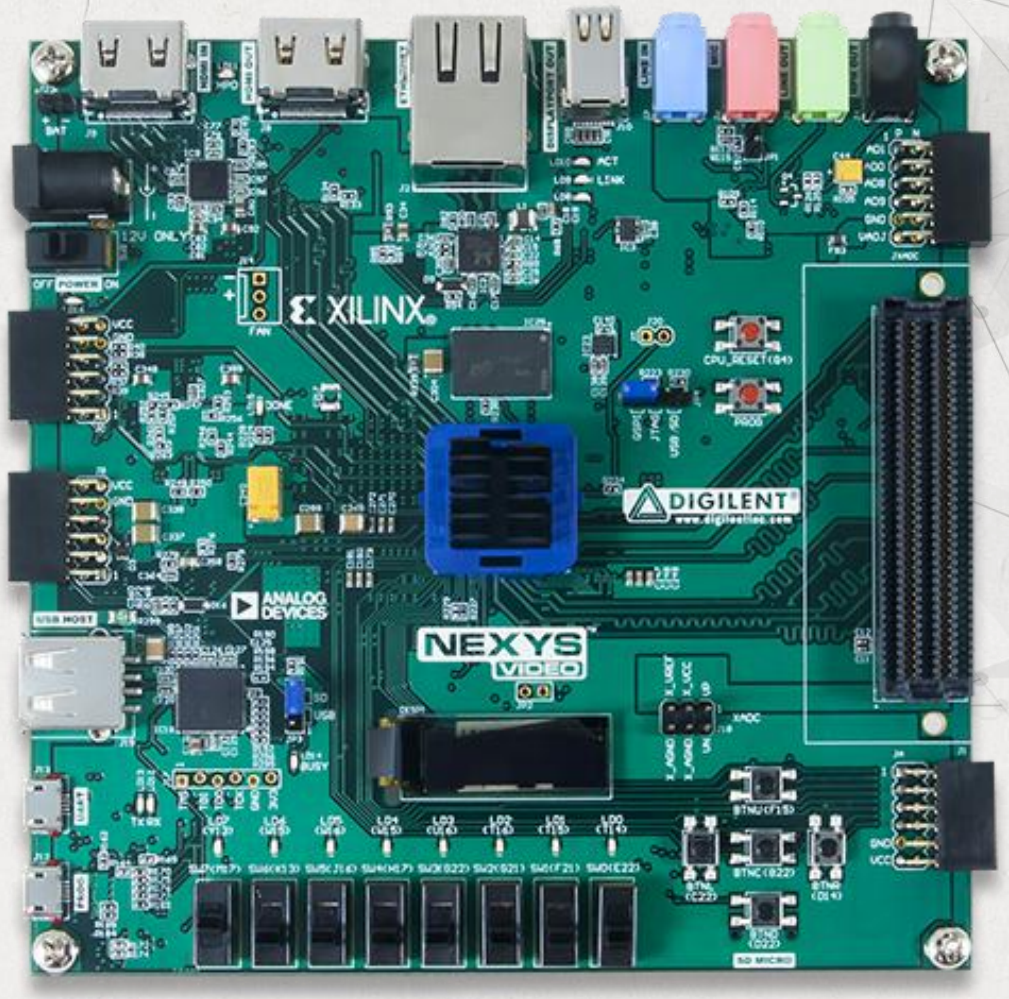
IO LAB实验

- 串口
- OLED
- LED
- SD卡读写
- 按键中断驱动

NexyVideo板卡介绍

NexysVideo是Digilent 公司专为音视频应用设计的板卡，搭载了Xilinx® Artix-7系列中功能最强大的芯片Artix®-7 FPGA XC7A200T，资源容量非常大。

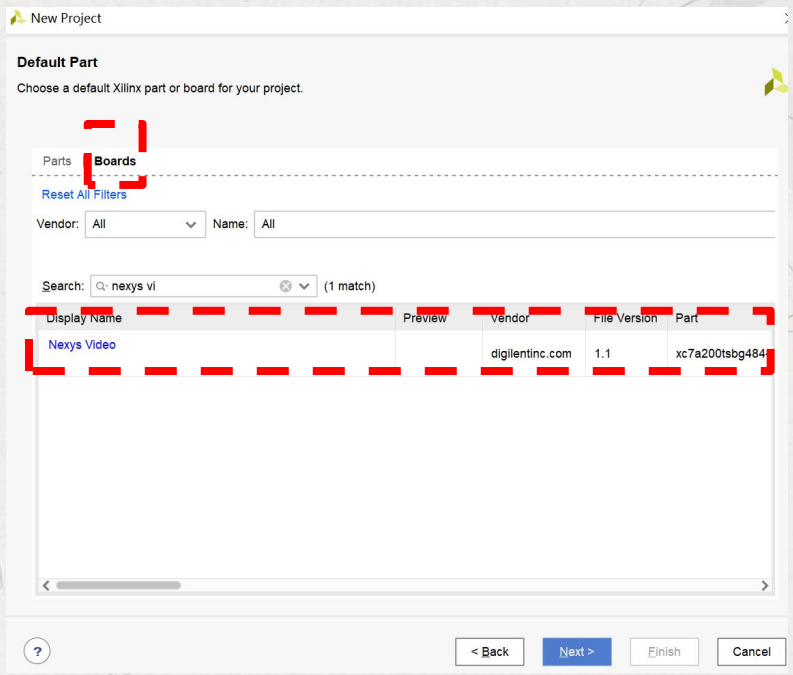
- OLED
- SD卡槽
- PMOD接口



FPGA实现流程

①添加board_file板级支持文件

Digilent官方有该FPGA板卡的板级支持文件，在建立工程时即可选择板卡配置而无需对具体的FPGA芯片型号进行选择。除此之外还提供了每个板卡的xdc约束脚本文件，便于进行信号与引脚的配置。上述文件Digilent均在github上开源。



FPGA实现流程

②添加设计源文件

wujian100 开源工程　Synplify　+　Vivado

通过Synplify对SoC进行综合，然后把其综合结果edf网表文件在Vivado上进行布局布线，最终生成bitstream文件。使用Synplify的原因在于Synplify是功能超强的FPGA综合软件，是业界领先的FPGA逻辑综合解决方案。

FPGA实现流程 ②添加设计源文件

wujian100 开源工程 Synplify + Vivado

Synplify

- 针对于工程
- 针对于FPGA器件

```
#project files
add_file -verilog "../wujian100_open_fpga_top.v"
add_file -verilog "../../soc/ahb_matrix_top.v"
add_file -verilog "../../soc/smu_top.v"
add_file -verilog "../../soc/sms.v"
add_file -verilog "../../soc/ls_sub_top.v"
add_file -verilog "../../soc/retu_top.v"
add_file -verilog "../../soc/tim5.v"
add_file -verilog "../../soc/tim.v"
add_file -verilog "../../soc/dmac.v"
add_file -verilog "../../soc/pdu_top.v"
                .
                .
                .
set_option -project_relative_includes 1
set_option -include_path {../../soc/params/}
add_file -constraint "wujian100_open.sdc"
```

综合时需要添加的设计文件(Synplify工程文件)

Vivado模板工程设计文件

③板卡适配

- 时钟信号
- 复位信号
- JTAG调试信号
- 串口

- 这些约束最终需要放在xdc约束脚本中与设计源文件一起综合实现。

FPGA实现流程

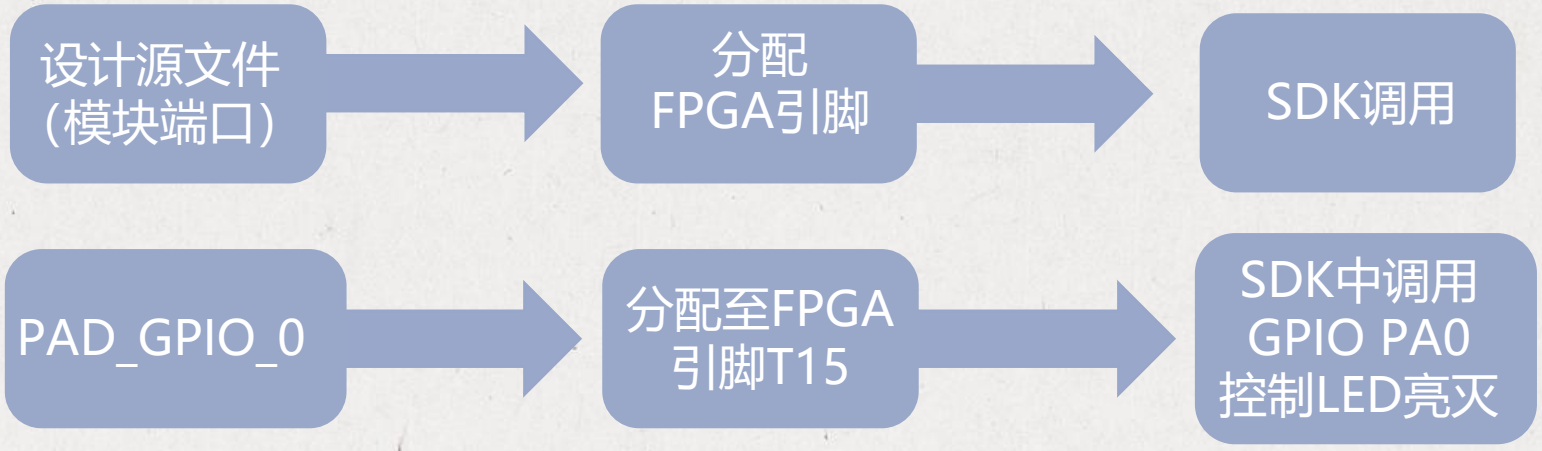
③板卡适配

- 时钟信号
- 时钟信号
- 复位信号
- JTAG调试信号
- 串口

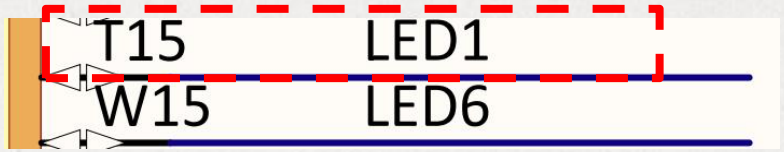
这些约束最终需要放在xdc约束脚本中与设计源文件一起综合实现。

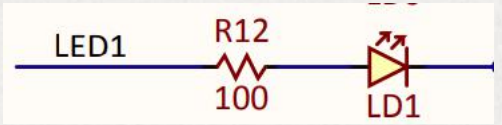
FPGA实现流程

③板卡适配



引脚的映射关系





硬件电路连接

以GPIO_0为例，在xdc约束脚本中将其在SoC设计的信号分配至硬件上与LED相连接的FPGA引脚，再通过软件中的GPIO PA0完成对GPIO_0的控制，从而实现LED亮灭的控制。硬件实现和软件控制上要保证一致性。

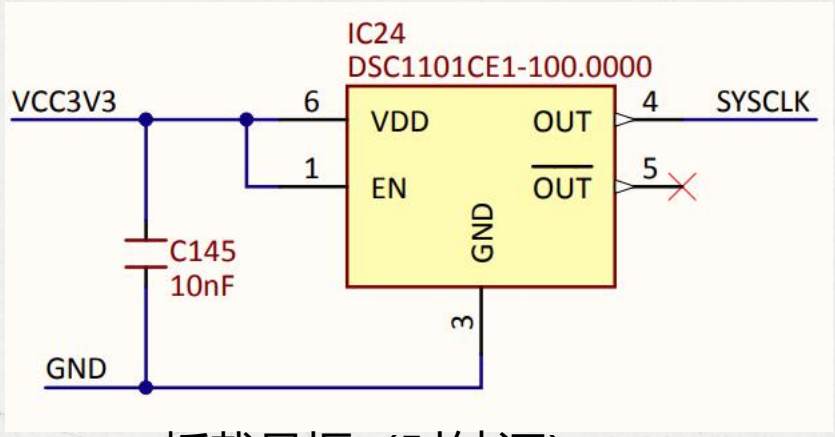
FPGA实现流程

③板卡适配：时钟信号

时钟信号的分配

- 板载晶振为100MHz
- wujian100 SoC在Nexy Video板卡的FPGA型号上工作最高主频为20MHz

这里需要采用Vivado clocking wizard IP进行分频，再将分频后20MHz的时钟信号送入PIN_EHS信号提供给SoC。该分频模块的复位信号由系统复位输入提供，低电平复位。



板载晶振（时钟源）

clocking wizard IP

FPGA实现流程

③板卡适配：时钟信号

注意

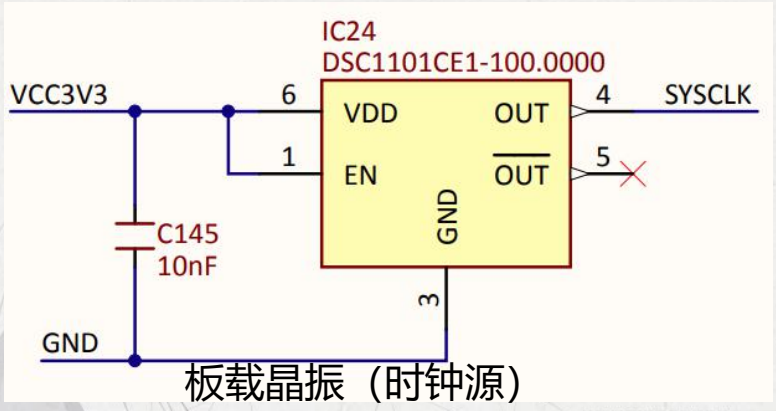
时钟信号的产生需要使用时钟分频模块，一定要用clocking wizard IP，不要使用自己用counter实现的时钟分频模块。

FPGA内部全局时钟布线资源

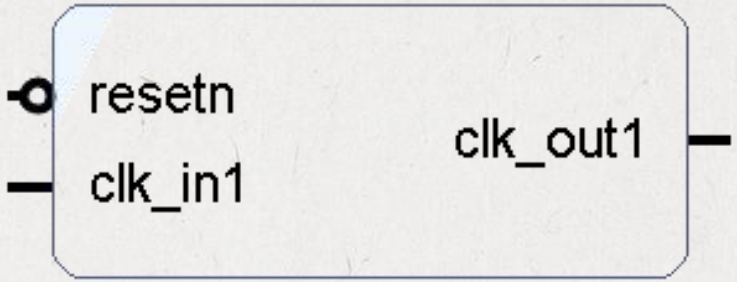
时钟布线资源较少但是在整个芯片上传输非常快、相位延迟也很低，输出信号和输入时钟有固定的相位关系；

逻辑布线资源

逻辑资源分布在整个芯片上，延时大、时钟歪斜严重，并且在综合实现过程中，因优化程度不同每次时序关系都会发生改变。



板载晶振（时钟源）



clocking wizard IP

FPGA实现流程

③板卡适配：串口信号

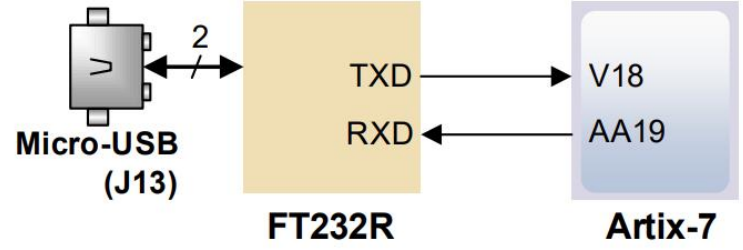
UART　SPI　I2C → USI模块

SoC共集成有三个USI模块，其中USI0模块默认被用作串口并完成了printf函数的串口重定向，可以通过使用printf函数直接通过串口将内容输出。

FT2232接口转换芯片主要功能

①为FPGA加载bitstream

②USB转换成UART与FPGA引脚相连。



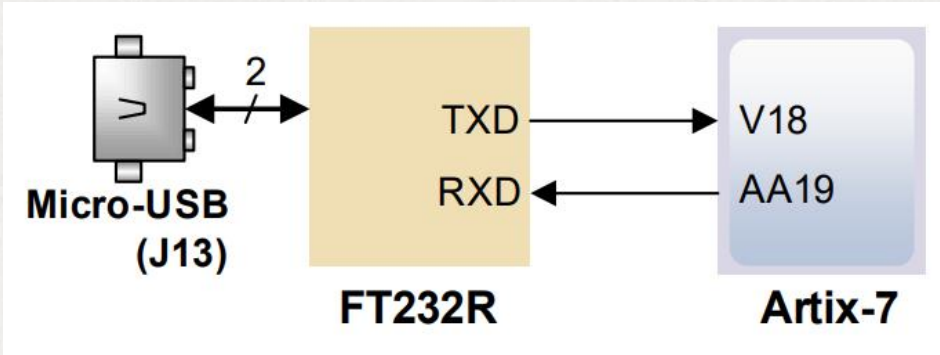
Serial Ports(PAD)	UART	I2C	SPI
SCLK	RXD	SCL	SCK
SD0	TXD	SDA	MOSI
SD1	CTS		MISO
NSS	RTS		NSS

```
## UART
set_property -dict { PACKAGE_PIN AA19  IOSTANDARD LVCMOS33 } [get_ports { PAD_USI0_SD0 }]; #IO_L15P_T2_DQS_RDWR_B_14 Sch=uart_rx_out
set_property -dict { PACKAGE_PIN V18   IOSTANDARD LVCMOS33 } [get_ports { PAD_USI0_SCLK }]; #IO_L14P_T2_SRCC_14 Sch=uart_tx_in
```

硬件设计

FPGA实现流程

③板卡适配：串口信号



Serial Ports(PAD)	UART	I2C	SPI
SCLK	RXD	SCL	SCK
SD0	TXD	SDA	MOSI
SD1	CTS		MISO
NSS	RTS		NSS

```
## UART
set_property -dict { PACKAGE_PIN AA19   IOSTANDARD LVCMOS33 } [get_ports { PAD_USI0_SD0 }]; #IO_L15P_T2_DQS_RDWR_B_14 Sch=uart_rx_out
set_property -dict { PACKAGE_PIN V18   IOSTANDARD LVCMOS33 } [get_ports { PAD_USI0_SCLK }]; #IO_L14P_T2_SRCC_14 Sch=uart_tx_in
```

硬件设计

串口信号的分配

这里将USI0的串口模块的RXD/TXD引脚分配至FPGA的V18/AA19后，即可在SDK直接使用printf相关函数，便于调试。

- 本课程SOC教学中所有实验内容都使用Nexys Video板卡完成
- 只要FPGA板卡的资源能够承载无剑SoC，都可以用来学习与实践本课程的内容。

I/O LAB实验

- 串口
- OLED
- LED
- SD卡读写
- 按键中断驱动

FPGA实现流程

③板卡适配：串口信号

在C语言标准库中，fputc函数是printf函数内部的一个函数，功能是将字符写入到文件指针所指向文件的当前写指针位置，简单理解就是把字符写入到特定文件中。

通过将串口发送函数重新修改 fputc函数内容，达到运行printf函数时从串口输出字符串。同样，通过将串口接收函数重新修改fgetc函数即可调用scanf、getchar等函数，通过串口获取字符。

```c
int fputc(int ch, FILE *stream)
{
    (void)stream;

    if (console_handle == NULL) {
        return -1;
    }

    if (ch == '\n') {
        csi_usart_putchar(console_handle, '\r');
    }

    csi_usart_putchar(console_handle, ch);

    return 0;
}
```

```c
int fgetc(FILE *stream)
{
    uint8_t ch;
    (void)stream;

    if (console_handle == NULL) {
        return -1;
    }

    csi_usart_getchar(console_handle, &ch);

    return ch;
}
```

重定向printf、scanf函数到串口

FPGA实现流程

③板卡适配：串口信号

wujian100 SDK中在进入main用户应用程序前会先将USI0模块初始化并配置为串口通信方式，且波特率为115200。SDK还完成了printf、scanf函数的重定向，因此用户可以直接在应用程序中调用printf、scanf相关函数，完成串口的输入输出，便于用户调试。

```c
int fputc(int ch, FILE *stream)
{
    (void)stream;

    if (console_handle == NULL) {
        return -1;
    }

    if (ch == '\n') {
        csi_usart_putchar(console_handle, '\r');
    }

    csi_usart_putchar(console_handle, ch);

    return 0;
}
```

```c
int fgetc(FILE *stream)
{
    uint8_t ch;
    (void)stream;

    if (console_handle == NULL) {
        return -1;
    }

    csi_usart_getchar(console_handle, &ch);

    return ch;
}
```

重定向printf、scanf函数到串口

FPGA实现流程

③板卡适配：复位信号、调试信号

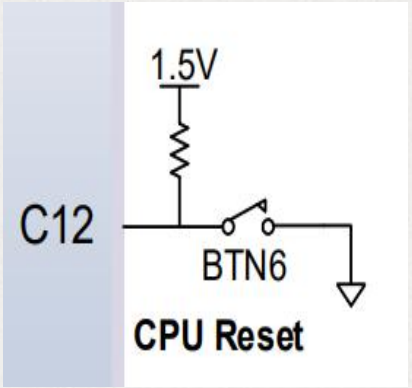
复位信号

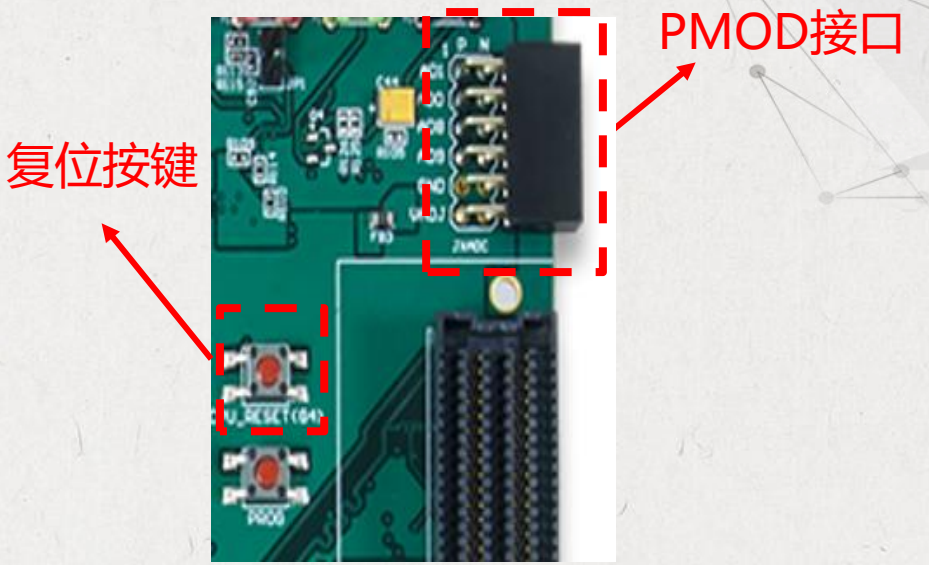
wujian100 SoC中复位信号为PAD_MCURST，低电平复位， Clocking Wizard IP的复位信号也设置为此信号提供，这里将其分配至按键上，当按键按下后处理器复位。

调试信号

wujian100 SoC采用双线JTAG接口，这里将其分配至PMOD接口上，便于与调试器连接。Nexys Video板卡上共有3个PMOD接口，可任意选择一个进行分配。

为了在后续I/O LAB实验中进行更多的应用，这里还需要把wujian100 SoC的相关信号分配至LED/switch开关和OLED等。



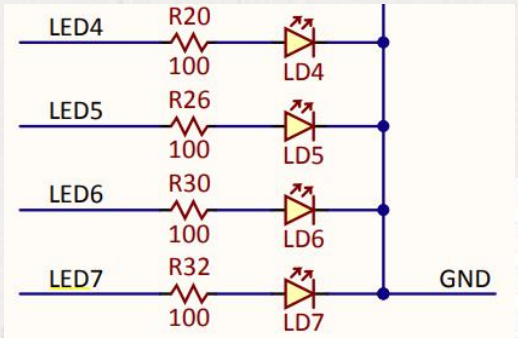


复位按键和PMOD接口

FPGA实现流程

③板卡适配：LED控制信号

将wujian100 SoC中GPIO控制相关信号PAD_GPIO_X分配至FPGA与LED灯相连的引脚，当软件控制GPIO输出高电平时LED灯点亮，输出低电平时LED熄灭。




```
set_property -dict { PACKAGE_PIN T15   IOSTANDARD LVCMOS25 } [get_ports { PAD_GPIO_0 }];
set_property -dict { PACKAGE_PIN T16   IOSTANDARD LVCMOS25 } [get_ports { PAD_GPIO_1 }];
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS25 } [get_ports { PAD_GPIO_2 }];
set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS25 } [get_ports { PAD_GPIO_3 }];
```

硬件设计

为实现LED灯每秒闪烁一次，在程序中使用提供的mdelay函数实现，该函数利用处理器内部的定时器实现较准确的ms级延迟。

```
while(1)
{
    LED_ON();
    mdelay(500);
    LED_OFF();
    mdelay(500);
}
```

软件实现

```
static void _mdelay(void)
{
    uint32_t load = csi_coret_get_load();
    uint32_t start = csi_coret_get_value();
    uint32_t cur;
    uint32_t cnt = (drv_get_sys_freq() / 1000);

    while (1) {
        cur = csi_coret_get_value();

        if (start > cur) {
            if (start - cur >= cnt) {
                return;
            }
        } else {
            if (load - cur + start > cnt) {
                return;
            }
        }
    }
}
```

```
void mdelay(uint32_t ms)
{
    if (ms == 0) {
        return;
    }

    while (ms--) {
        _mdelay();
    }
}
```

ms级延时的实现

FPGA实现流程

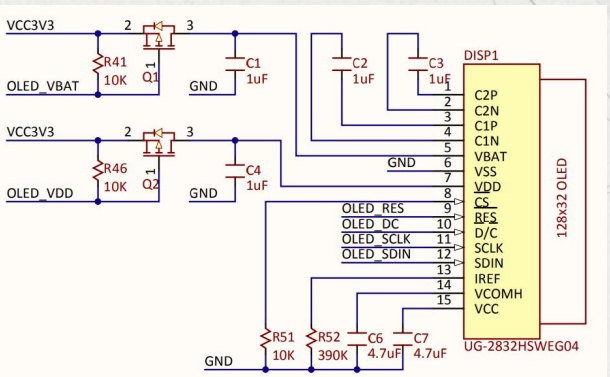
③板卡适配：OLED显示控制信号

OLED显示器与LED点阵彩色显示器的原理类似，但它采用的像素单元是“有机发光二极管”，所以像素密度比普通LED点阵显示器高得多。OLED显示器具有不需要背光源、对比度高、轻薄、视角广及响应速度快等优点。

Nexys Video板卡上搭载有128*32的OLED显示器和SSD1306显示驱动芯片，其面向FPGA的为四线SPI接口，通过SPI协议配置该芯片或传输数据从而完成显示器的控制。此处通过GPIO实现软件SPI协议控制显示驱动芯片。

Signal	Description	Polarity	FPGA pin
RES#	Reset	Active-low	U21
CS#	Chip select (always active)	Active-low	N/A
D/C#	Data (high)/Command (low)	Both	W22
SCLK	Serial Clock	Active-high	W21
SDIN	Serial Data	Active-high	Y22
VBAT#	Power enable for internal power supply	Active-low	P20
VDD#	Power enable for digital power	Active-low	V22

Table 112. OLED signal description.




```
## OLED Display
set_property -dict { PACKAGE_PIN W22   IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_21 }]
set_property -dict { PACKAGE_PIN U21   IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_22 }];
set_property -dict { PACKAGE_PIN W21   IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_23 }]
set_property -dict { PACKAGE_PIN Y22   IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_24 }];
set_property -dict { PACKAGE_PIN P20   IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_25 }];
set_property -dict { PACKAGE_PIN V22   IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_26 }];
```

FPGA实现流程

③板卡适配：OLED显示控制信号

OLED显示驱动的SPI控制协议和实现如下图。在CS片选信号低电平时开始进行数据传输，在SCLK时钟的上升沿完成数据的采样和传输，因此SDIN数据信号必须在SCLK时钟的上升沿保持稳定。当处理器向显示驱动传输的是命令时需要将D/C引脚拉低，当传输的是数据时需要将其拉高处理。

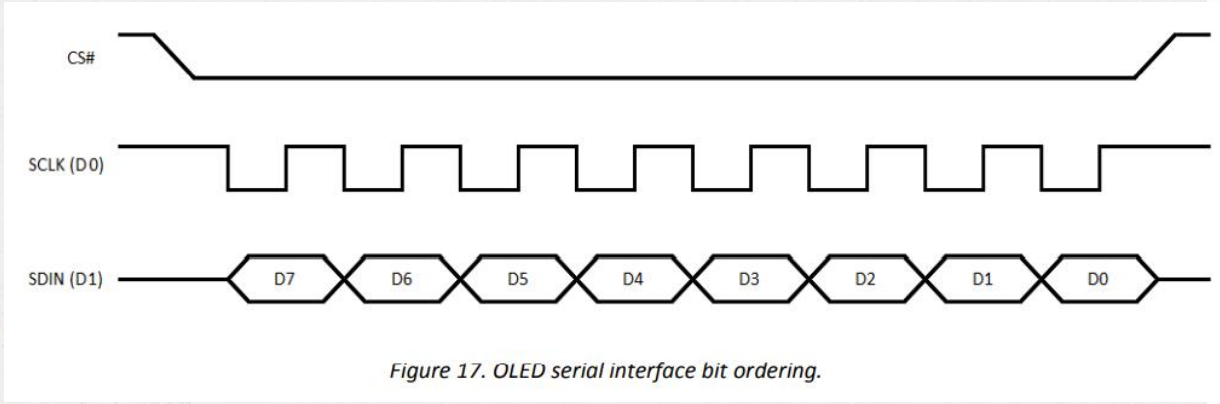


Figure 17. OLED serial interface bit ordering.

```
void OLED_WR_Byte(char dat,char cmd)
{
    char i;
    if(cmd)
      OLED_DC_Set();
    else
      OLED_DC_Clr();
    for(i=0;i<8;i++)
    {
        OLED_SCL_Clr();
        if(dat&0x80)
        {
         OLED_SDA_Set();
        }
        else
        {
         OLED_SDA_Clr();
        }
      OLED_SCL_Set();
        dat<<=1;
    }
    OLED_DC_Set();
```

SPI协议及实现

FPGA实现流程

③板卡适配：OLED显示控制信号

当实现底层的显示驱动控制逻辑后，即可实现处理器对显示驱动的配置和数据的传输。在上电后需要按照显示驱动芯片手册中的要求执行如下操作，即处理器需要完成对应的配置后才可以传输数据从而显示。这部分可以参考芯片厂商提供的驱动，无需自行实现，只需根据需求修改配置即可。

在完成上电配置后即可向显示驱动发送数据，从而让OLED显示字符等。

Start-up sequence:

1. Power up VDD by pulling OLED_VDD **low**. Wait 1ms.
2. Pulse RES# low for at least 3us.
3. Send initialization/configuration commands (see Table 12).
4. Power up VBAT by pulling OLED_VBAT **low**. Wait 100ms for voltage to stabilize.
5. Clear screen by writing zero to the display buffer.
6. Send "Display On" command (0xAF).

Command function	Command bytes
Charge pump enable	0x8D, 0x14
Set pre-charge period	0xD9, 0xF1
Contrast control	0x81, 0x0F
Column inversion disable	0xA0
Scan direction	0xC0
COM pins configuration	0xDA, 0x00
Addressing mode: horizontal	0x20

Table 12. OLED configuration commands.

上电配置和相关命令

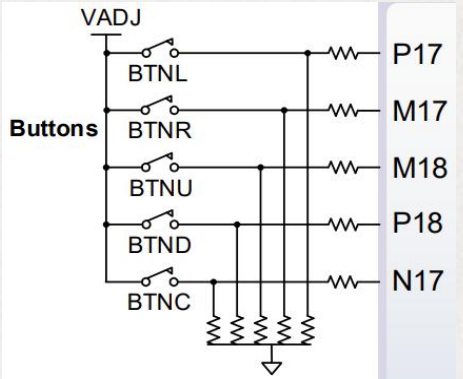
FPGA实现流程

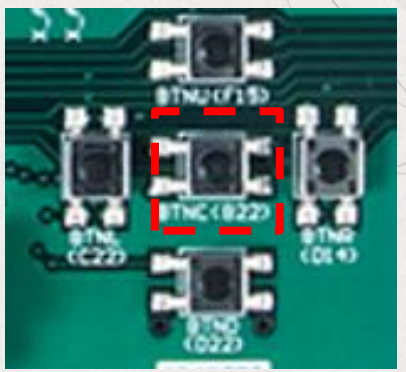
③板卡适配： GPIO按键中断

此处将PAD_GPIO_8分配至中间的按键，默认为低电平，当按键按下时该引脚为高电平。因此将该GPIO中断配置为上升沿触发，在中断处理程序中通过串口输出中断信息。

```
static void gpio_interrupt_handler(int32_t idx)
{
    printf("gpio interrupt is triggled\n");
}


void gpio_falling_edge_interrupt(pin_name_e gpio_pin)
{


    example_pin_gpio_init();

    printf("press the center button will triggle the gpio interrupt \n");
    pin = csi_gpio_pin_initialize(gpio_pin, gpio_interrupt_handler);

    csi_gpio_pin_config_mode(pin, GPIO_MODE_PULLNONE);
    csi_gpio_pin_config_direction(pin, GPIO_DIRECTION_INPUT);
    csi_gpio_pin_set_irq(pin, GPIO_IRQ_MODE_FALLING_EDGE, 1);
}
```

软件设置为输入模式、上升沿触发中断





```
## Buttons
set_property -dict { PACKAGE_PIN B22 IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_8 }]; #IO_L20N_T3_16 Sch=btnc
```

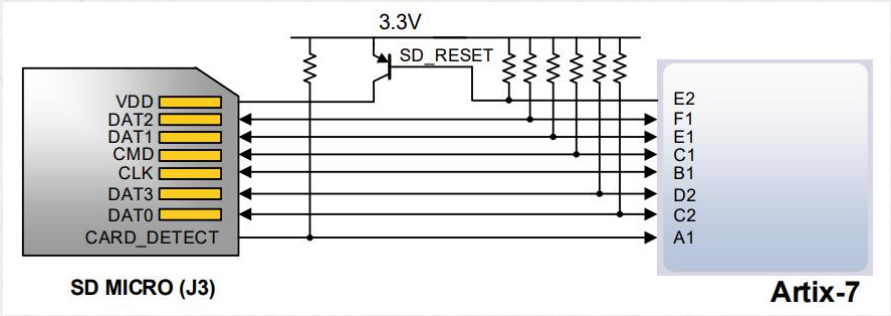
硬件设计

FPGA实现流程

③板卡适配： SD卡读写控制

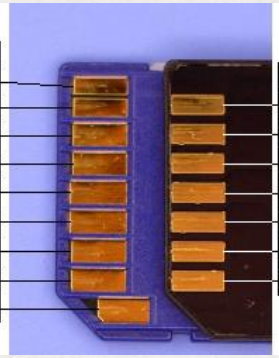
处理器对SD卡进行读写通信操作一般有两种通信接口可选，一种是SPI接口，另外一种是SDIO接口。此处采用GPIO软件模拟SPI协议读写SD卡。

下图是Nexys Video板卡上SD卡槽接口的原理图和SD卡的两种通信方式下的引脚定义。当SD_RESET引脚拉低时SD卡停止供电实现复位。CARD_DETECT可用于检测SD卡是否插入卡槽，默认为高电平，当SD卡插入卡槽时该引脚会在机械结构作用下与GND相连拉低。



SDC

No	SD	SPI
8	DAT1	
7	DAT0	DO
6	Vss2	
5	CLK	SCLK
4	Vcc	
3	Vss1	
2	CMD	DI
1	CAT3	CS
9	DAT2	



MMC

No	MMC	SPI
7	DAT	DO
6	Vss2	
5	CLK	SCLK
4	Vcc	
3	Vss1	
2	CMD	DI
1	RES	CS

```
## SD card
set_property -dict { PACKAGE_PIN W19   IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_6 }]; #IO_L12P_T1_MRCC_14 Sch=sd_cclk        sclk       PAD_GPIO_6
set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_18 }]; #IO_L20N_T3_A07_D23_14 Sch=sd_cd       card detect
set_property -dict { PACKAGE_PIN W20   IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_5 }]; #IO_L12N_T1_MRCC_14 Sch=sd_cmd         mosi       PAD_GPIO_5
set_property -dict { PACKAGE_PIN V19   IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_7 }]; #IO_L14N_T2_SRCC_14 Sch=sd_d[0]        miso       PAD_GPIO_7
#set_property -dict { PACKAGE_PIN T21   IOSTANDARD LVCMOS33 } [get_ports { sd_d[1] }]; #IO_L4P_T0_D04_14 Sch=sd_d[1]
#set_property -dict { PACKAGE_PIN T20   IOSTANDARD LVCMOS33 } [get_ports { sd_d[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sd_d[2]
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_19 }]; #IO_L18N_T2_A11_D27_14 Sch=sd_d[3]       cs         PAD_GPIO_19
set_property -dict { PACKAGE_PIN V20   IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_20 }]; #IO_L11N_T1_SRCC_14 Sch=sd_reset       rst        PAD_GPIO_20
```

硬件设计

FPGA实现流程

③板卡适配： SD卡读写控制

在实际应用中直接操作SD卡存储单元是不现实的。SD卡一般用来存放文件，所以都需要加载文件系统到里面，此处移植FatFs文件系统进行对SD卡的控制。

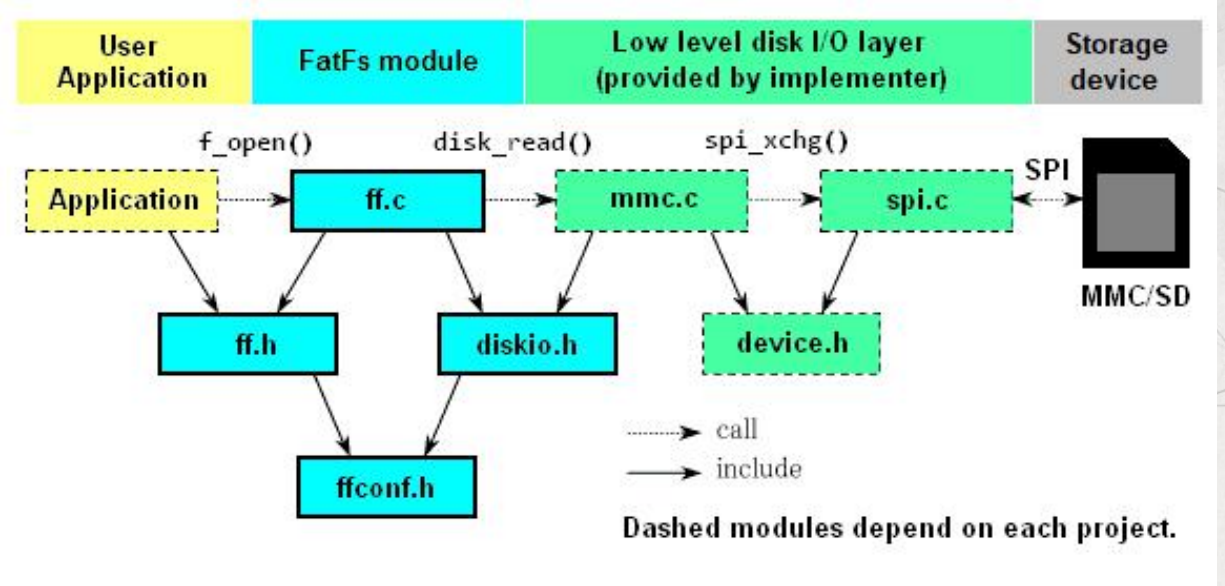
f_mount()

f_open()

f_write()

f_read()

这些应用函数的实现均在FatFs模块中，这部分只需在ffconf.h文件中完成配置以满足不同应用即可。需要用户实现的就是绿色部分的底层存储设备的读写操作函数、存储设备信息获取函数等。



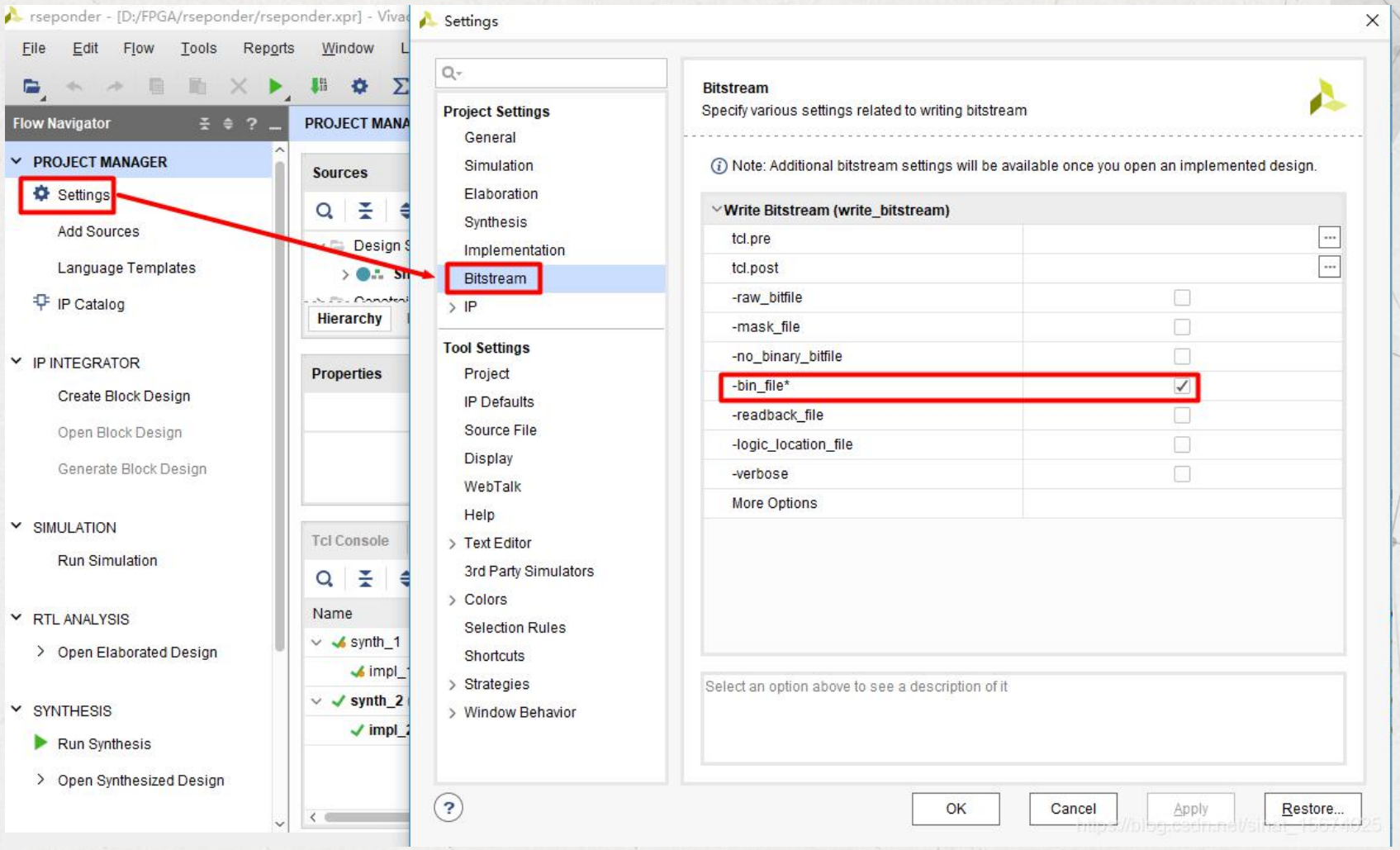
Fatfs文件系统结构图

FPGA实现流程

④FLASH固化

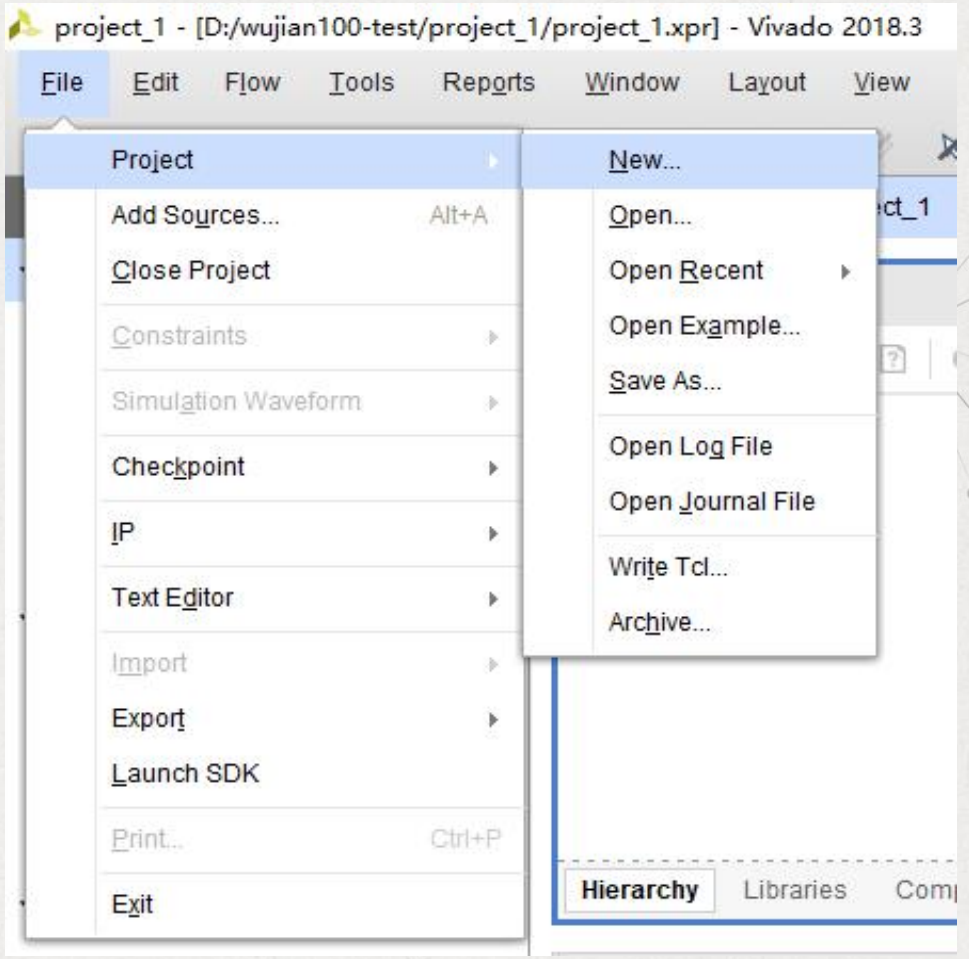
Nexys Video板卡上搭载的FLASH型号为S25FL256S，通过上图的勾选生成bin文件选项后即可在综合实现后生成。由于FLASH读写速度相对较慢，因此将bin文件写入FLASH需要较长时间。

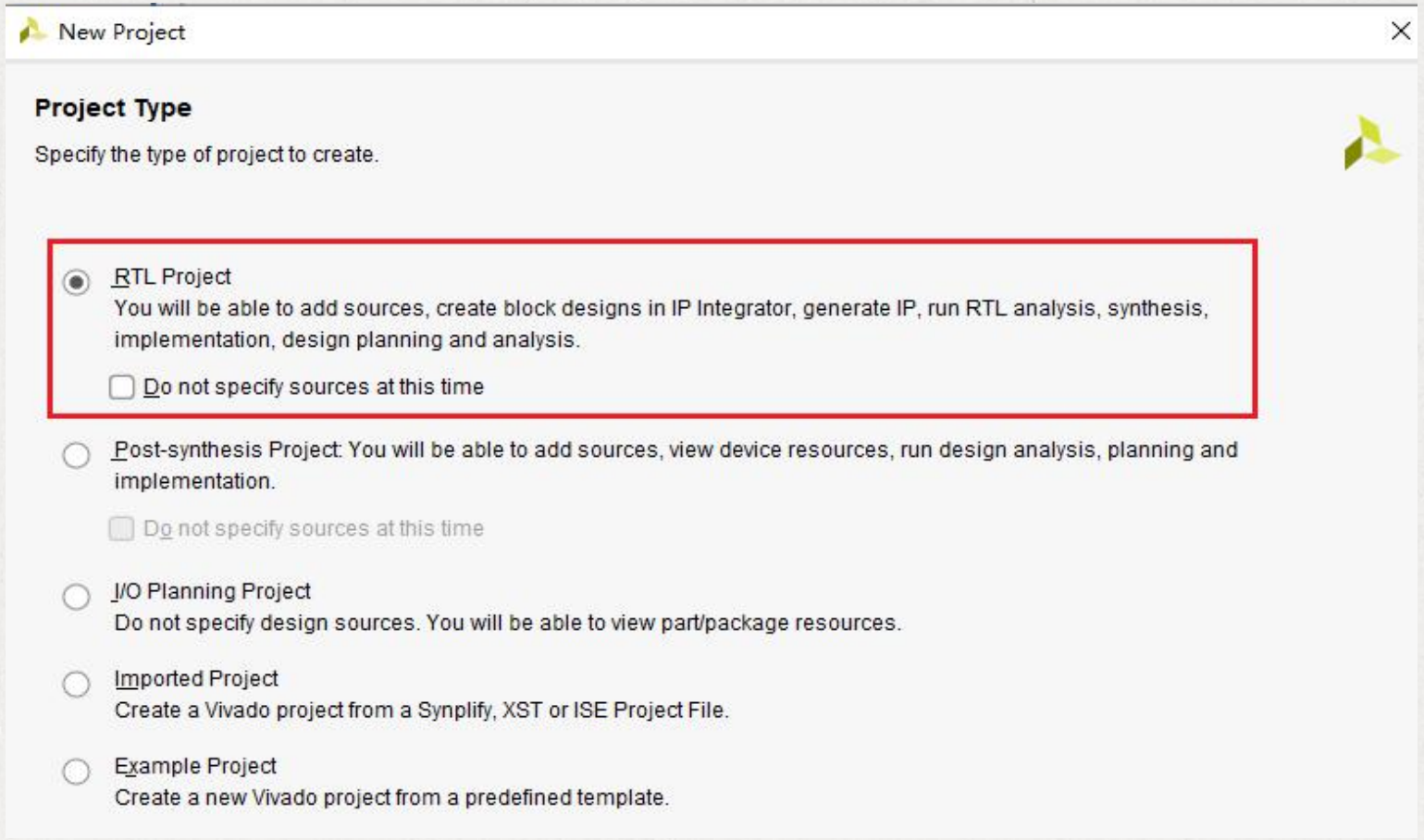
完成以上步骤后即可综合实现生成bitstream文件并加载入FPGA，但在FPGA掉电后需要重新加载。因此可以通过生成bin文件并加载入FLASH存储，在FPGA上电后自动从FLASH中读取并完成配置。



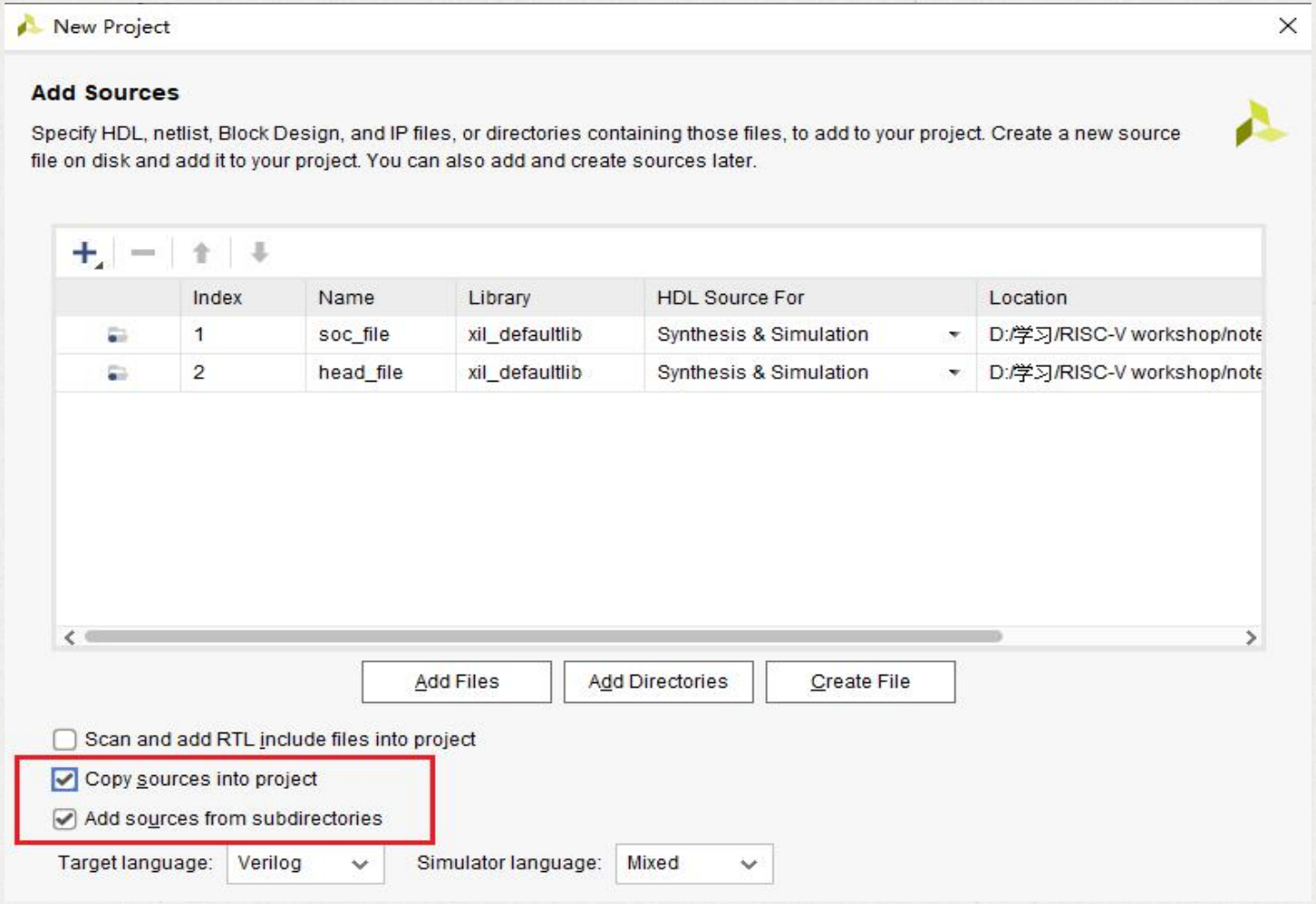
第一步 创建工程

在Vivado中创建一个新的工程

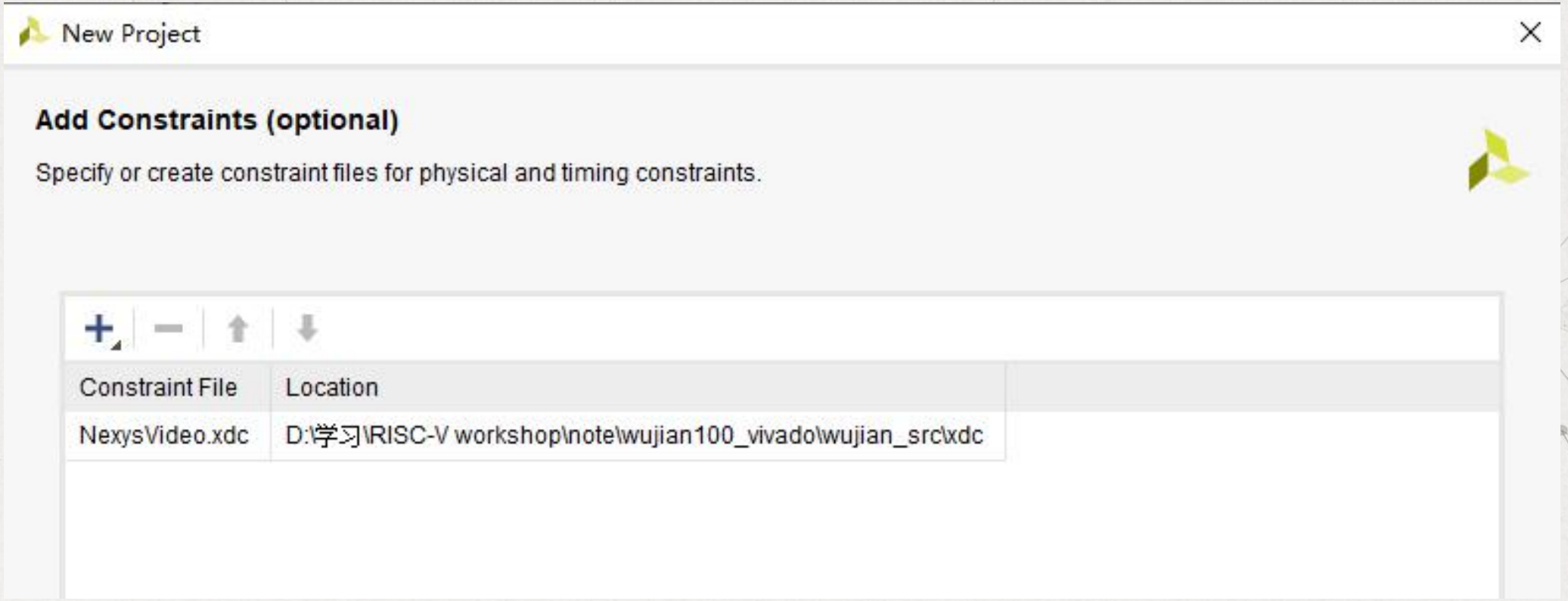




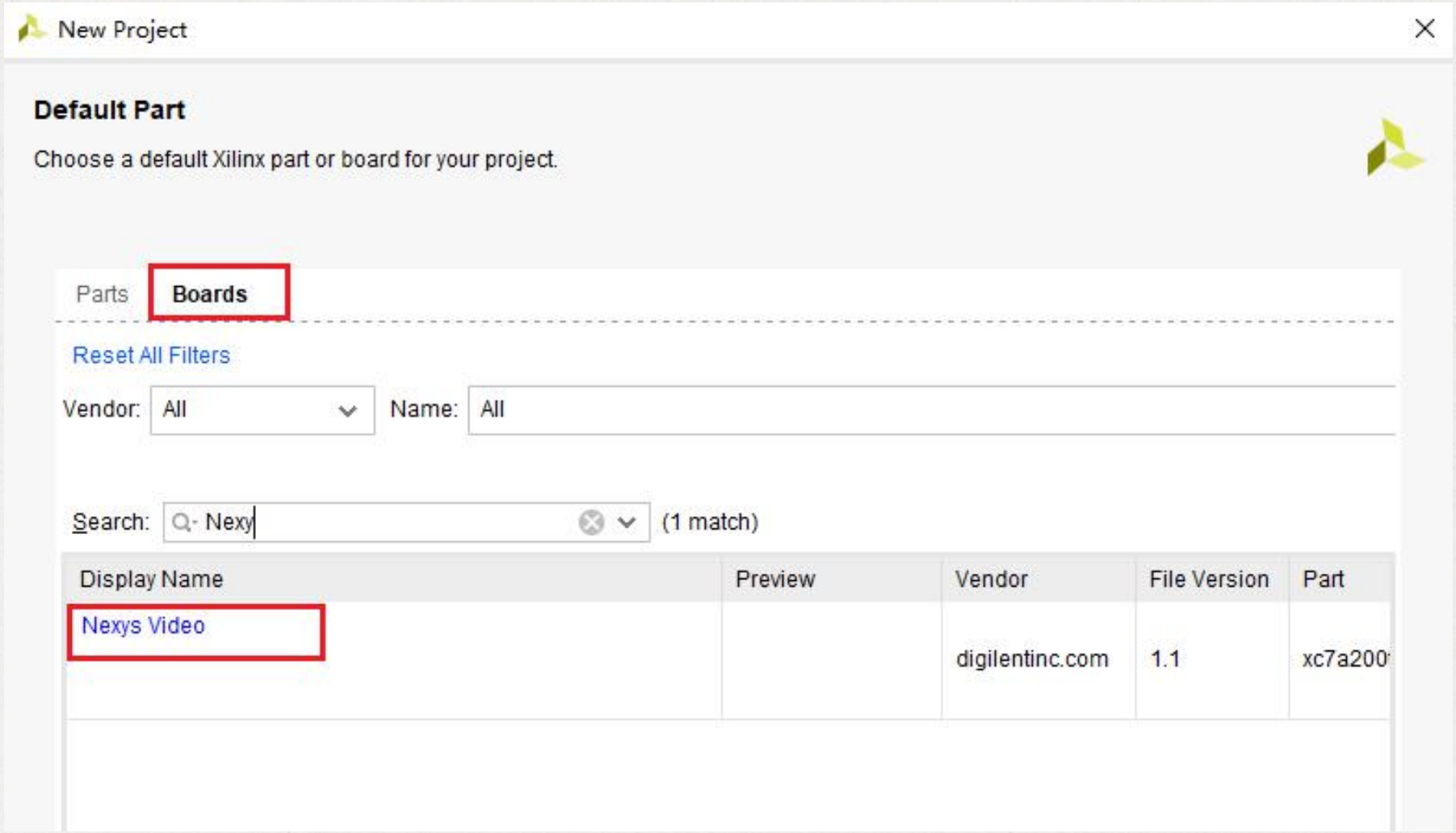
（1）注意要选择RTL Project



(2) 添加设计源文件和头文件


New Project
Add Constraints (optional)
Specify or create constraint files for physical and timing constraints.
Constraint File
Location
NexysVideo.xdc
D:\学习\RISC-V workshop\note\wujian100_vivado\wujian_src\xdc

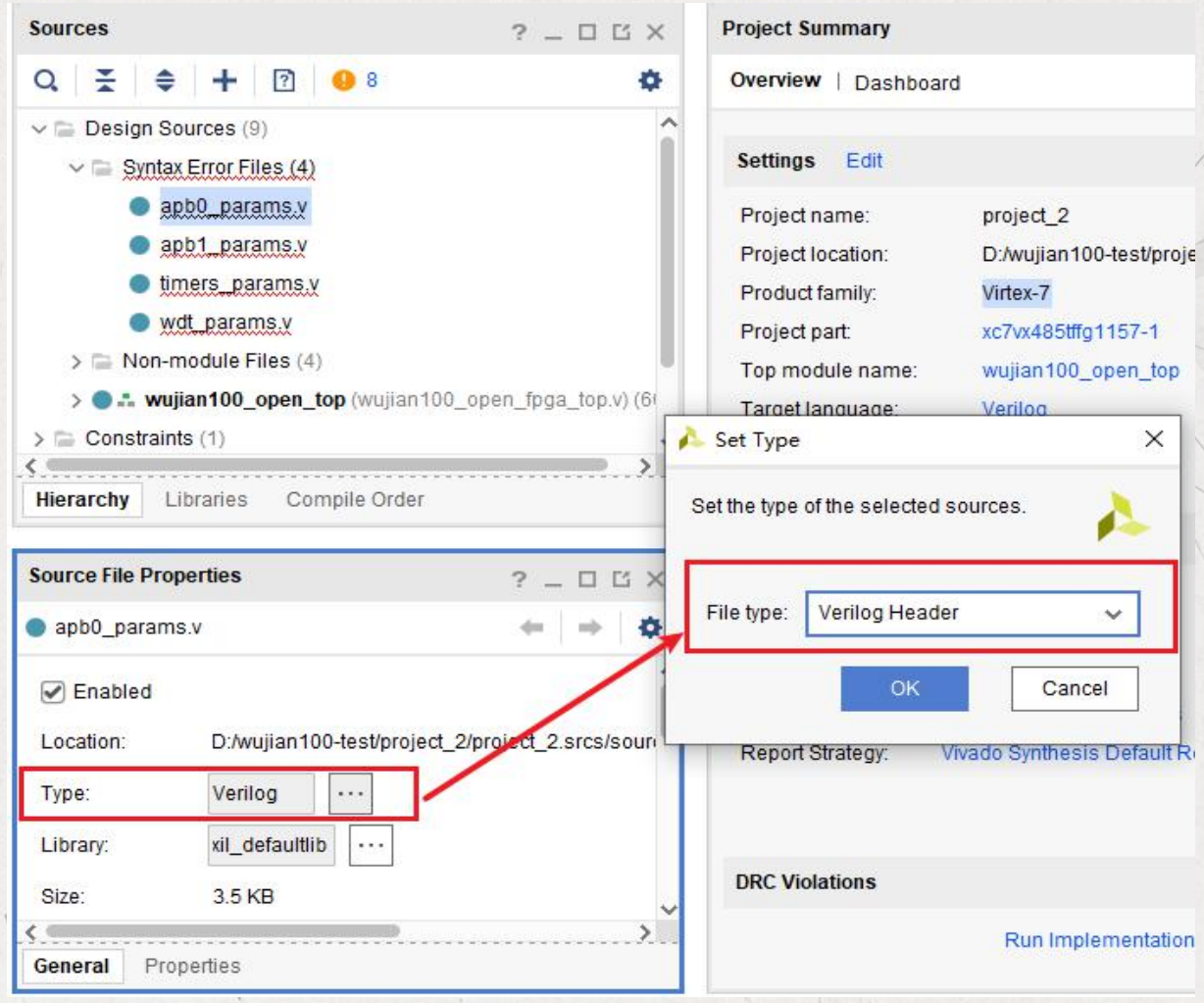
(3) 添加约束文件

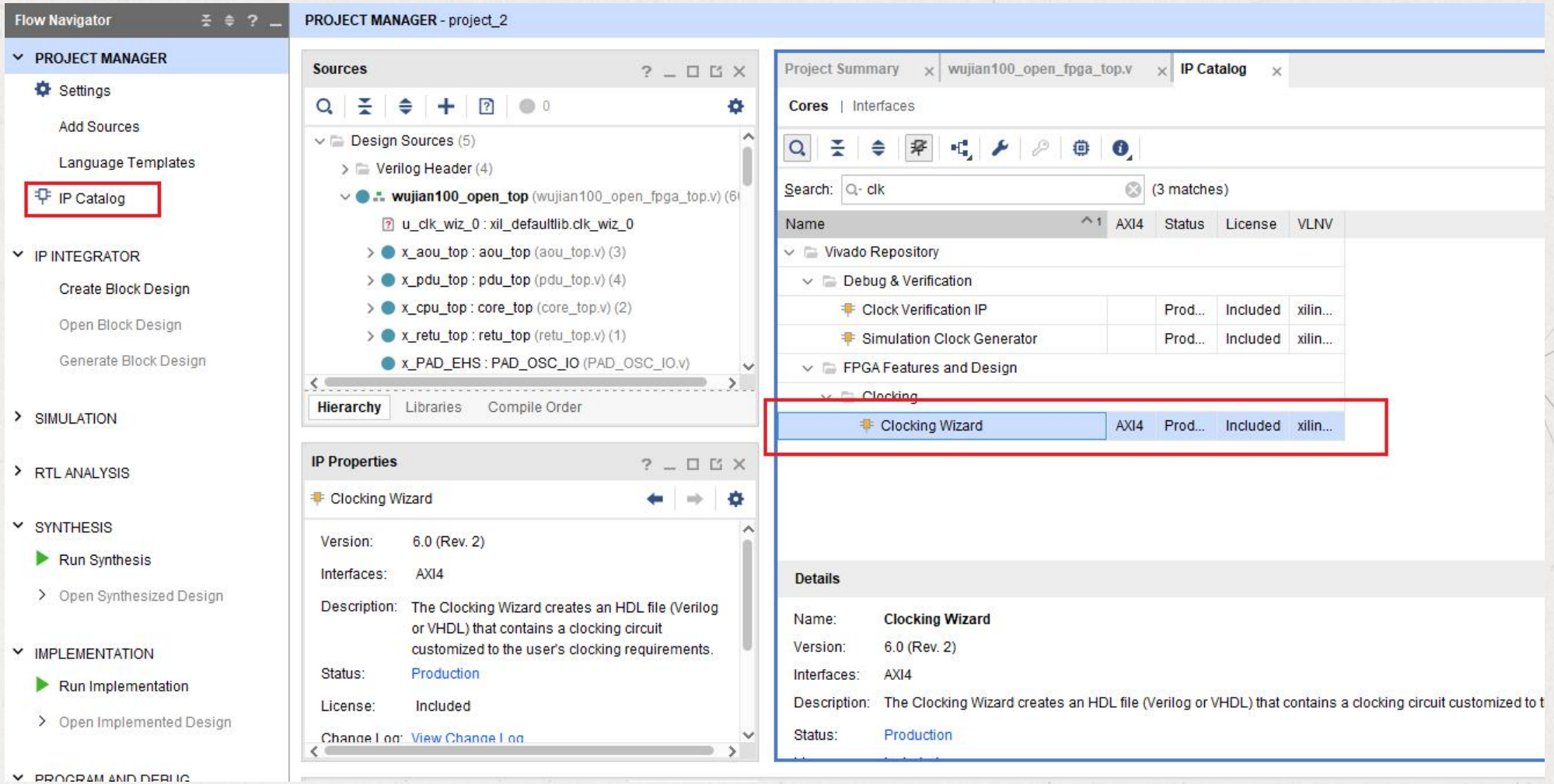

New Project
Default Part
Choose a default Xilinx part or board for your project.
Parts
Boards
Reset All Filters
Vendor: All
Name: All
Search: Nexy
(1 match)
Display Name
Preview
Vendor
File Version
Part
Nexys Video
digilentinc.com
1.1
xc7a200

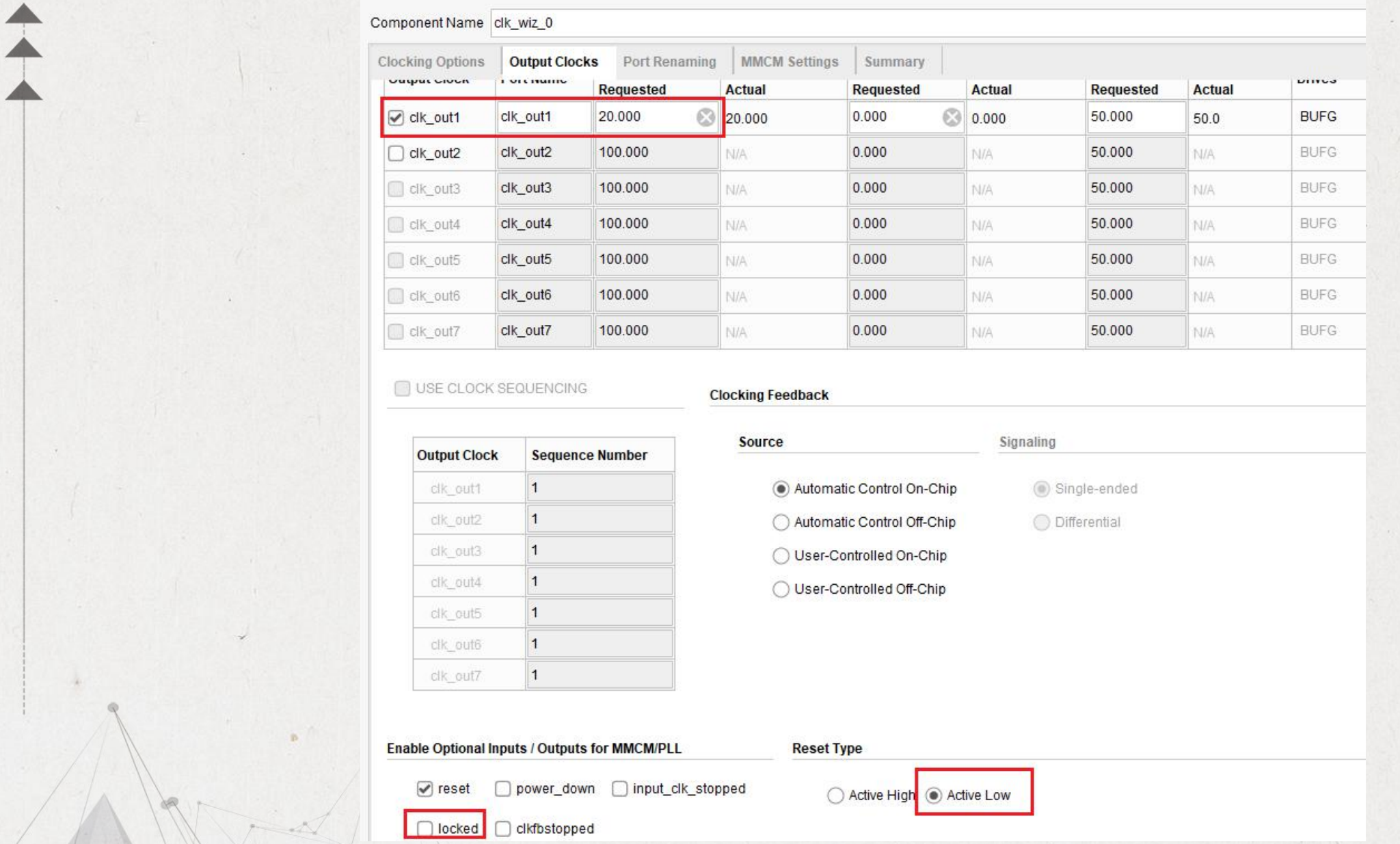
（4）选择正确的板子

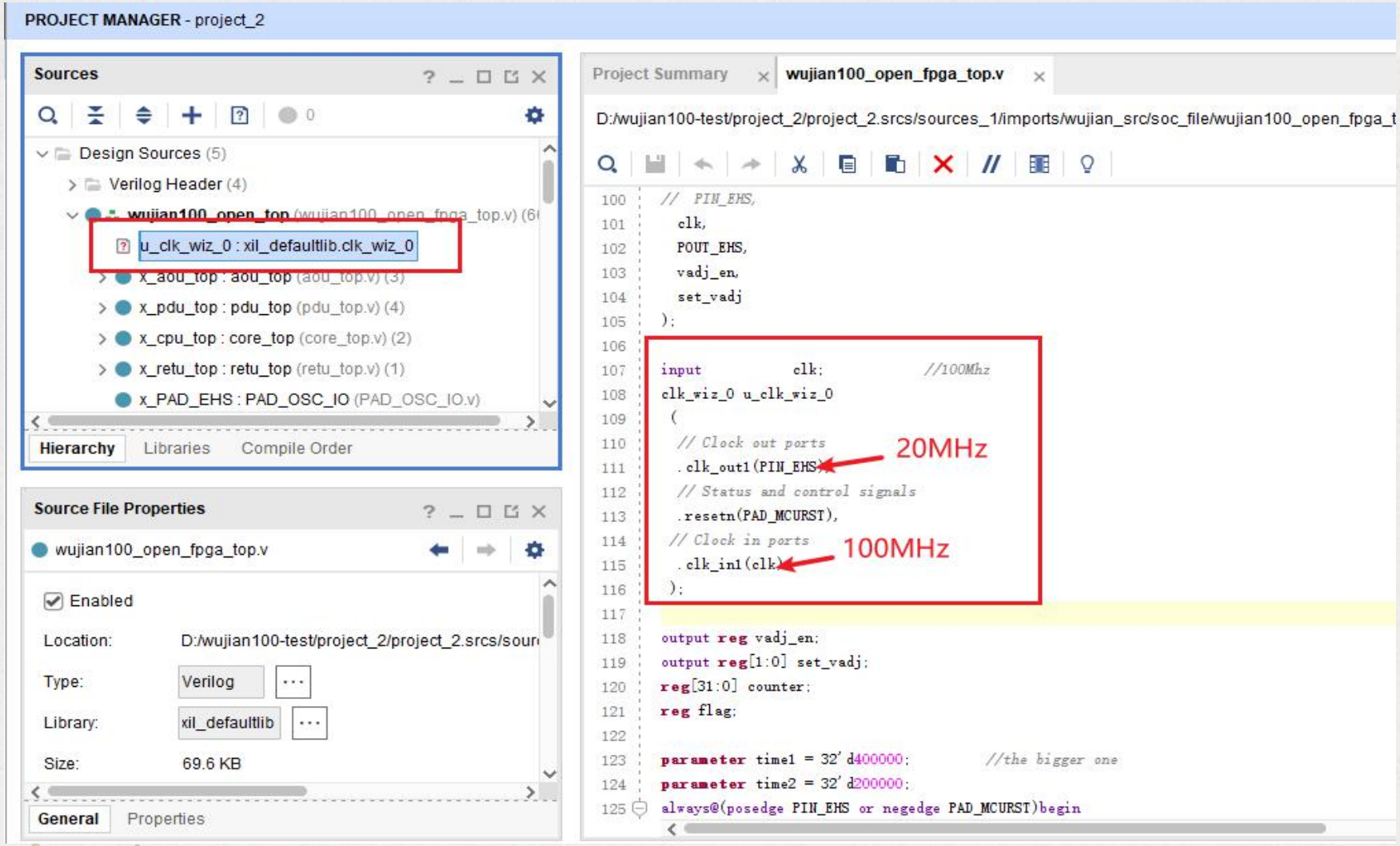
第二步 修正错误、加载IP

(1) 有错误，需修改类型




Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
RTL ANALYSIS
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
PROJECT MANAGER - project_2
Sources
Design Sources (5)
Verilog Header (4)
wujian100_open_top (wujian100_open_fpga_top.v) (6
u_clk_wiz_0 : xil_defaultlib.clk_wiz_0
x_aou_top : aou_top (aou_top.v) (3)
x_pdu_top : pdu_top (pdu_top.v) (4)
x_cpu_top : core_top (core_top.v) (2)
x_retu_top : retu_top (retu_top.v) (1)
x_PAD_EHS : PAD_OSC_IO (PAD_OSC_IO.v)
Hierarchy
Libraries
Compile Order
IP Properties
Clocking Wizard
Version: 6.0 (Rev. 2)
Interfaces: AXI4
Description: The Clocking Wizard creates an HDL file (Verilog or VHDL) that contains a clocking circuit customized to the user's clocking requirements.
Status: Production
License: Included
Change Log: View Change Log
Project Summary
wujian100_open_fpga_top.v
IP Catalog
Cores | Interfaces
Search: clk
(3 matches)
Name
AXI4
Status
License
VLNV
Vivado Repository
Debug & Verification
Clock Verification IP
Prod...
Included
xilin...
Simulation Clock Generator
Prod...
Included
xilin...
FPGA Features and Design
Clocking
Clocking Wizard
AXI4
Prod...
Included
xilin...
Details
Name: Clocking Wizard
Version: 6.0 (Rev. 2)
Interfaces: AXI4
Description: The Clocking Wizard creates an HDL file (Verilog or VHDL) that contains a clocking circuit customized to t
Status: Production


Component Name clk_wiz_0
Clocking Options
Output Clocks
Port Renaming
MMCM Settings
Summary
Requested
Actual
Requested
Actual
Requested
Actual
clk_out1 clk_out1 20.000 20.000 0.000 0.000 50.000 50.0 BUFG
clk_out2 clk_out2 100.000 N/A 0.000 N/A 50.000 N/A BUFG
clk_out3 clk_out3 100.000 N/A 0.000 N/A 50.000 N/A BUFG
clk_out4 clk_out4 100.000 N/A 0.000 N/A 50.000 N/A BUFG
clk_out5 clk_out5 100.000 N/A 0.000 N/A 50.000 N/A BUFG
clk_out6 clk_out6 100.000 N/A 0.000 N/A 50.000 N/A BUFG
clk_out7 clk_out7 100.000 N/A 0.000 N/A 50.000 N/A BUFG
USE CLOCK SEQUENCING
Clocking Feedback
Output Clock
Sequence Number
clk_out1 1
clk_out2 1
clk_out3 1
clk_out4 1
clk_out5 1
clk_out6 1
clk_out7 1
Source
Automatic Control On-Chip
Automatic Control Off-Chip
User-Controlled On-Chip
User-Controlled Off-Chip
Signaling
Single-ended
Differential
Enable Optional Inputs / Outputs for MMCM/PLL
reset
power_down
input_clk_stopped
locked
clkfbstopped
Reset Type
Active High
Active Low


PROJECT MANAGER - project_2
Sources
Design Sources (5)
Verilog Header (4)
wujian100_open_top (wujian100_open_fpga_top.v) (6
u_clk_wiz_0 : xil_defaultlib.clk_wiz_0
x_aou_top : aou_top (aou_top.v) (3)
x_pdu_top : pdu_top (pdu_top.v) (4)
x_cpu_top : core_top (core_top.v) (2)
x_retu_top : retu_top (retu_top.v) (1)
x_PAD_EHS : PAD_OSC_IO (PAD_OSC_IO.v)
Hierarchy
Libraries
Compile Order
Source File Properties
wujian100_open_fpga_top.v
Enabled
Location:
D:/wujian100-test/project_2/project_2.srcs/sour
Type:
Verilog
Library:
xil_defaultlib
Size:
69.6 KB
General
Properties
Project Summary
wujian100_open_fpga_top.v
D:/wujian100-test/project_2/project_2.srcs/sources_1/imports/wujian_src/soc_file/wujian100_open_fpga_t
100 // PIN_EHS,
101 clk,
102 POUT_EHS,
103 vadj_en,
104 set_vadj
105);
106
107 input clk; //100Mhz
108 clk_wiz_0 u_clk_wiz_0
109 (
110 // Clock out ports
111 .clk_out1(PIN_EHS),
112 // Status and control signals
113 .resetn(PAD_MCURST),
114 // Clock in ports
115 .clk_in1(clk)
116);
117
118 output reg vadj_en;
119 output reg[1:0] set_vadj;
120 reg[31:0] counter;
121 reg flag;
122
123 parameter time1 = 32'd400000; //the bigger one
124 parameter time2 = 32'd200000;
125 always@(posedge PIN_EHS or negedge PAD_MCURST)begin
20MHz
100MHz

(2) 加时钟分频IP

Project Summary | wujian100_open_fpga_top.v | IP Catalog | NexysVideo.xdc

D:/学习/RISC-V workshop/note/wujian100_vivado/wujian_src/xdc/NexysVideo.xdc

```
1  ### This file is a general .xdc for the Nexys Video Rev. A
2  ### To use it in a project:
3  ### - uncomment the lines corresponding to used pins
4  ### - rename the used ports (in each line, after get_ports) according to the top level signal names in the project
5
6
7  ## Clock Signal
8  set_property -dict { PACKAGE_PIN R4    IOSTANDARD LVCMOS33 } [get_ports { clk }]; #IO_L13P_T2_MRCC_34 Sch=sysclk
9  create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]
10
11 set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets PAD_JTAG_TCLK]
12
13 ## FMC Transceiver clocks (Must be set to value provided by Mezzanine card, currently set to 156.25 MHz)
14 ## Note: This clock is attached to a MGTREFCLK pin
15 #set_property -dict { PACKAGE_PIN E6 } [get_ports { GTP_CLK_N }];
16 #set_property -dict { PACKAGE_PIN F6 } [get_ports { GTP_CLK_P }];
17 #create_clock -add -name gtpclk0_pin -period 6.400 -waveform {0 3.200} [get_ports {GTP_CLK_P}];
18 #set_property -dict { PACKAGE_PIN E10 } [get_ports { FMC_MGT_CLK_N }];
19 #set_property -dict { PACKAGE_PIN F10 } [get_ports { FMC_MGT_CLK_P }];
20 #create_clock -add -name mgtclk1_pin -period 6.400 -waveform {0 3.200} [get_ports {FMC_MGT_CLK_P}];
```

(3) 确认约束内容

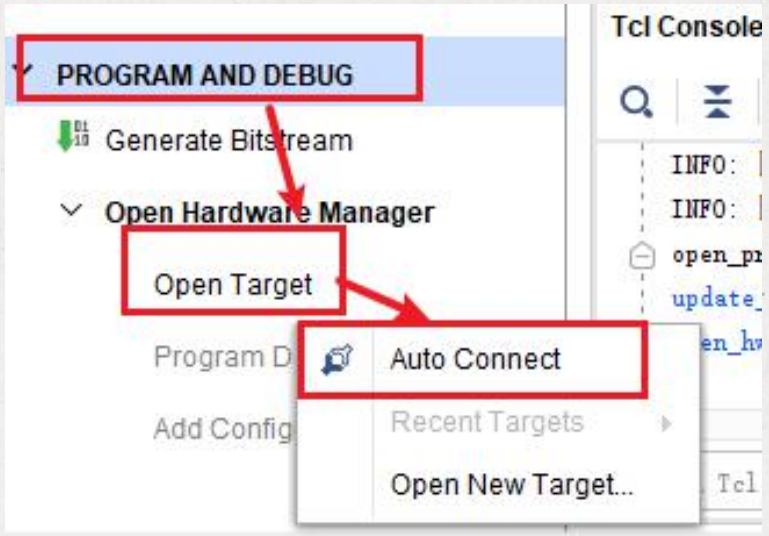
板上端口

设计模块中的端口

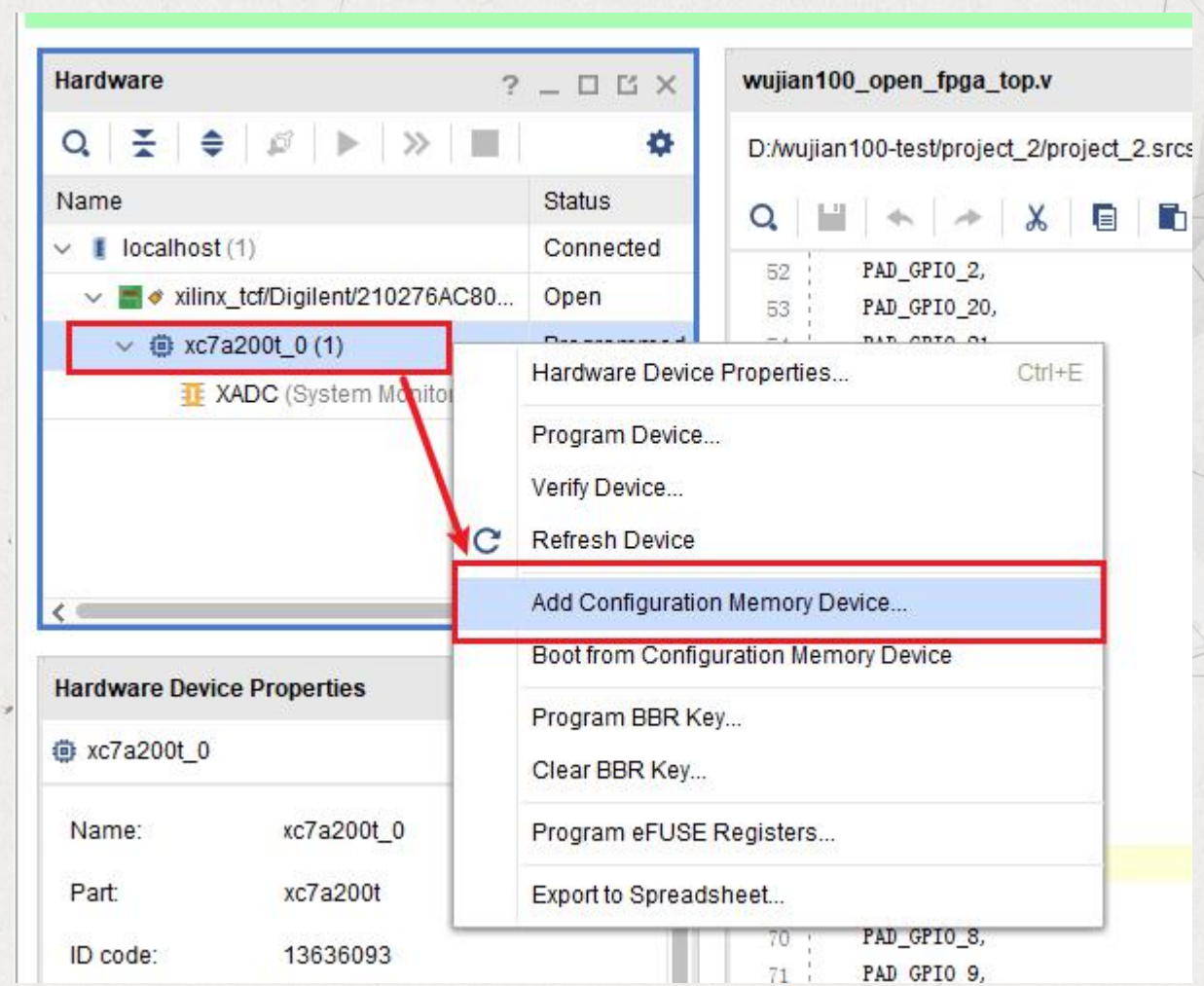
```
## UART
set_property -dict { PACKAGE_PIN AA19  IOSTANDARD LVCMOS33 } [get_ports { PAD_USIO_SDO }]; #IO_L15P_T2_DQS_RDWR_B_14 Sch=uart_rx_out
set_property -dict { PACKAGE_PIN V18   IOSTANDARD LVCMOS33 } [get_ports { PAD_USIO_SCLK }]; #IO_L14P_T2_SRCC_14 Sch=uart_tx_in
```

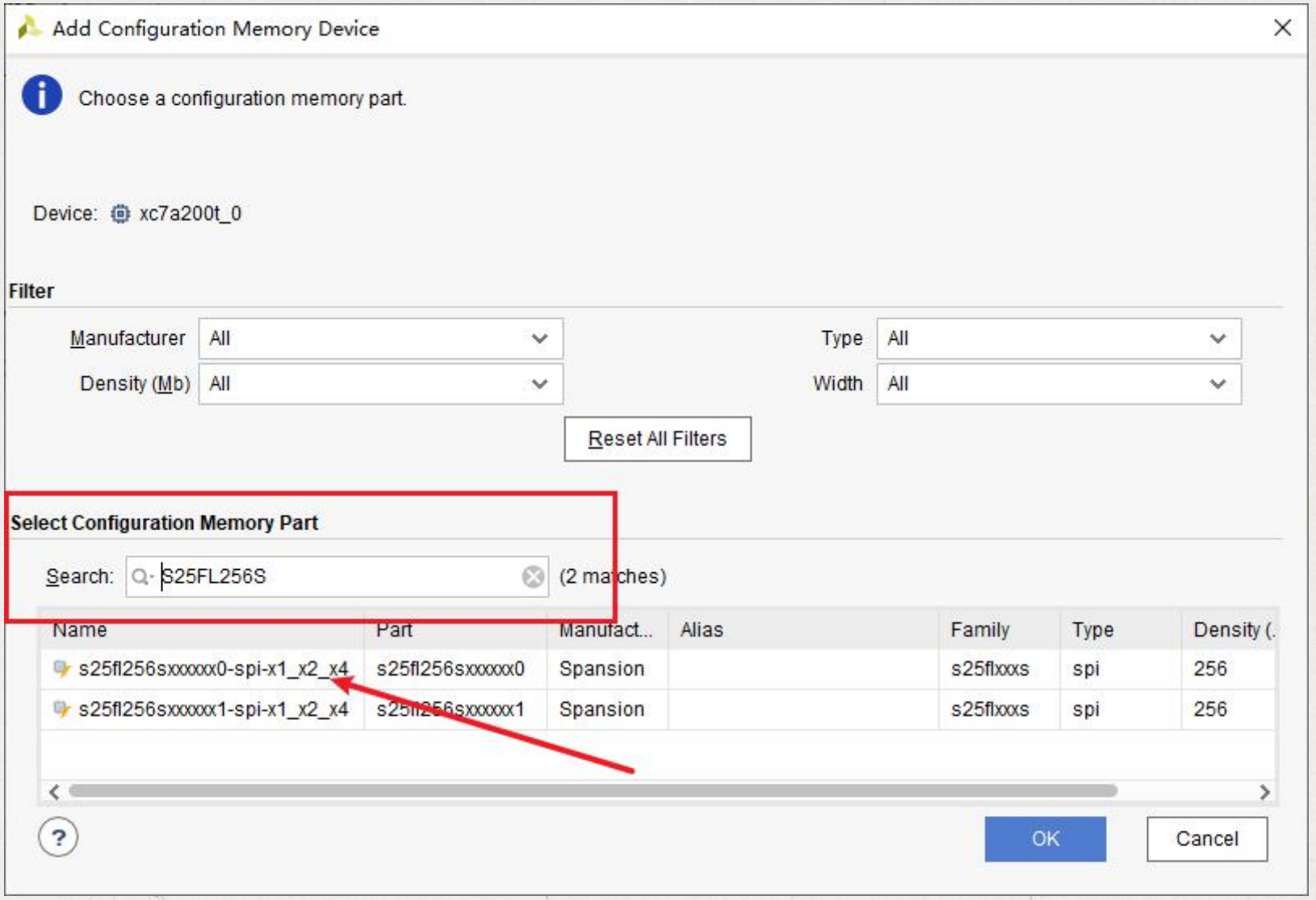
```
## Buttons
set_property -dict { PACKAGE_PIN B22 IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_8 }]; #IO_L20N_T3_16 Sch=btnc
set_property -dict { PACKAGE_PIN D22 IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_9 }]; #IO_L22N_T3_16 Sch=btnd
set_property -dict { PACKAGE_PIN C22 IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_10 }]; #IO_L20P_T3_16 Sch=btnl
set_property -dict { PACKAGE_PIN D14 IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_11 }]; #IO_L6P_T0_16 Sch=btnr
set_property -dict { PACKAGE_PIN F15 IOSTANDARD LVCMOS33 } [get_ports { PAD_GPIO_12 }]; #IO_0_16 Sch=btnu
set_property -dict { PACKAGE_PIN G4  IOSTANDARD LVCMOS15 } [get_ports { PAD_MCURST }]; #IO_L12N_T1_MRCC_35 Sch=cpu_resetn
```

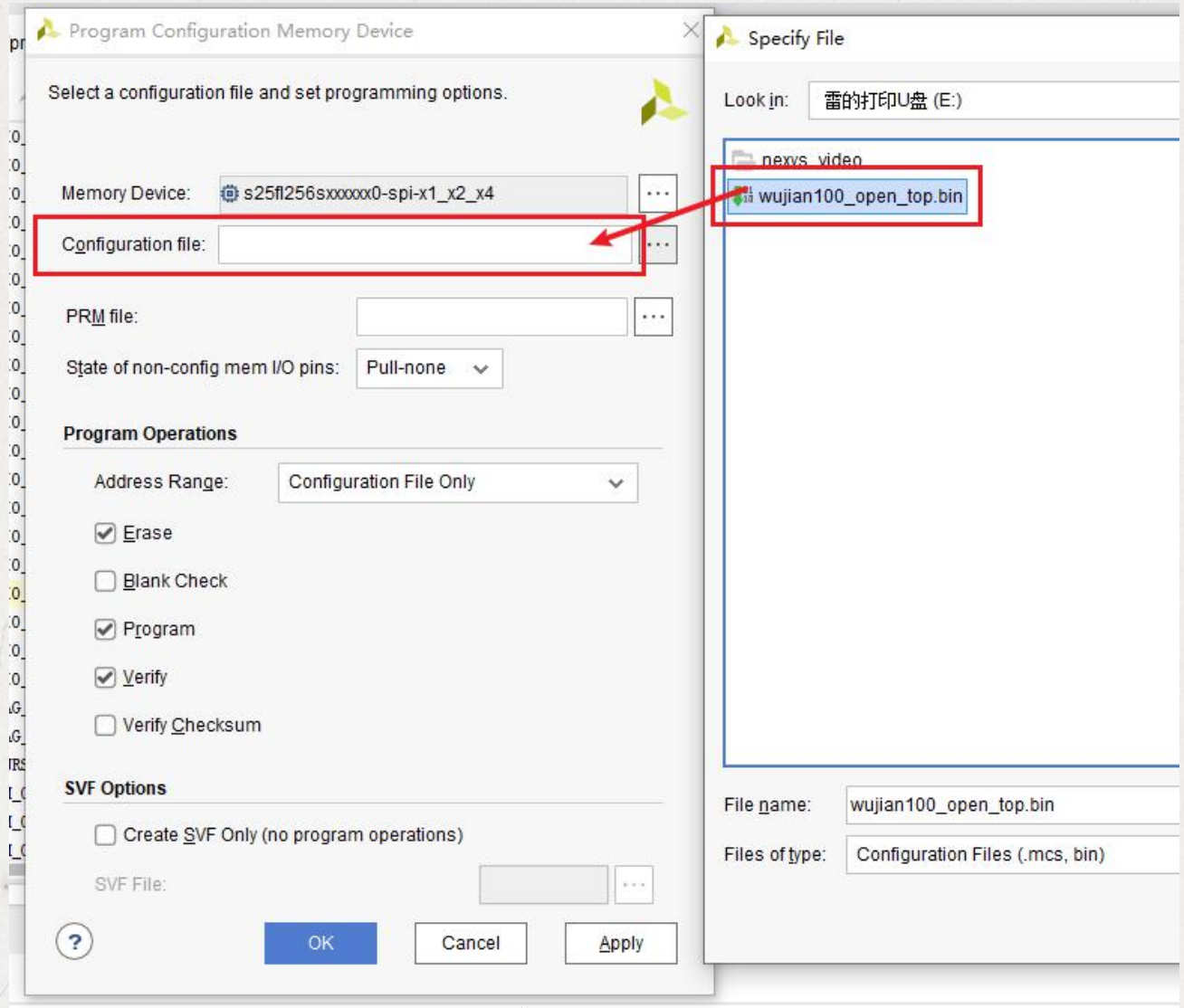
FLASH固化流程

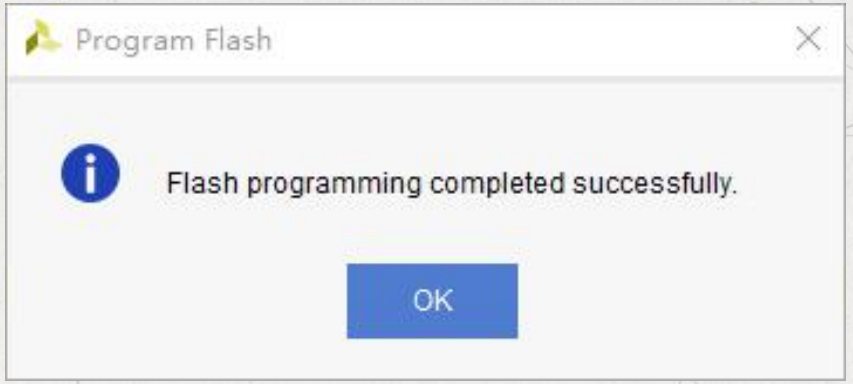


打开一个最近打开的目标



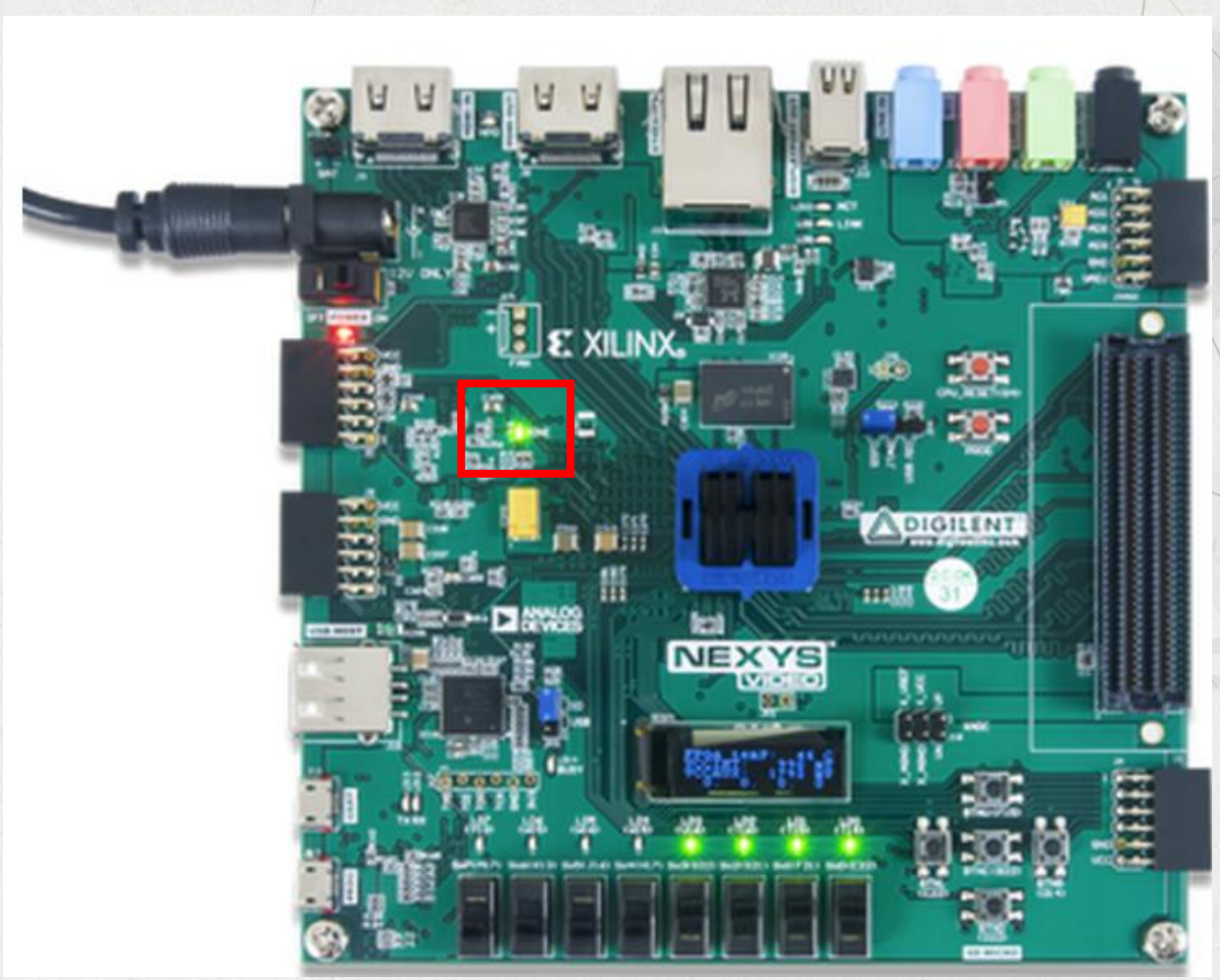

Add Configuration Memory Device
Choose a configuration memory part.
Device: xc7a200t_0
Filter
Manufacturer All
Type All
Density (Mb) All
Width All
Reset All Filters
Select Configuration Memory Part
Search: S25FL256S
(2 matches)
Name
Part
Manufact...
Alias
Family
Type
Density (.
s25fl256sxxxxxx0-spi-x1_x2_x4
s25fl256sxxxxxx0
Spansion
s25flxxxs
spi
256
s25fl256sxxxxxx1-spi-x1_x2_x4
s25fl256sxxxxxx1
Spansion
s25flxxxs
spi
256
OK
Cancel





加载到FLASH的过程会很慢……

烧写成功后的现象



先按下PROG按键，等待DONE亮起。

I/O LAB实验

I/O LAB实验主要完成操作：

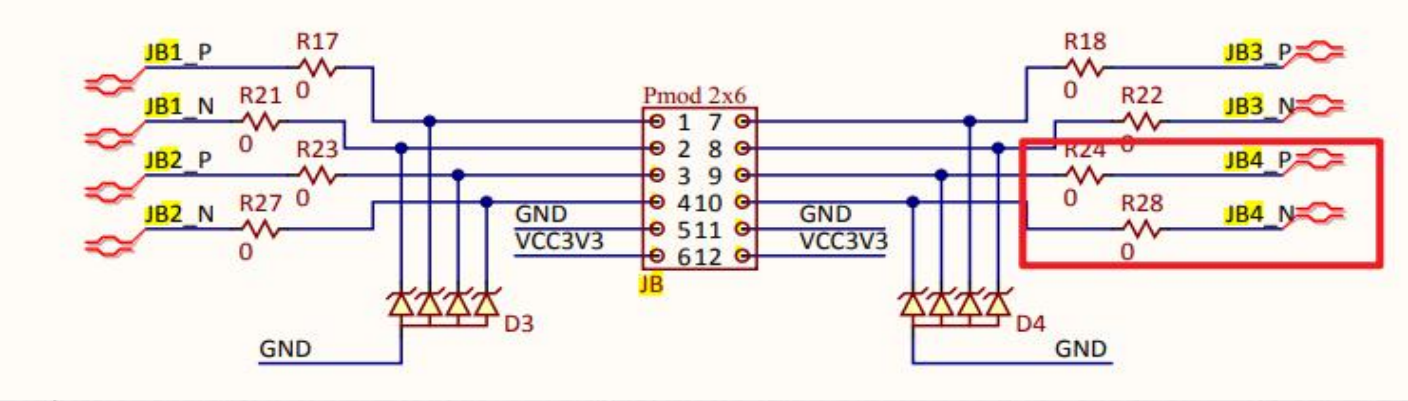
- 串口输出调试信息
- LED1-LED3每秒闪烁一次
- OLED显示字符串
- 按下center button之后，会触发GPIO中断从而输出信息。
- MCU在SD卡中新建文件并写入内容

第一步 连接板子

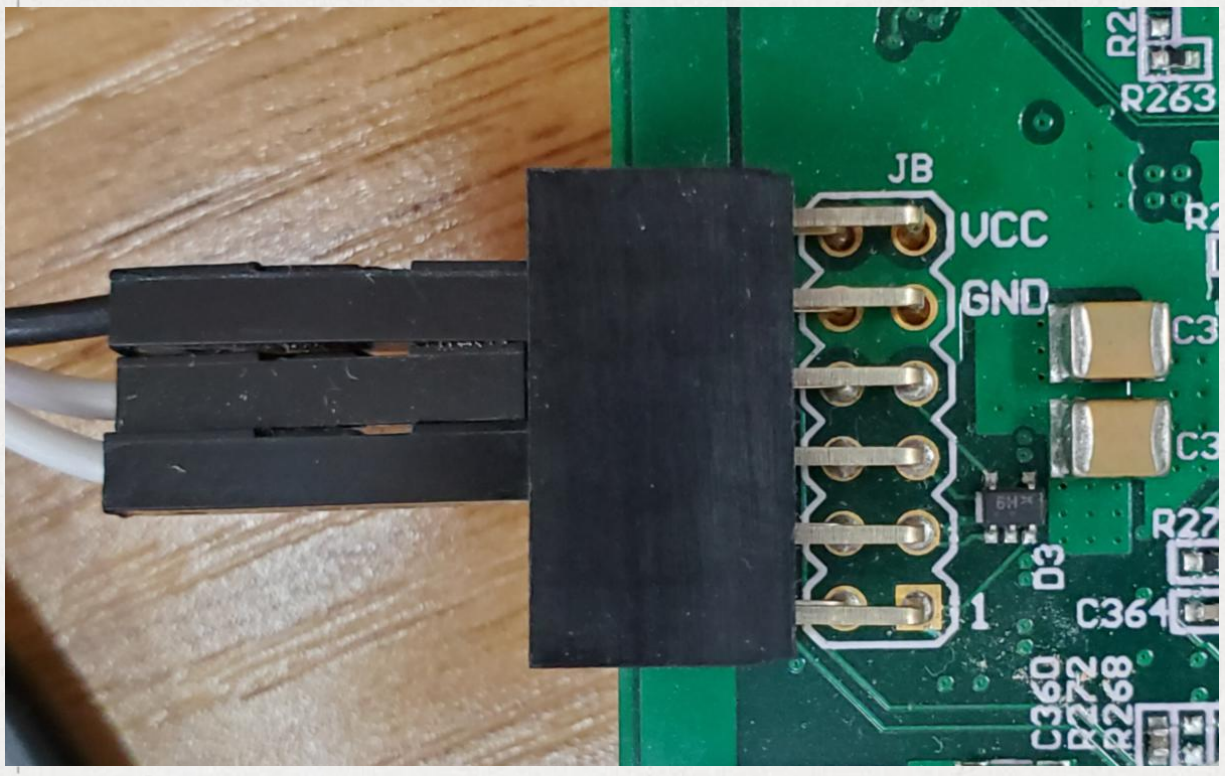
NexysVideo_Master.xdc - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
## Pmod header JB
set_property -dict { PACKAGE_PIN V9    IOSTANDARD LVCMOS33 } [get_ports { PAD_PWM_CH3 }]; #IO_L21P_T3_DQS_34 Sch=jb_p[1]
set_property -dict { PACKAGE_PIN V8    IOSTANDARD LVCMOS33 } [get_ports { PAD_PWM_CH4 }]; #IO_L21N_T3_DQS_34 Sch=jb_n[1]
set_property -dict { PACKAGE_PIN V7    IOSTANDARD LVCMOS33 } [get_ports { PAD_PWM_CH5 }]; #IO_L19P_T3_34 Sch=jb_p[2]
set_property -dict { PACKAGE_PIN W7    IOSTANDARD LVCMOS33 } [get_ports { PAD_PWM_CH6 }]; #IO_L19N_T3_VREF_34 Sch=jb_n[2]
set_property -dict { PACKAGE_PIN W9    IOSTANDARD LVCMOS33 } [get_ports { PAD_PWM_CH7 }]; #IO_L24P_T3_34 Sch=jb_p[3]
set_property -dict { PACKAGE_PIN Y9    IOSTANDARD LVCMOS33 } [get_ports { PAD_PWM_CH8 }]; #IO_L24N_T3_34 Sch=jb_n[3]
set_property -dict { PACKAGE_PIN Y8    IOSTANDARD LVCMOS33 } [get_ports { PAD_JTAG_TMS }]; #IO_L23P_T3_34 Sch=jb_p[4]
set_property -dict { PACKAGE_PIN Y7    IOSTANDARD LVCMOS33 } [get_ports { PAD_JTAG_TCLK }]; #IO_L23N_T3_34 Sch=jb_n[4]
```

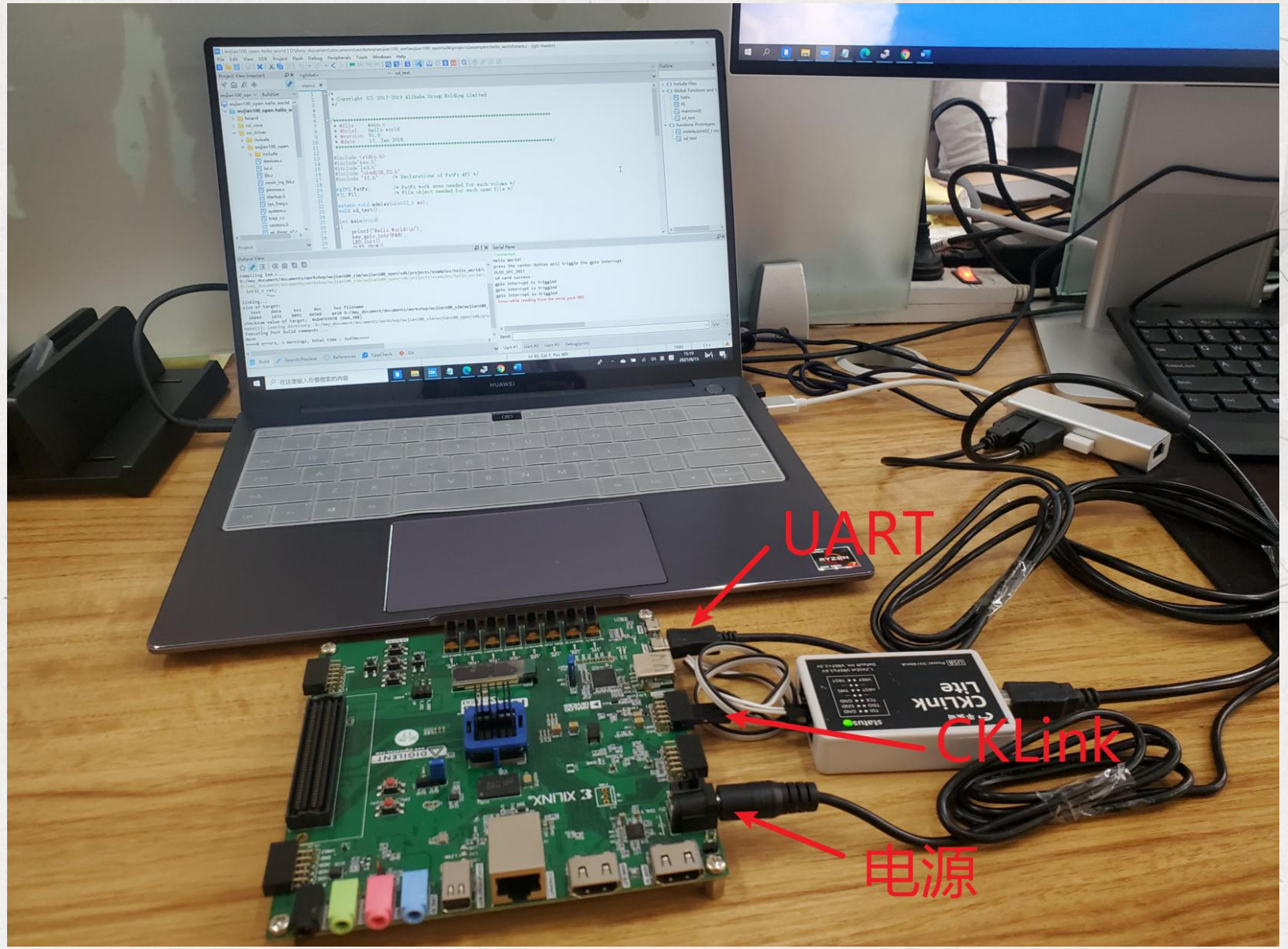


(1) 查看CKLINK与板子的连接方式


JB
VCC
GND
R263
C3
C3
R27
D3
C364
1

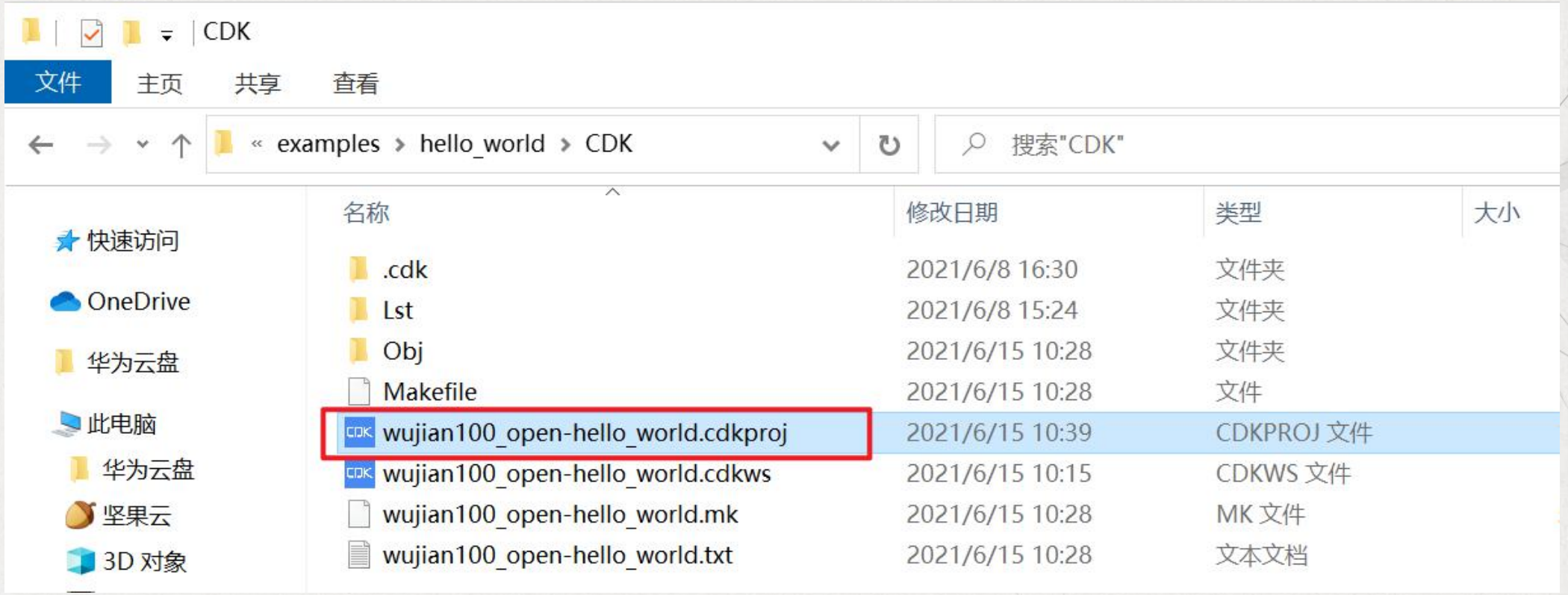

平头哥
CKLink
Lite
status
TDI GND
TDO GND
TCK GND
nRST TMS
VREF TRST
USB Power:5V/40mA
1.2V≤Ext.VREF≤3.6V
Default Int. VREF=3.3V

(2) 总体连接



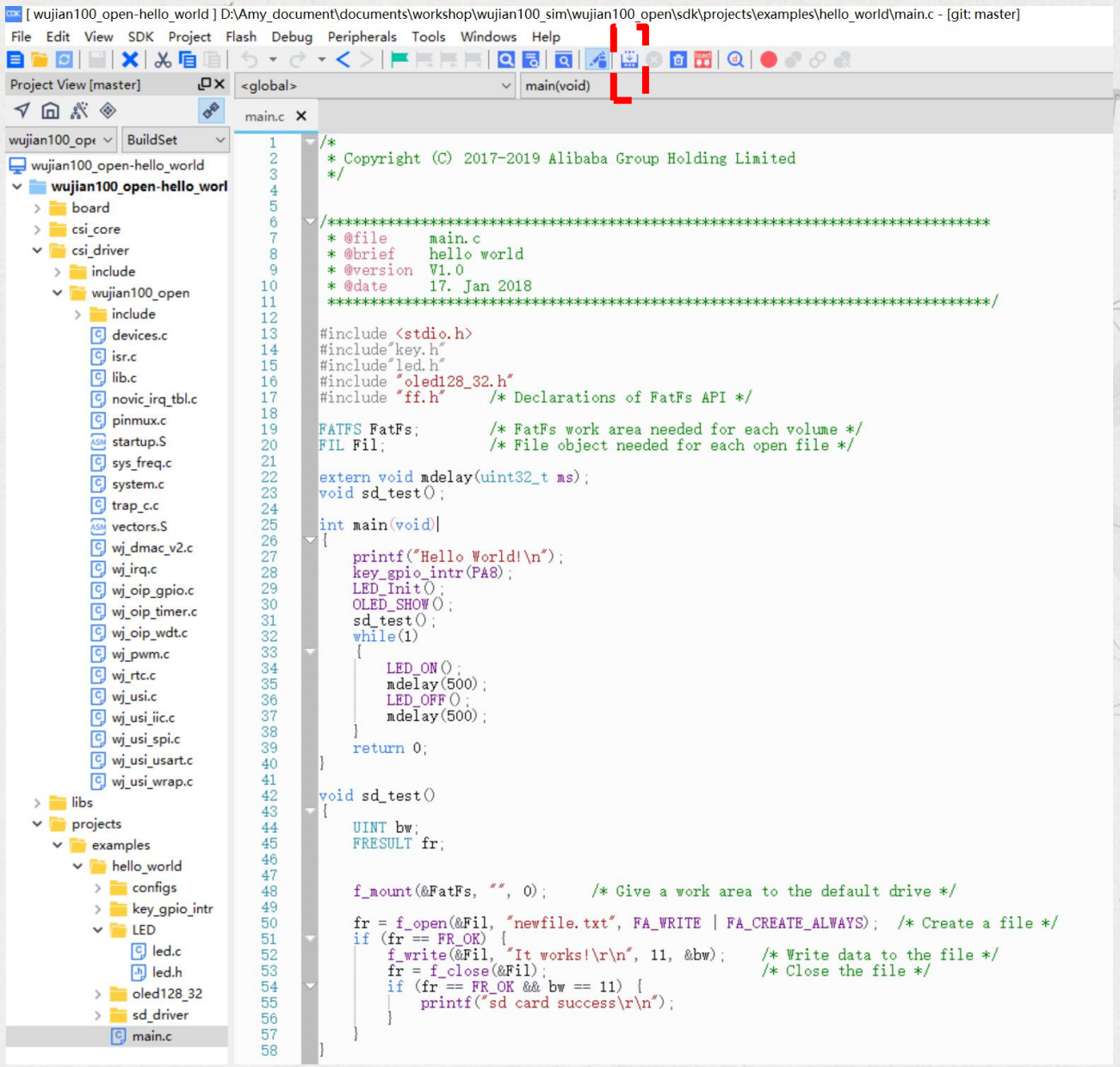
第二步

打开project、下载程序

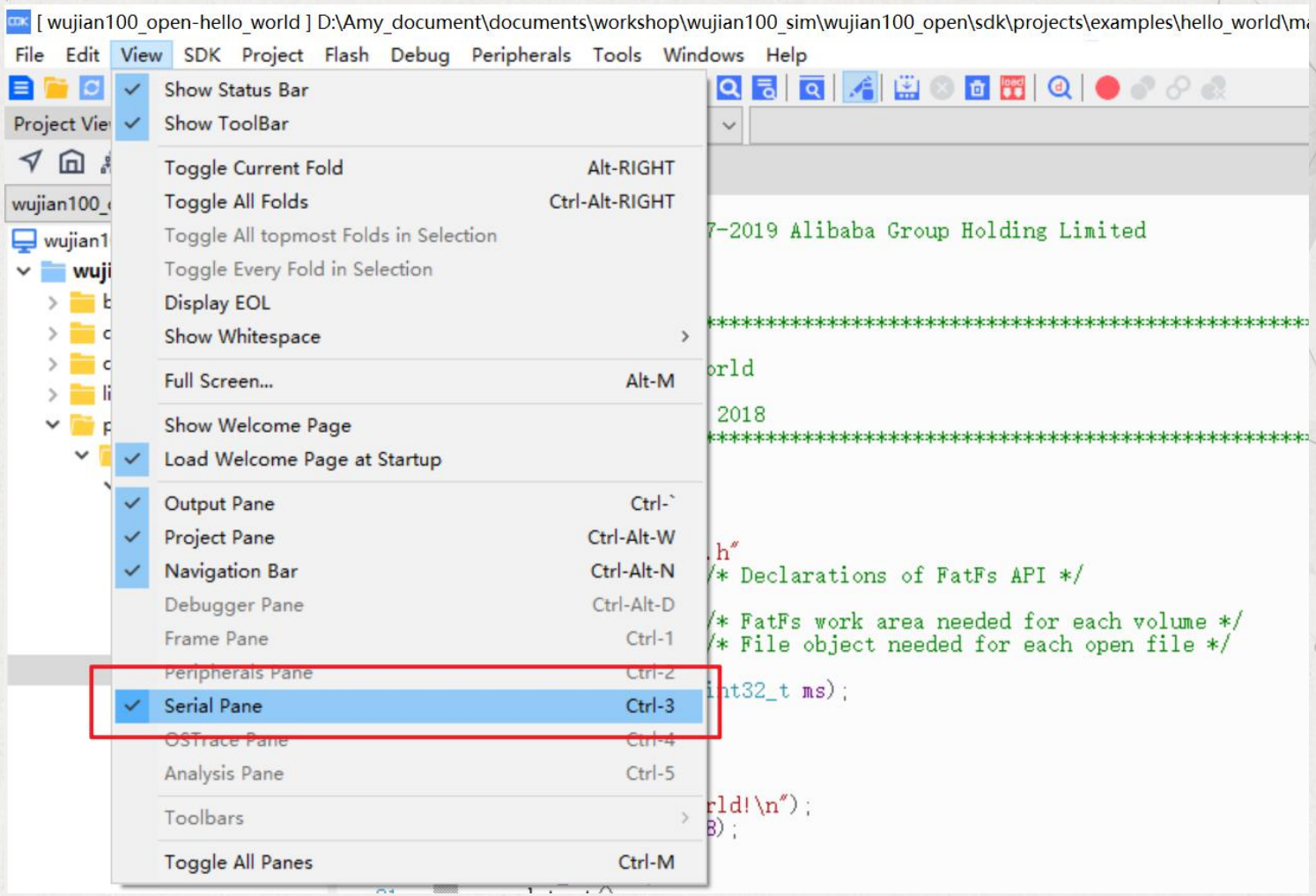


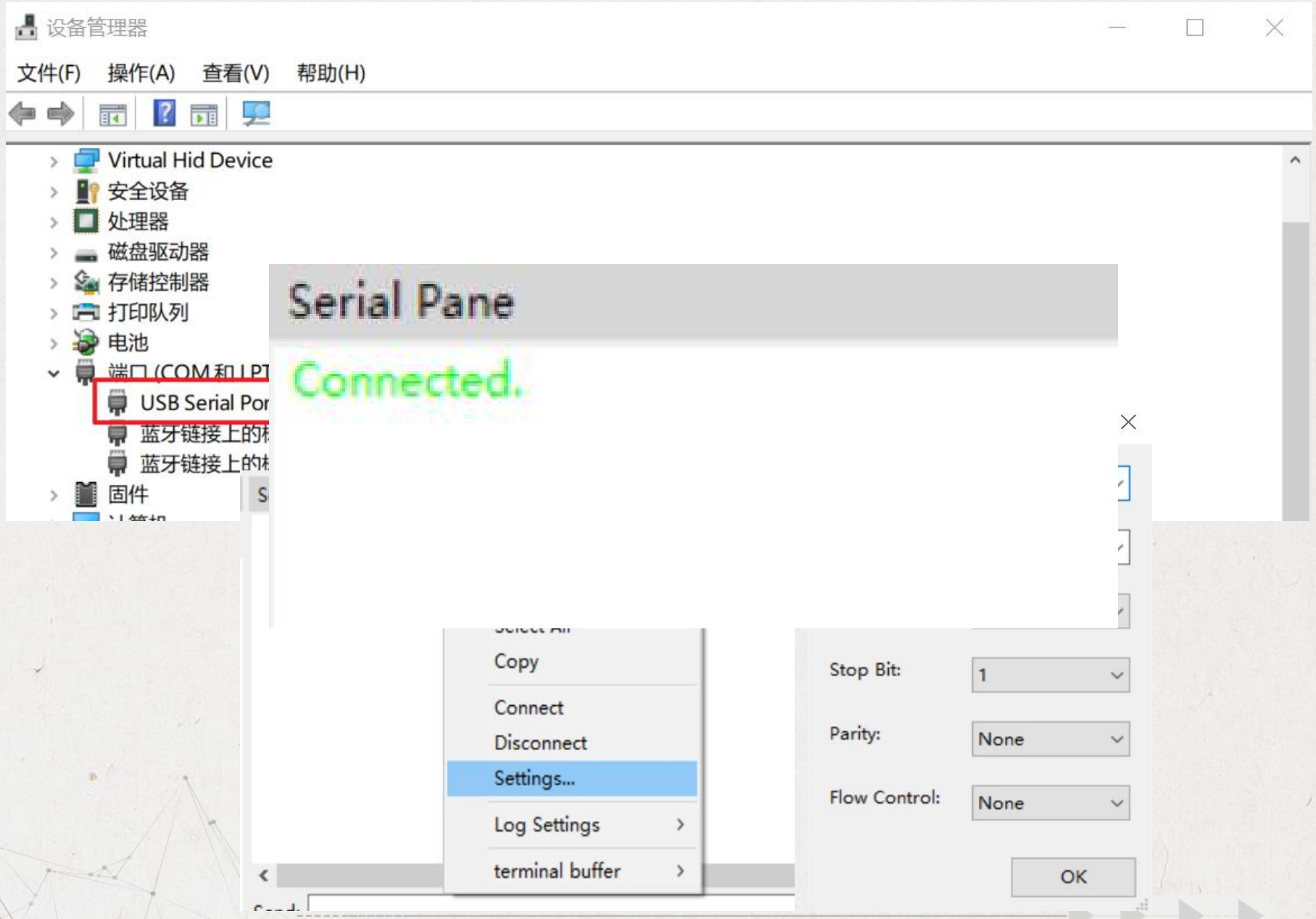
(1) 用CDK打开project

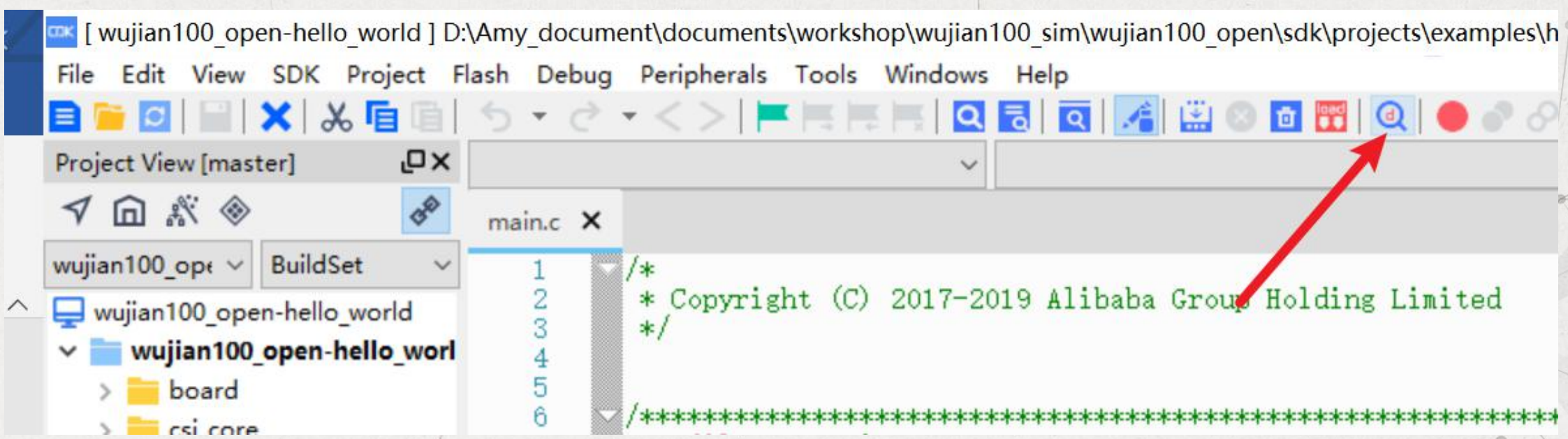
(2) 在CDK中打开main.c



(3) 打开串口界面，设置串口




设备管理器
文件(F)
操作(A)
查看(V)
帮助(H)
Virtual Hid Device
安全设备
处理器
磁盘驱动器
存储控制器
打印队列
电池
USB Serial Por
固件
Serial Pane
Connected.
Copy
Connect
Disconnect
Settings...
Log Settings
terminal buffer
Stop Bit:
1
Parity:
None
Flow Control:
None
OK



(4) 下载程序到板子上

点击一次开始调试，再点击一次终止调试，此时程序就下进板子里了

第三步 板上运行

Serial Pane

```
Connected.
Hello World!
press the center button will triggle the gpio interrupt
OLED_SPI_INIT
sd card success
```

(1) 首先观察串口界面的调试信息

(2) 观察LED闪烁情况和OLED显示内容

```
while(1)
{
    LED_ON();
    mdelay(500);
    LED_OFF();
    mdelay(500);
}
```

LED控制程序

```
void OLED_SHOW()
{
    OLED_Init();//初始化OLED
    OLED_ColorTurn(0);//0正常显示，1 反色显示
    OLED_DisplayTurn(1);//0正常显示 1 屏幕翻转显示
    OLED_Clear();
    OLED_ShowString(0,0,"Welcome to use",16);
    OLED_ShowString(16,2,"WujianSoC",16);
}
```

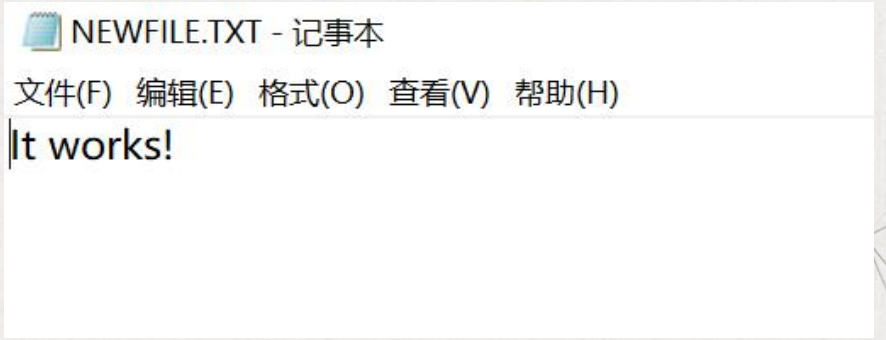
OLED显示程序

Serial Pane

```
Connected.
Hello World!
press the center button will triggle the gpio interrupt
OLED_SPI_INIT
sd card success
gpio interrupt is triggled
```

(3) 按center button测试中断功能

(4) 板子断电拔出SD卡，在电脑上查看是否写入内容



SD卡测试现象

```
void sd_test()
{
    UINT bw;
    FRESULT fr;


    f_mount(&FatFs, "", 0);      /* Give a work area to the default drive */

    fr = f_open(&Fil, "newfile.txt", FA_WRITE | FA_CREATE_ALWAYS);  /* Create a file */
    if (fr == FR_OK) {
        f_write(&Fil, "It works!\r\n", 11, &bw);     /* Write data to the file */
        fr = f_close(&Fil);                           /* Close the file */
        if (fr == FR_OK && bw == 11) {
            printf("sd card success\r\n");
        }
    }
}
```

SD卡测试程序