小眼睛科技

提供一站式 FPGA&嵌入式解决方案

《国产 FPGA 权威开发指南》教材

配套实验例程-基于盘古 676 MINI 系列

小眼睛科技 盘古 676 系列

盘古 100K MINI（MES2L676-100HP-MINI-MINI）开发板实验教程

紫光同创 Logos2 系列 PG2L100H 开发平台

文档版本修订记录

版本号	发布日期	修订记录
V1.0	2025/10/20	初始版本

公司名称: 深圳市小眼睛科技有限公司

地址: 深圳市宝安区西乡街道 F518 时尚创意园

官方网址: www.meyesemi.com

官方淘宝店铺: 小眼睛半导体

B 站: 小眼睛半导体 (视频教程免费学)

* 加入 FPGA 开发者技术交流与 5000+FPGA 开发者实时沟通

QQ2 群: 442106123QQ3 群: 882634519)

*配套资料下载、技术答疑请登录逻辑矩阵技术论坛

LOGIC MATRIX 逻辑矩阵

逻辑矩阵技术论坛欢迎各位发烧友加入

让我们共建开源生态, 持续赋能行业发展

https://www.szlogicmatrix.com/

*扫码注册开源技术论坛

*扫一扫关注官微

* 官方旗舰店

目录

1. LED 流水灯

1.1. 实验简介

实验目的:

控制 8 个 LED 灯按顺序依次点亮和熄灭。

实验环境:

Window11

PDS2022.2-SP6.4

硬件环境:

MES2L676-100HP-MINI

1.2. 实验原理

通常的时, 分, 秒的计时进位大家应该不陌生;

1 小时=60 分钟=3600 秒, 当时针转动 1 小时, 秒针跳动 3600 次;

在数字电路中的时钟信号也是有固定的节奏的, 这种节奏的开始到结束的时间,我们通常称之为周期 (T)。

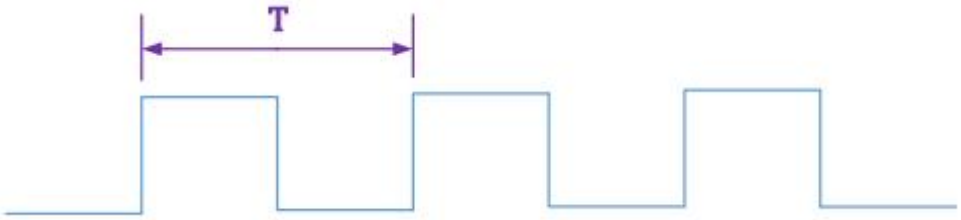


图 1.2-1

在数字系统中通常关注到时钟的频率,那频率与周期的关系如下:

$$f = 1/T$$

在 100K 板卡上单端时钟有一个 27MHZ 的时钟。

所以其周期约为 37.037ns。而在我们 FPGA 的设计中, 我们的 always 块通常都是在时钟的上升沿时对数据进行赋值, 因此我们可以定义一个变量, 每到时钟的上升沿就让该变量+1, 让其变成一个计数器, 该变量每加 1 就表示经过了 37.037ns, 那如果要定时 1s 的话, 只需要让其计数到 26999999 即可, 因为从 0 开始计数, 所以计数到 26999999 即可, 此时就是一秒了。以此类推, 13499999 就是 0.5s。

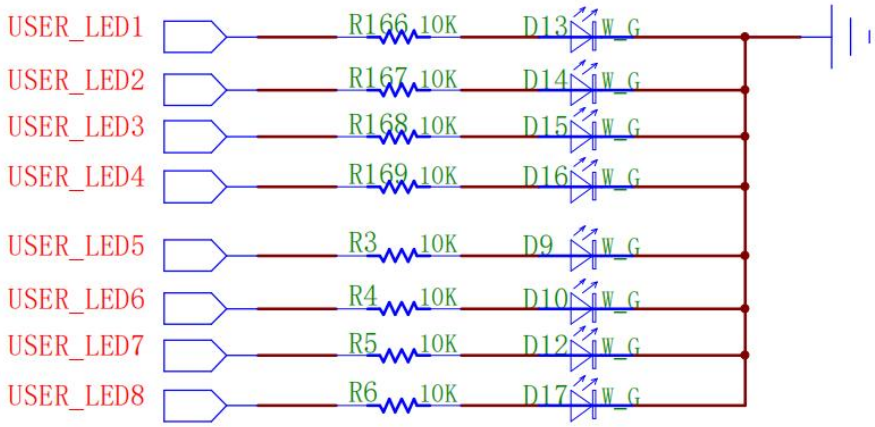


图 1.2-2

错误!未找到引用源。为开发板上 8 个 LED 灯的原理图。

控制 0.5s 更换一次 LED 灯状态, 使 LED 灯呈现流水灯现象, 可以每 0.5s 依次点亮一个 LED 灯。(高电平用 1 表示, 低电平用 0 表示)

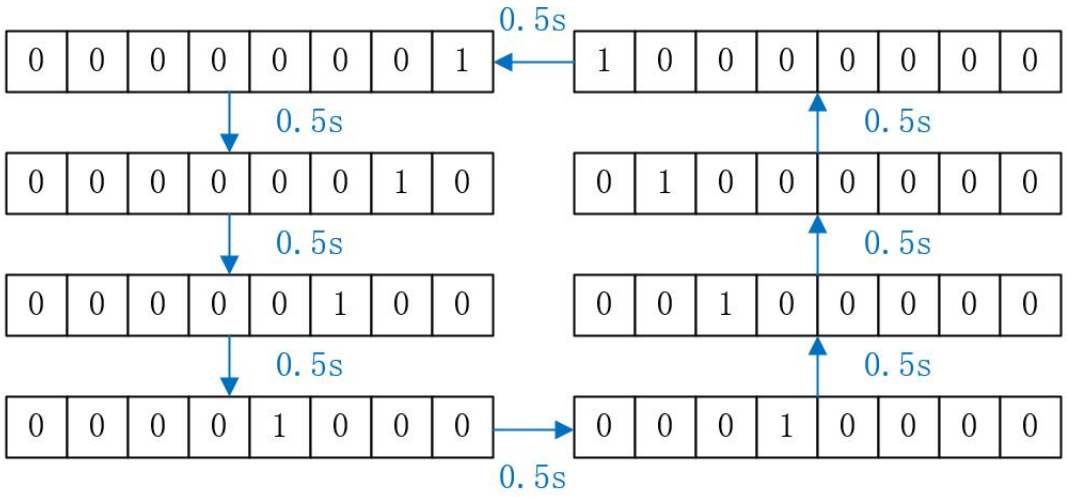


图 1.2-3

故只需要定义一个变量,让其在时钟上升沿达到时就+1,计数到 13500000-1 即可,此时就是 0.5S。

1.3. 接口列表

端口	I/O	位宽	描述
CNT_MAX	**Parameter**	**26**	**计数的最大值，修改定时的时间。**
clk	input	1	系统时钟, 27MHZ

rst_n	input	1	复位信号, 低有效
led[7:0]	output	8	led 灯控制信号

1.4. 工程说明

```
module led_test
#(
   parameter    CNT_MAX = 26'd13_500_000
)
(
   input        clk,
   input        rstn,

   output [7:0]   led
);

//=============================================================================
======
//reg and wire

   reg [25:0] led_light_cnt    = 26'd0       ;
   reg [ 7:0] led_status       = 8'b0000_0001  ;

//time counter
   always @(posedge clk)
   begin
      if(!rstn)
         led_light_cnt <= `UD 26'd0;
      else if(led_light_cnt == CNT_MAX-1)
         led_light_cnt <= `UD 26'd0;
      else
         led_light_cnt <= `UD led_light_cnt + 26'd1;
   end

//led status change
```

```
    always @(posedge clk)
    begin
      if(!rstn)
        led_status <= `UD 8'b0000_0001;
      else if(led_light_cnt == CNT_MAX-1)
        led_status <= `UD {led_status[6:0],led_status[7]};
    end

    assign led = led_status;

endmodule
```

代码的第三行所定义的参数 CNT_MAX 是用来设定计数的最大值, 默认是 13500000, 也就是定时 0.5S, 可以通过修改该值来改变定时的时间。

代码的 20-28 行, 用变量 led_light_cnt 实现了定时器的功能, 每到时钟的上升沿就让其加 1, 不断计数。由于从 0 开始计数, 所以计数到 CNT_MAX-1 即把它清 0。

代码的 31-37 行, 实现了 LED 灯的状态控制, led_status 是个 8bit 的变量, 当 Led_light_cnt=CNT_MAX-1 时,就让 led_status 向左移 1 位。36 行的写法也可以换成 led_status<<1。最终的效果就是 0000_0001->0000_0010->0000_0100.....->1000_0000。实现流水灯的效果。

代码 39 行, 就通过组合逻辑, 将 led_status 的值赋值给 led。

1.5. 代码仿真

```
module tb_led_test();

reg       clk    ;
reg       rst_n   ;
wire[7:0]   led    ;
reg [7:0]   data    ;

initial begin
    rst_n <= 0;
    clk   <= 0;
    #20;
    rst_n <= 1;
    #2000
    $display("I am stop"); //
    $stop;
```

```
end
always#10 clk = ~clk;//20ns  50MHZ

led_test
#(
   .CNT_MAX    (10)
)u_led_test(
 .clk      (clk    ),// input
 .rstn     (rst_n  ),// input
 .led      (led    ) // output [7:0]
);

initial begin
   $monitor("led:%b", led);
end

always@(posedge clk or negedge rst_n)    begin
   if(!rst_n)
      data    <=    8'd0;
   else
   begin
      data    <=    {$random}%256;
      $display("Now data is %d",data);
   end
end


endmodule
```

该 testbench 部分代码是用于测试一些仿真函数使用, 在前面的 Modelsim 的使用章节已经做了介绍,所以本次只关注代码的 19-26 行,例化了我们的流水灯模块。可以看到代码的 21 行,CNT_MAX 传入的参数给的是 10, 这是为了减少仿真的时间, 如果还是 13500000 的话, 我们需要仿真 0.5s 才能看到结果, 这非常的久。

代码的 8-16 行依旧是对复位和时钟赋初值, 延时 20ns 后, 复位结束。时钟依然是每隔 10ns 取反一次, 来生成 50MHZ 的时钟。

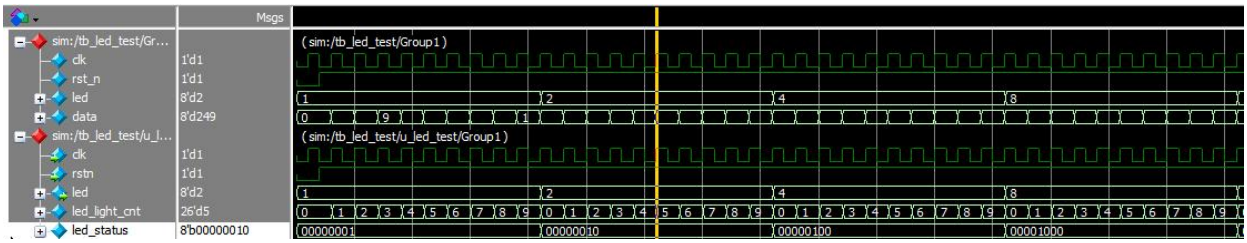


图 1.5-1

错误!未找到引用源。为仿真结果，可以看到 led_light_cnt 每次计数 10 个数据后，led_status 会向左移一位，符合实验结果。

1.6. 实验步骤

这里将会详细介绍从新建工程到下载程序的具体步骤，后续的工程将不再详细解释。

1.6.1.打开 PDS 软件，创建工程

Step1：打开 PDS 软件，点击 NEW Project，然后对其设置完成新建工程；

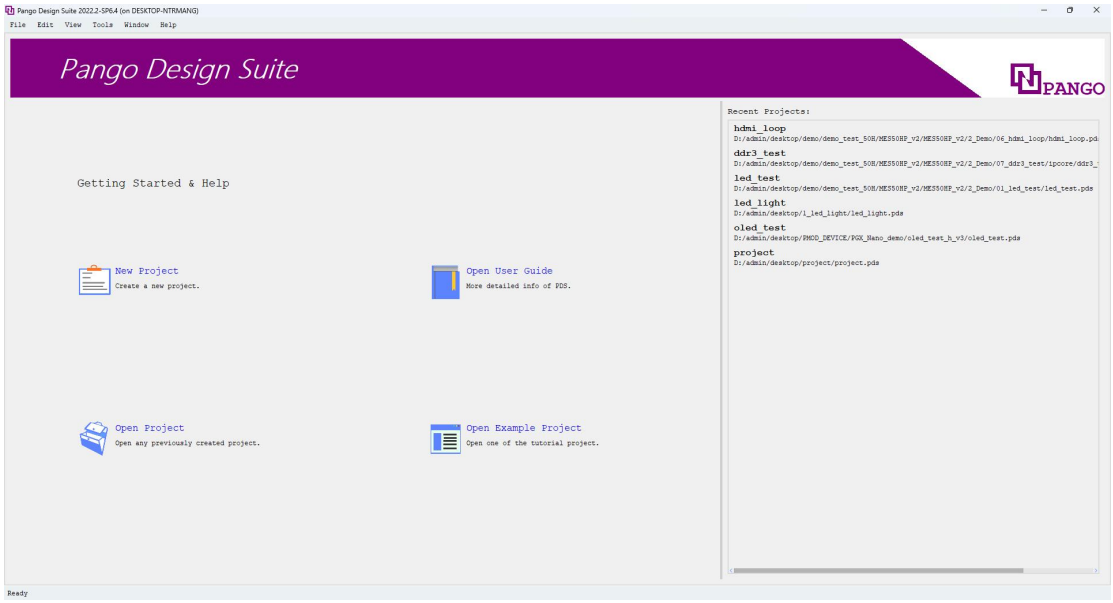


图 1.6-1

Step2：单击 NEXT；

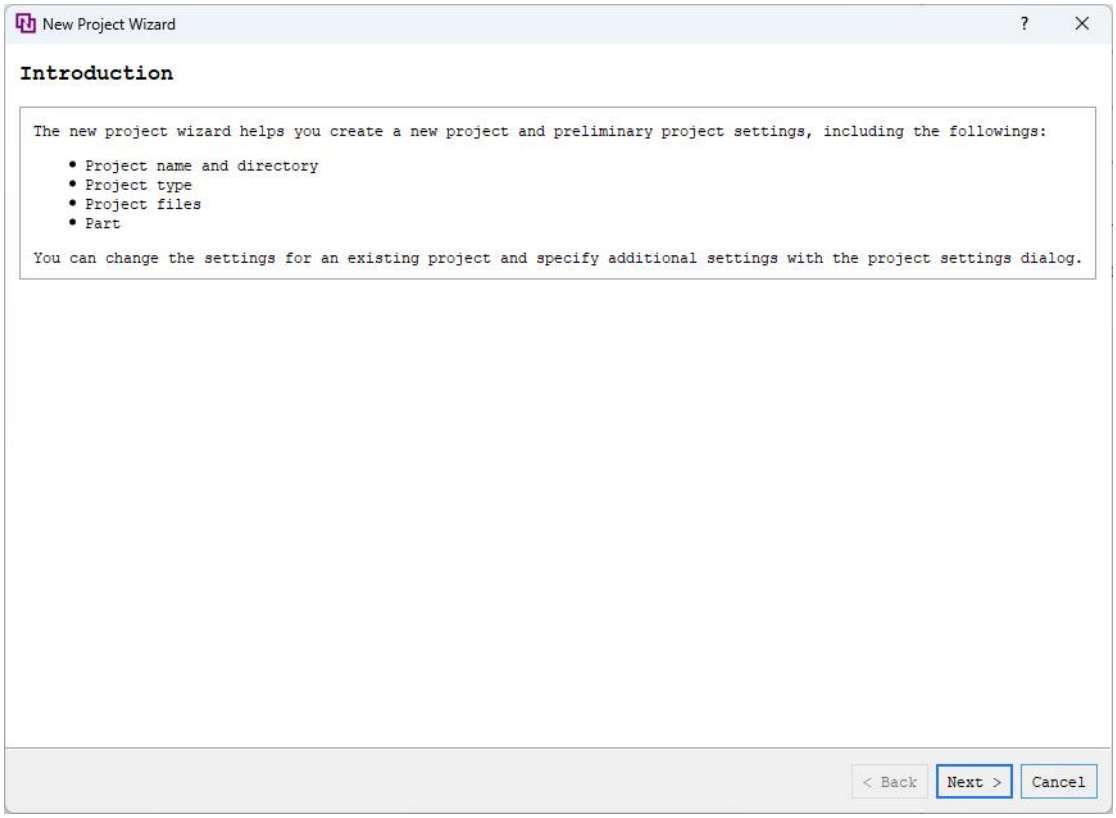


图 1.6-2

Step3: 创建名为 led_water 的工程到对应的文件目录, 之后单击 Next;

新建工程大致包括设置工程名和工程路径、工程类型、工程文件及器件信息。

【Project Name】是工程文件名称, 默认为 project。(只允许字母、数字、下划线 (_)、杠 (-)、点 (.))。

【Project Location】用于选择新工程的工作路径, 文件夹名只允许字母、数字、下划线 (_)、杠 (-)、点 (.)、@、~、,、、+、=、#、空格 ( ), 但空格不能出现在路径名首尾, 即工程文件放置的路径。

【Create Preject Subdirectory】将工程文件名作为工作目录的一部分。

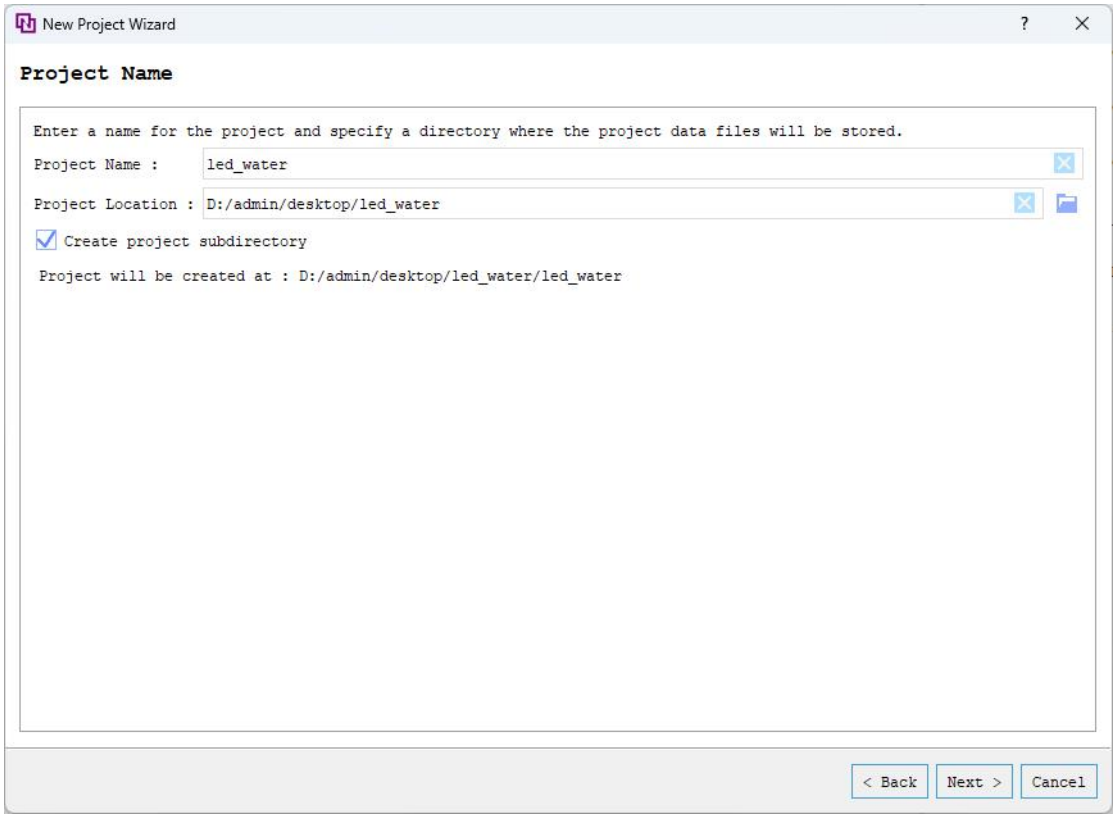


图 1.6-3

Step4: 选择 RTL project, 点击 Next;

【RTL Project】用于创建 RTL 工程。新建的工程可以执行 synthesize, device map, place& route, report timing, report power, generate netlist 及 generate bitstream 等。

【Post-Synthesize Project】用于创建综合后工程。新建的工程可以执行 device map, place& route, report timing, report power, generate netlist 及 generate bitstream 等。

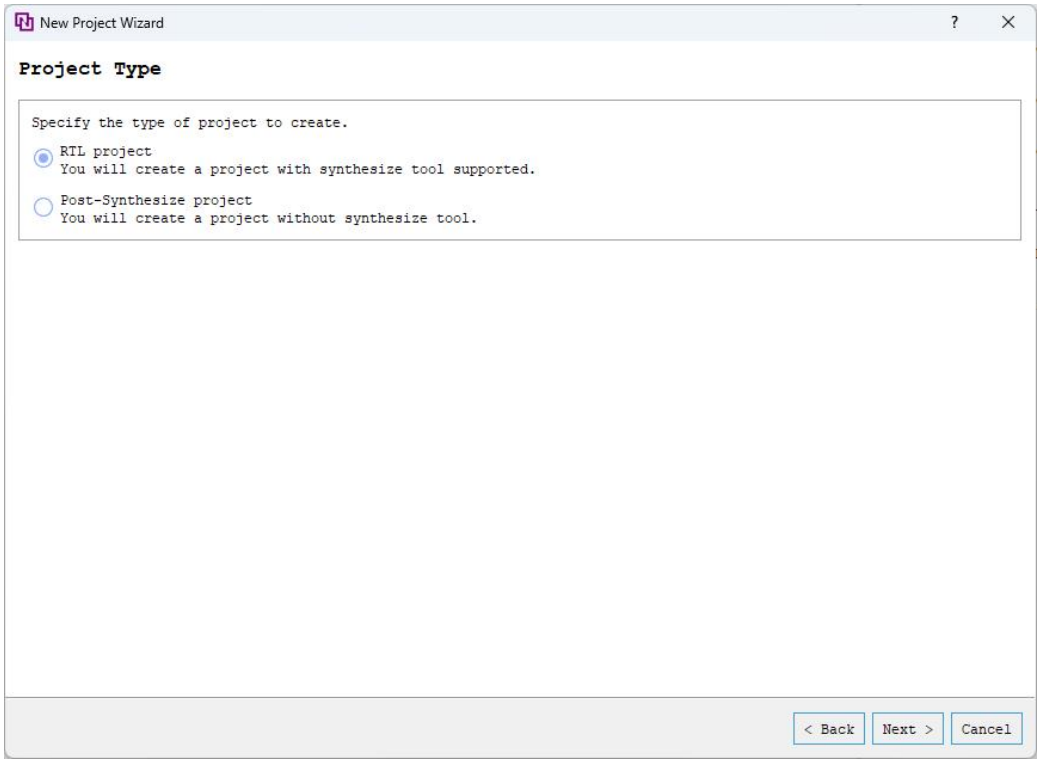


图 1.6-4

Step5: 单击 Next;

该界面可以 Add Files 和 Add Directories 来添加 rtl 源文件及新建 rtl 源文件, 以及调整 rtl 文件编译顺序, Add Files 添加选中的文件, Add Directories 添加选中的文件夹下所有合适的文件, 若勾选了下方的 Add source from subdirecotires 则添加所有的子目录下合适的文件, 也可直接 NEXT 跳过添加文件。

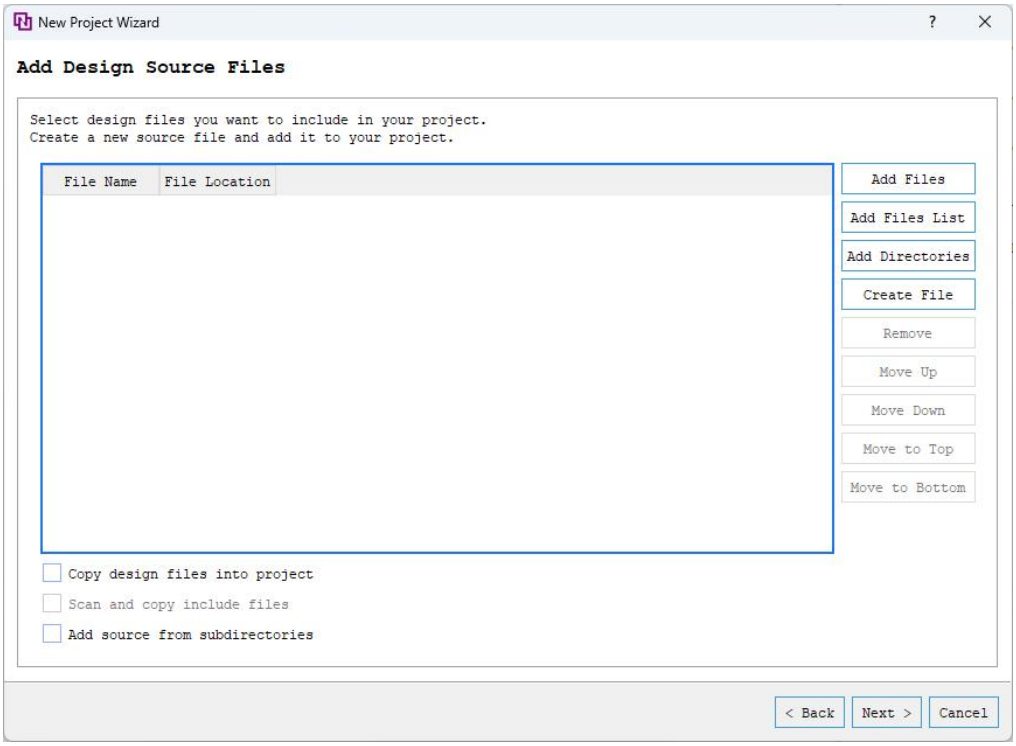


图 1.6-5

Step6: 单击 Next;

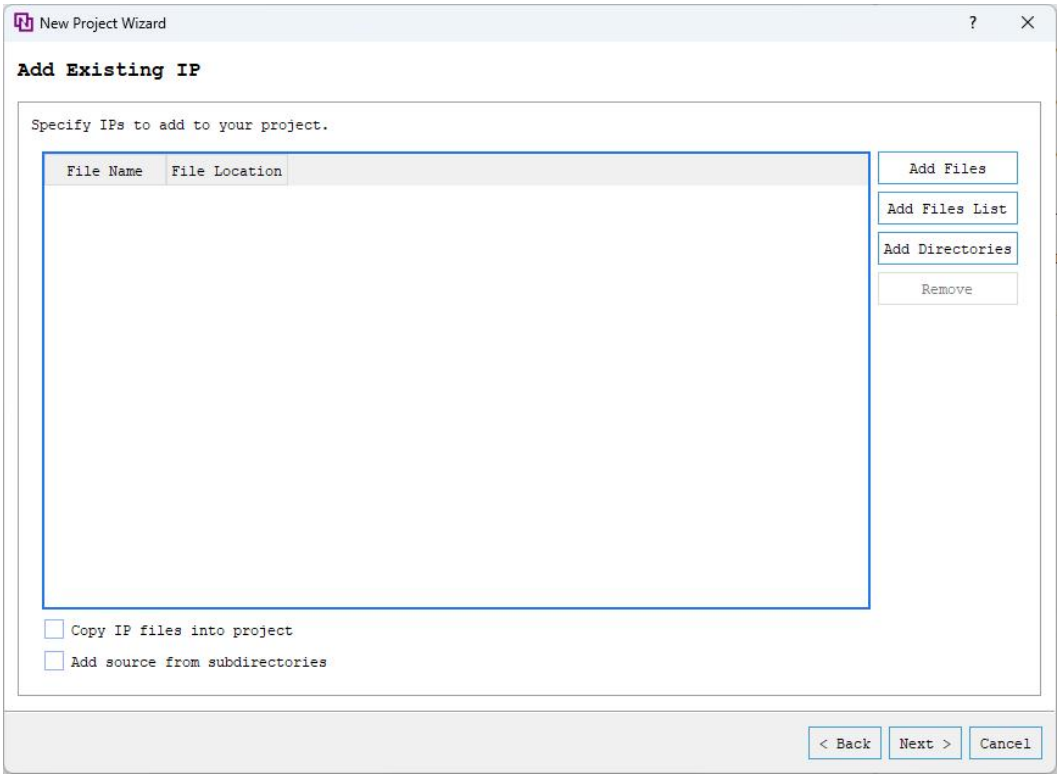


图 1.6-6

Step7: 单击 Next;

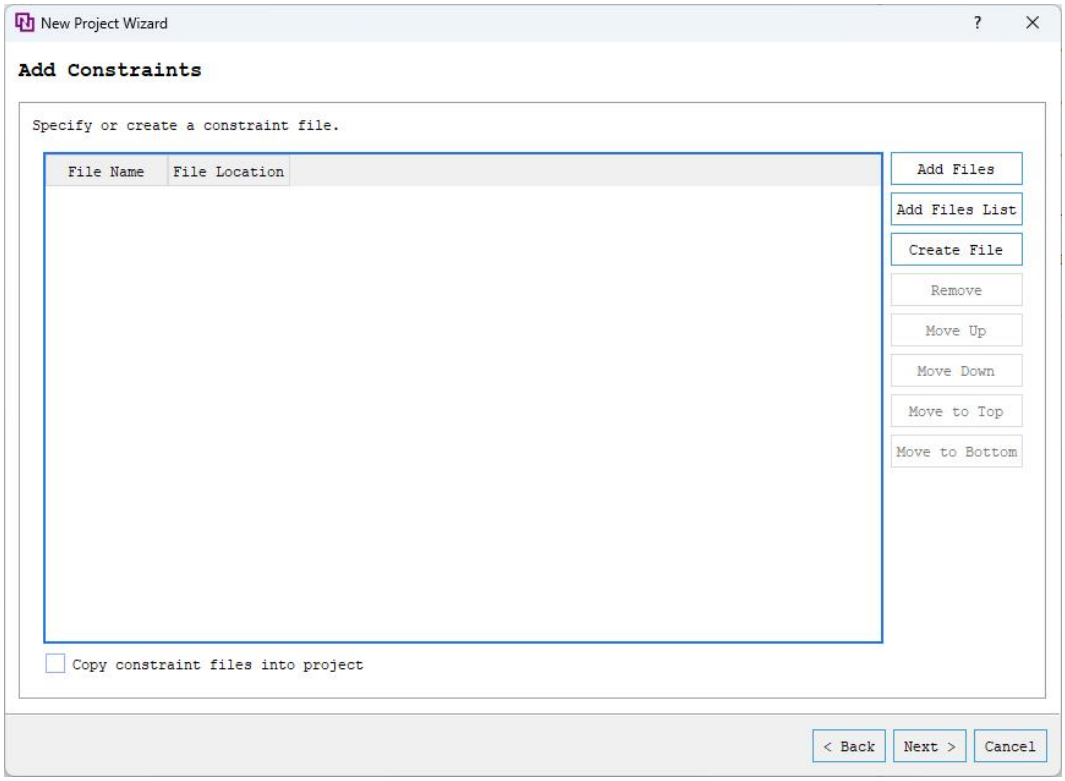


图 1.6-7

Step8: 选择器件系列、型号、封装、速率、综合工具, 之后单击 Next;

synthesize tool 中可以选择综合工具为 Synplify Pro 或 ADS, 在实验中使用 ADS 综合工具。

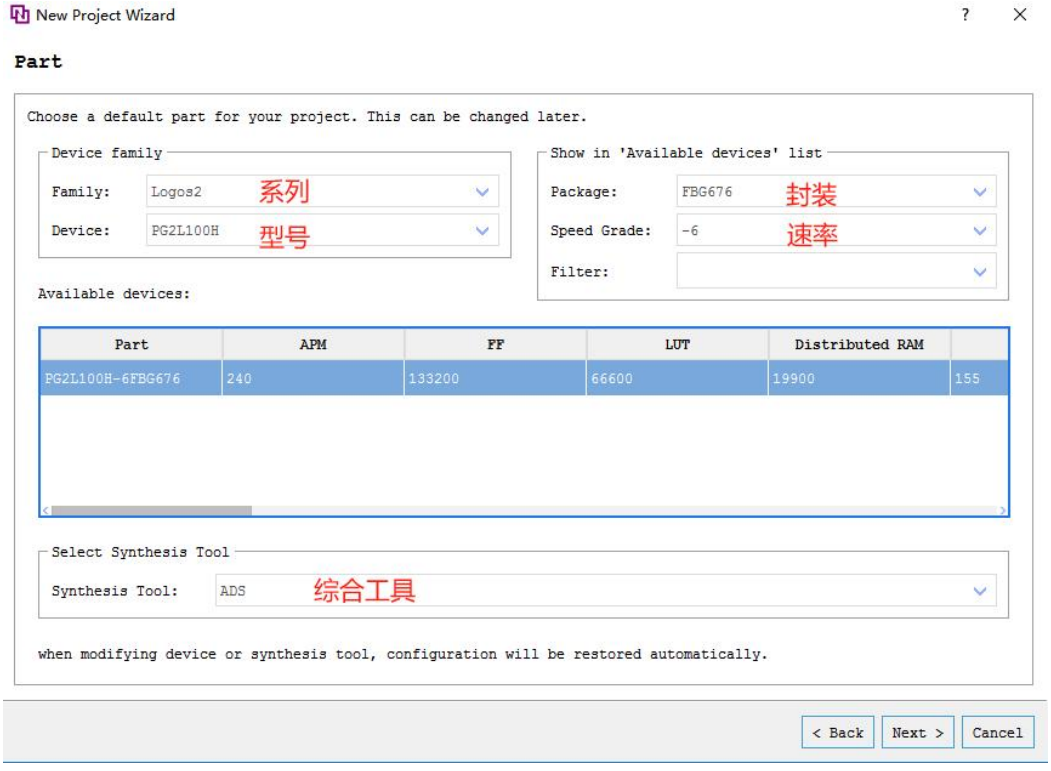


图 1.6-8

Step9: 在 summary 单击 Finish, 完成工程的创建;

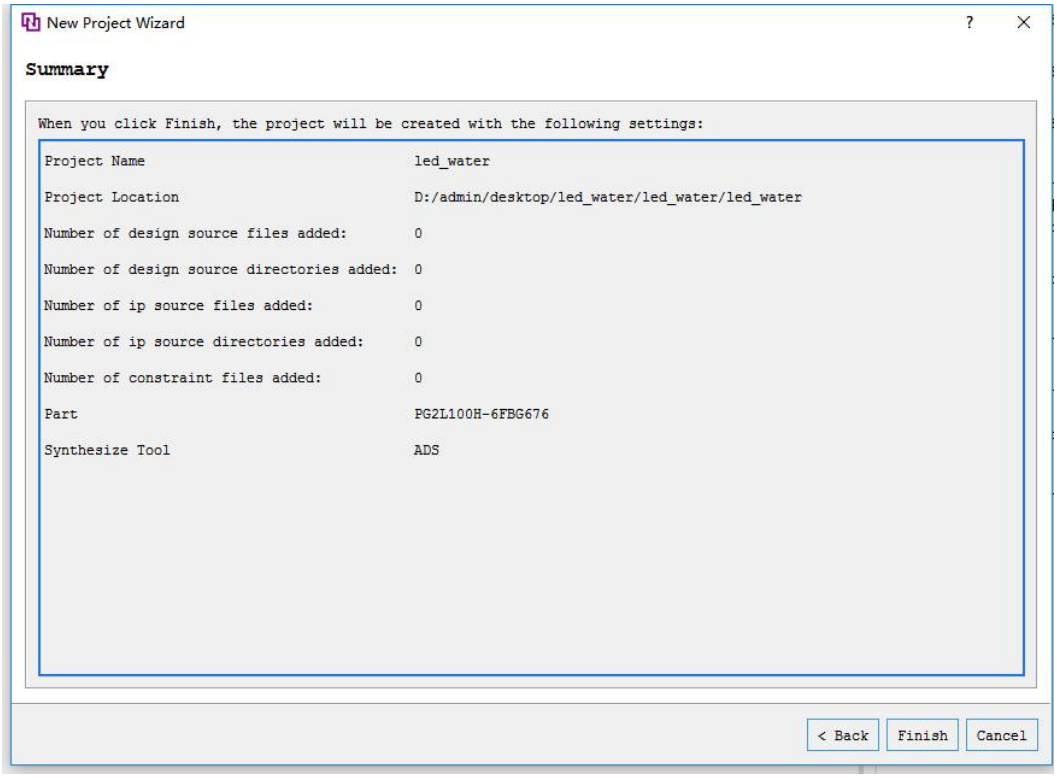


图 1.6-9

1.6.2.添加设计文件

PDS 软件界面如下图:

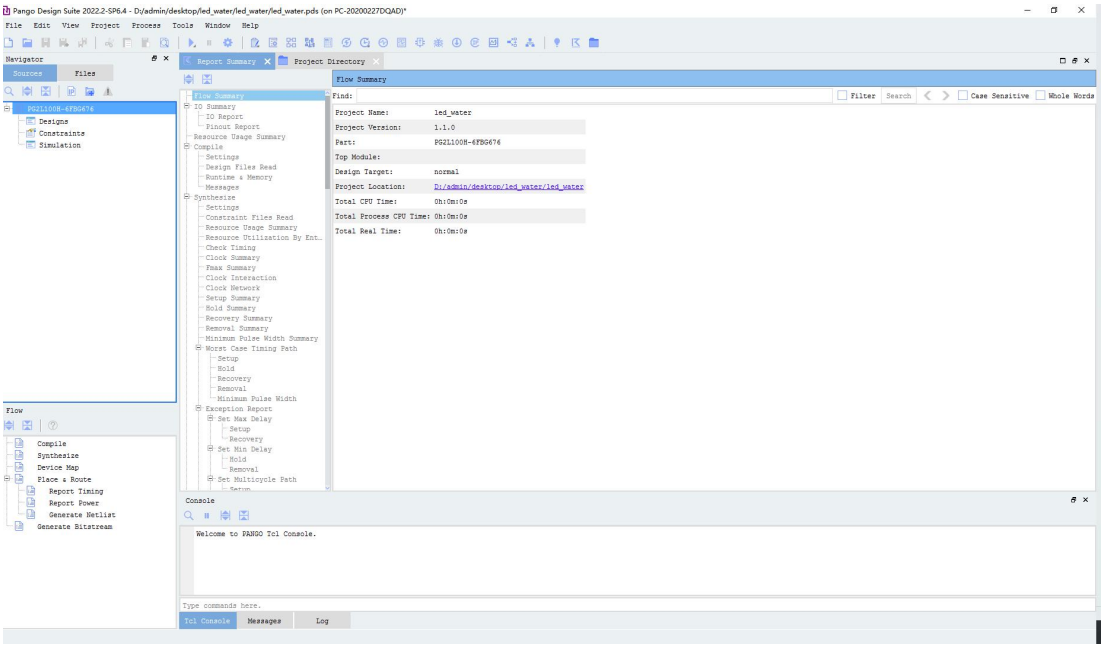


图 1.6-10

双击 Designs, 将前面设计的 module 新建到文件中, 或者将前面编辑好的 verilog 文件添加到工程中;

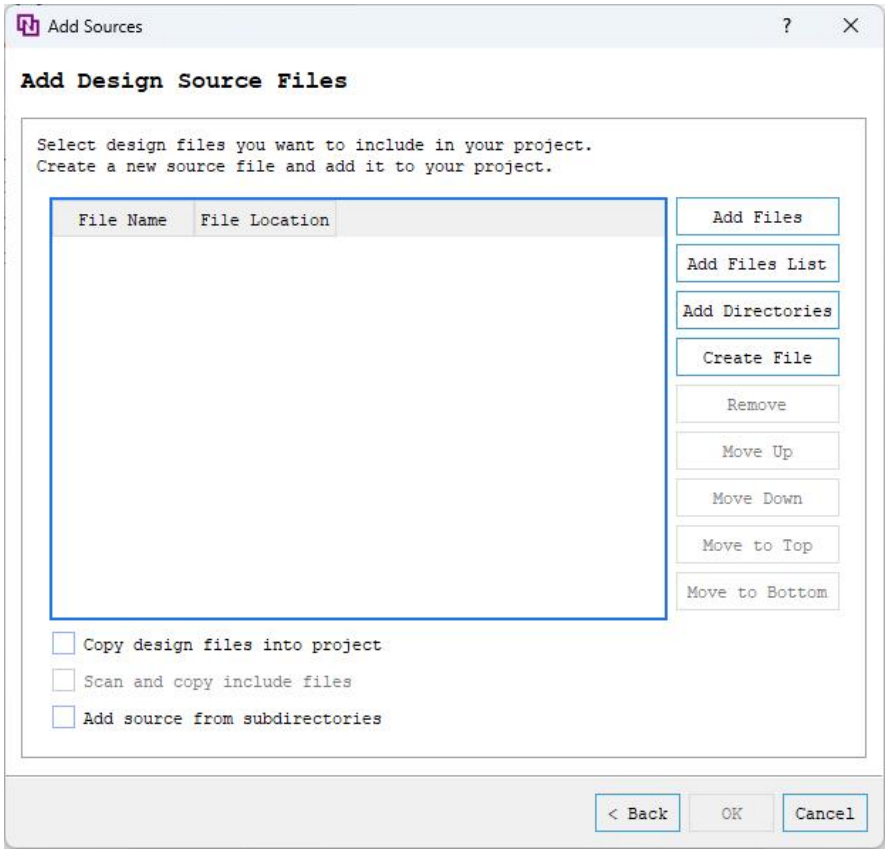


图 1.6-11

添加文件到工程:

在窗口中点击 Add Files, 选择添加文件到工程;

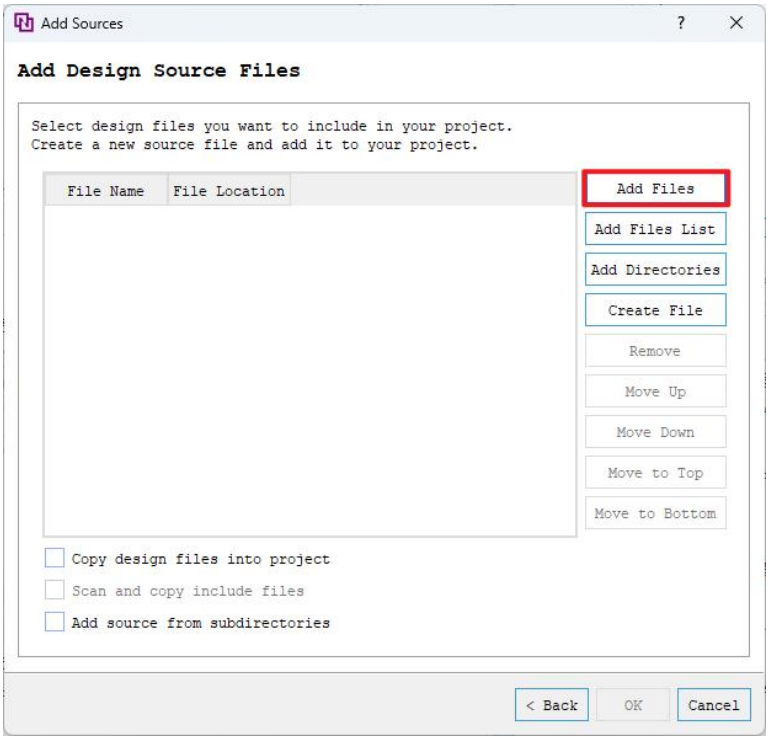


图 1.6-12

新建文件到工程:

1)在窗口中点击 Create File;

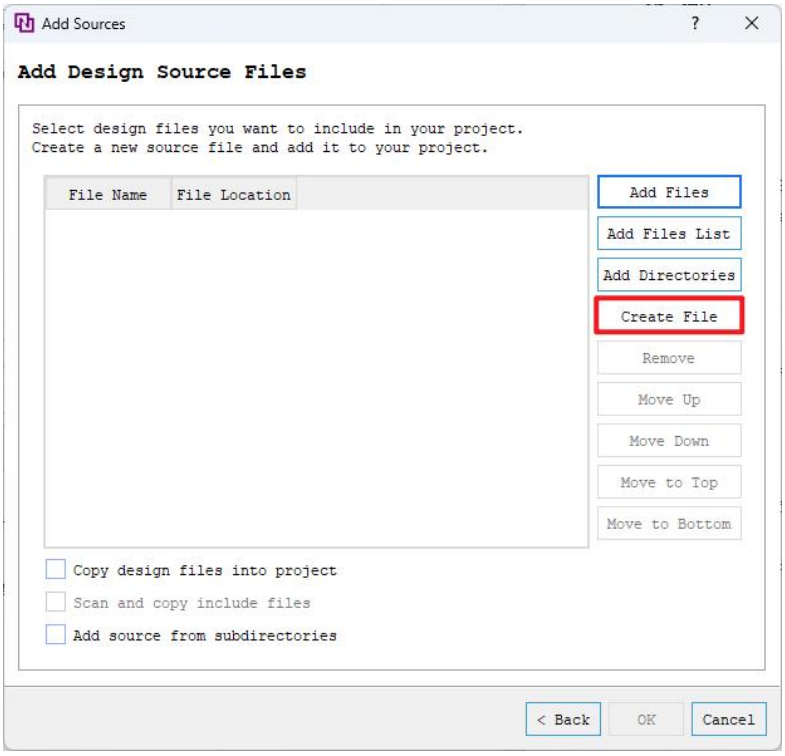


图 1.6-13

2)选择 Verilog Design File, 文件名和 module 名一致, 默认路径, 点击 OK;

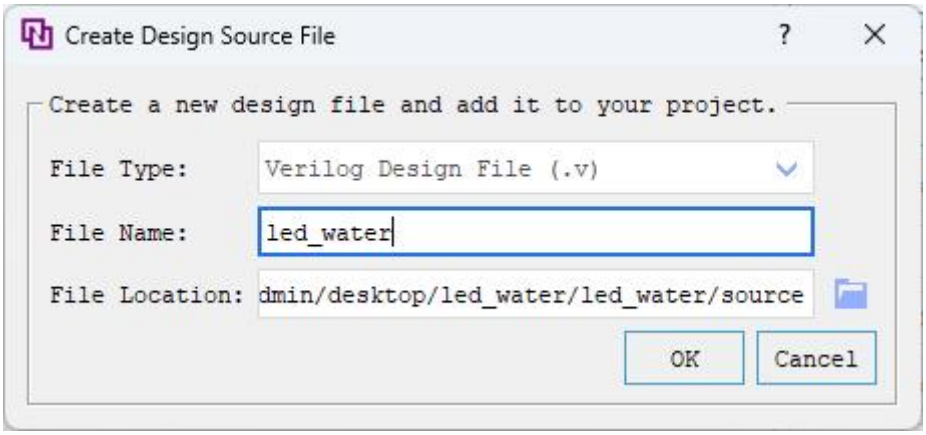


图 1.6-14

点击 OK;

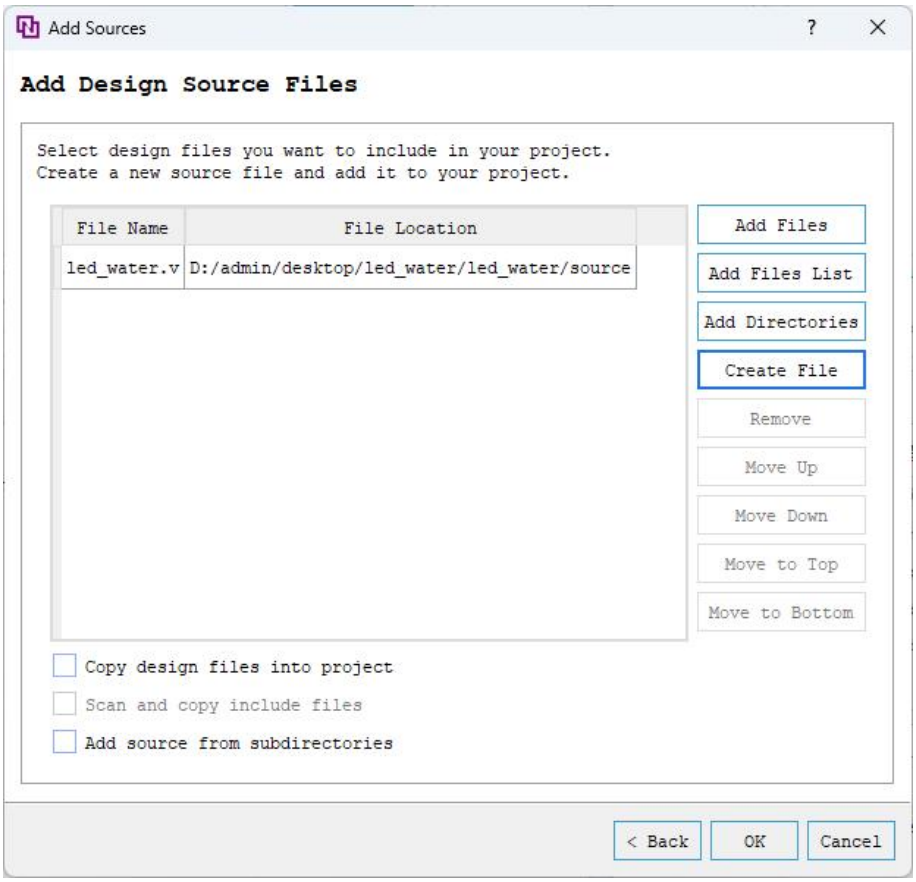


图 1.6-15

4)点击 Cancel;

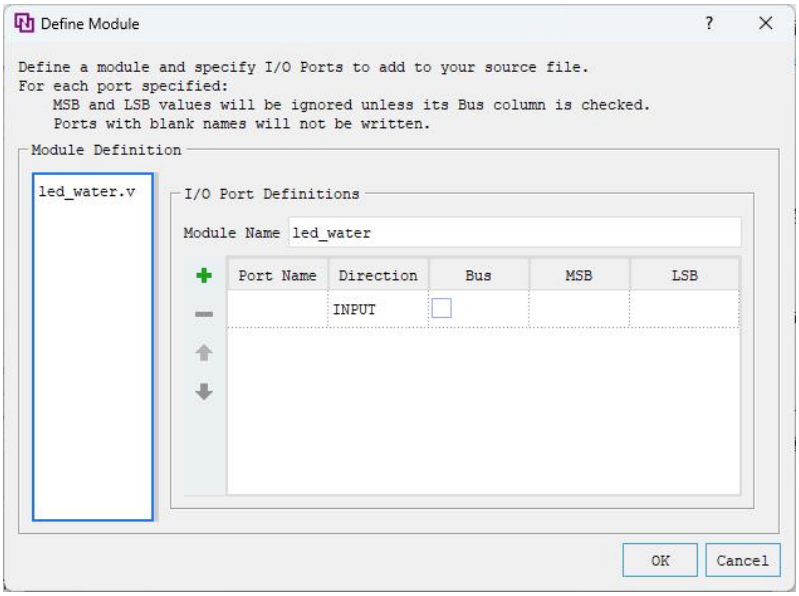


图 1.6-16

5)默认打开新建文件, 将前面设计的 module 复制进去;

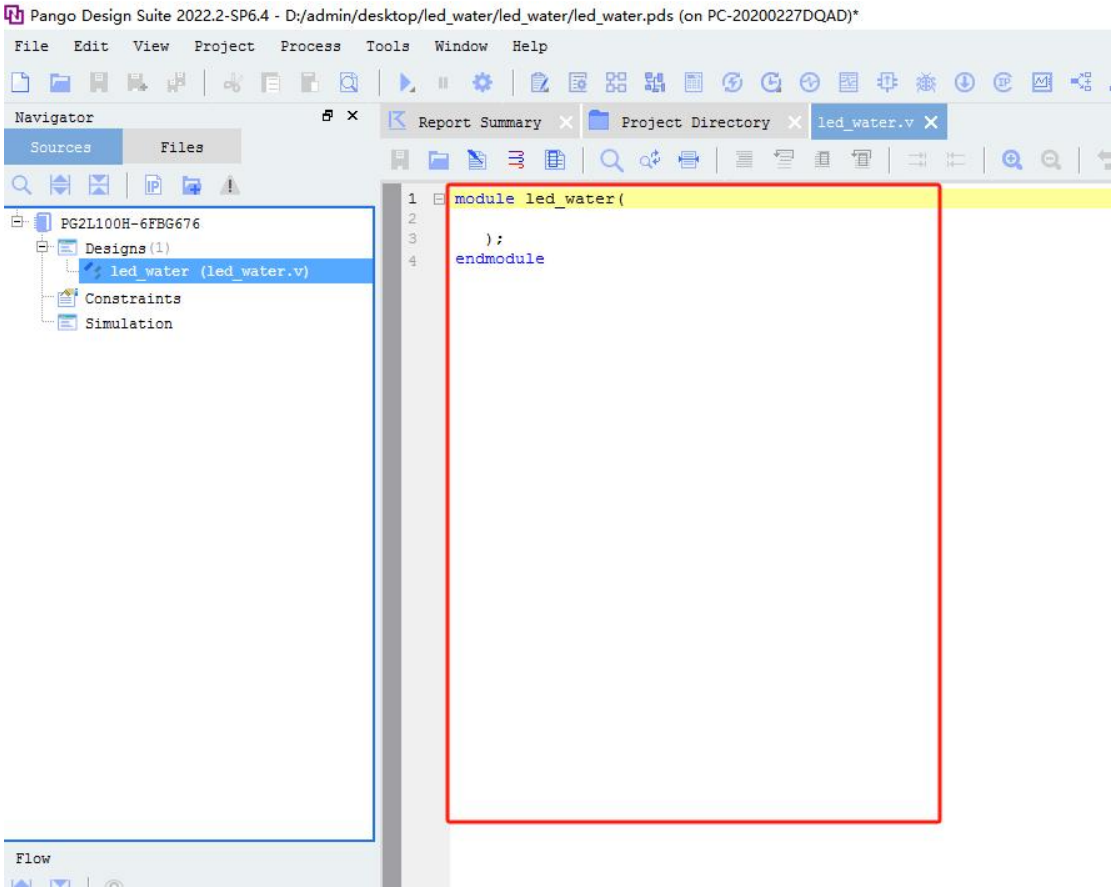


图 1.6-17

点击保存, 新建文件完成。

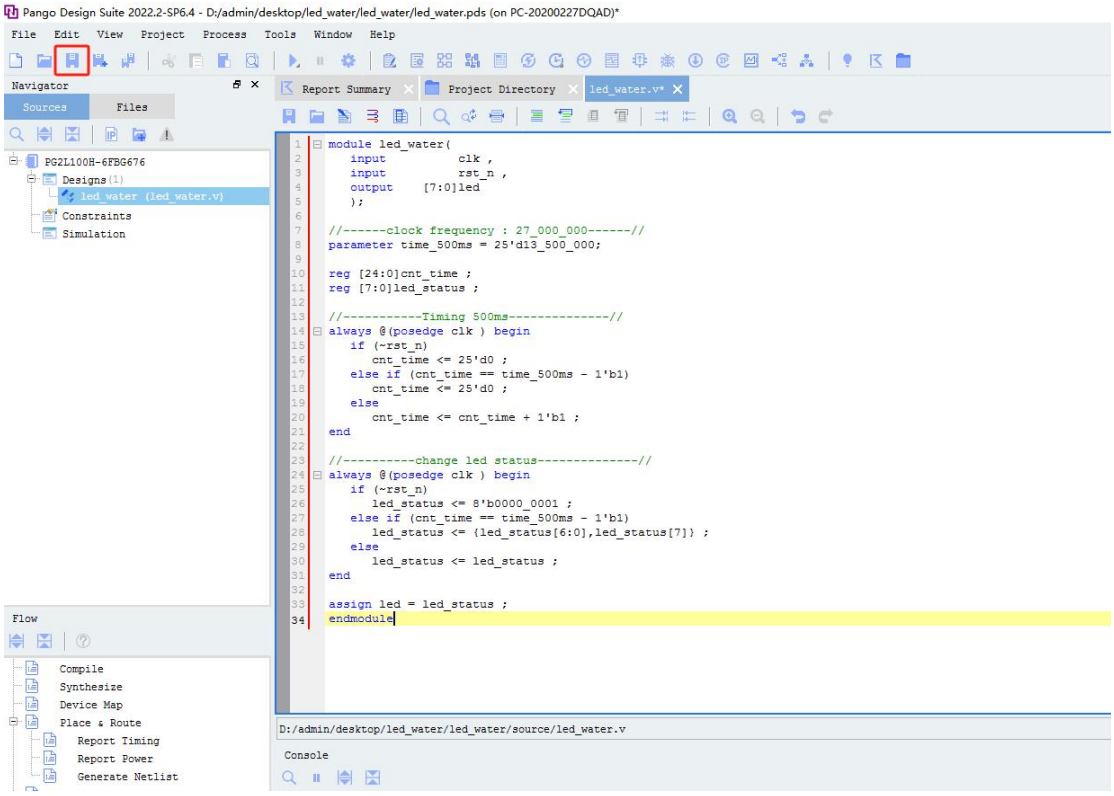


图 1.6-18

1.6.3.编译

可采用以下方式运行 Compile 流程:

(1)双击 Flow 中的 Compile 进行综合;

(2)右击 Compile 点击 Run 进行综合;

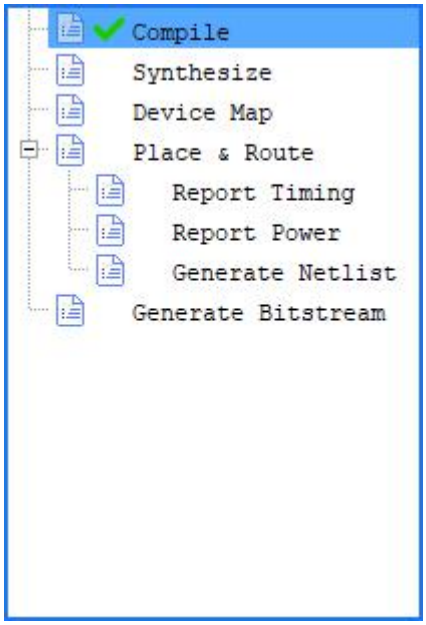


图 1.6-19

1.6.4.工程约束

点击 Tools 选择 User Constraint Editor(Timing and Logic)或者点击工具栏图标 ，User Constraint Editor(Timing and Logic) 选择 Pre Synthesize UCE, 如下图所示:

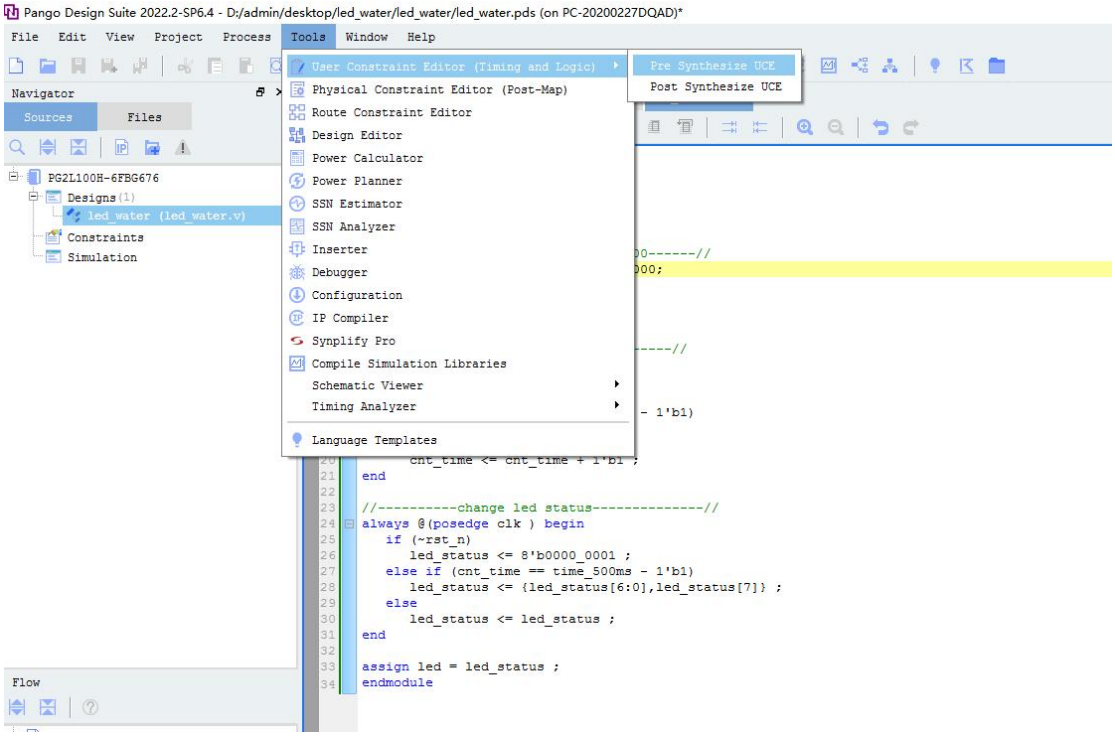


图 1.6-20

Tools 下的 User Constraint Editor(Timing and Logic)。

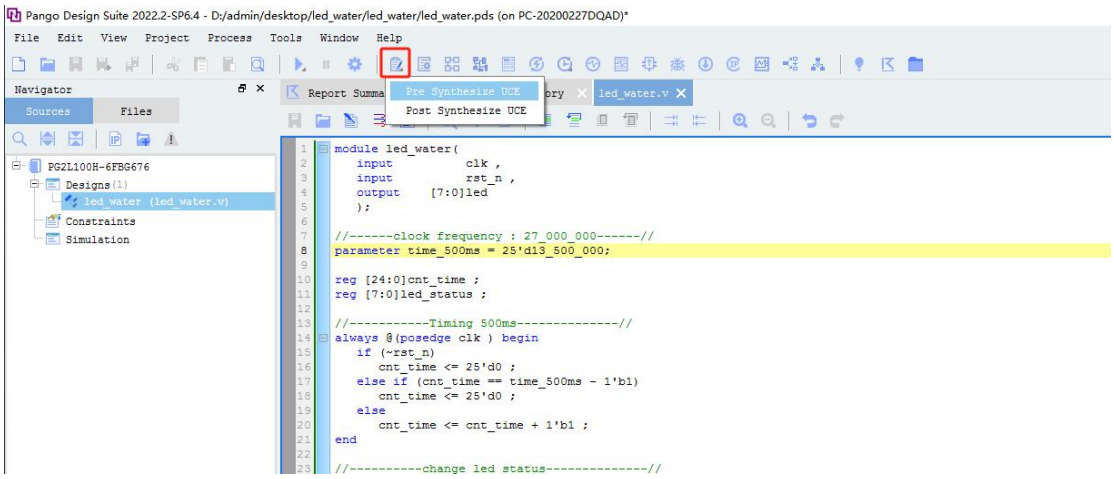


图 1.6-21

工具栏 User Constraint Editor(Timing and Logic)图标。

1.6.4.1.1. 时钟约束

打开 UCE 后,选择 Timing Constraints 后选择 Create Clock 添加基准时钟,基准时钟一般是通过输入 port 输入用户所使用的板上时钟。

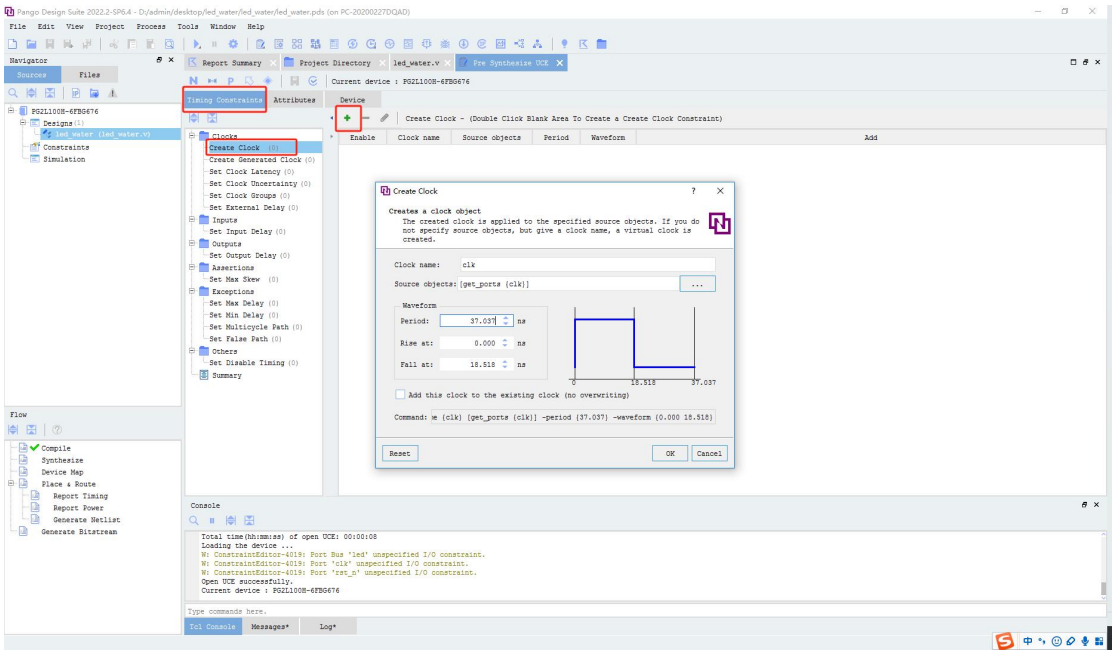


图 1.6-22

在弹窗中对时钟命名, 关联时钟管脚, 添加时钟参数, 点击 OK 会创建一条时钟约束, Reset 重置该页面。创建完成如下图所示:

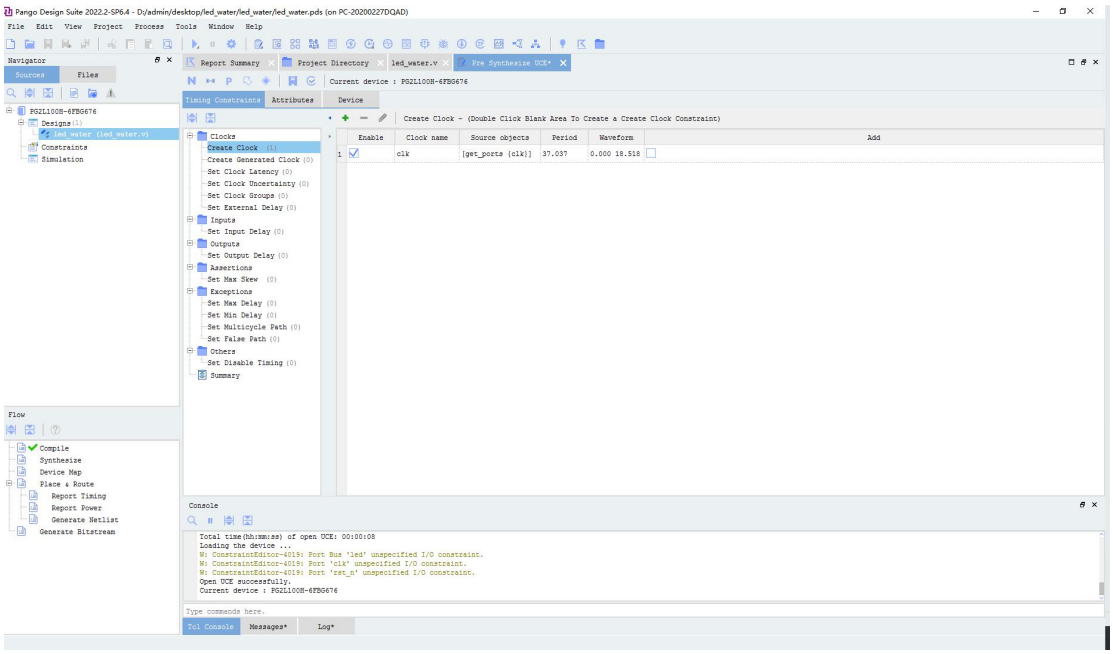


图 1.6-23

1.6.4.1.2. 物理约束

打开 UCE 后, 选择 Device 后选择 I/O, 根据原理图编辑 IO 的分配。

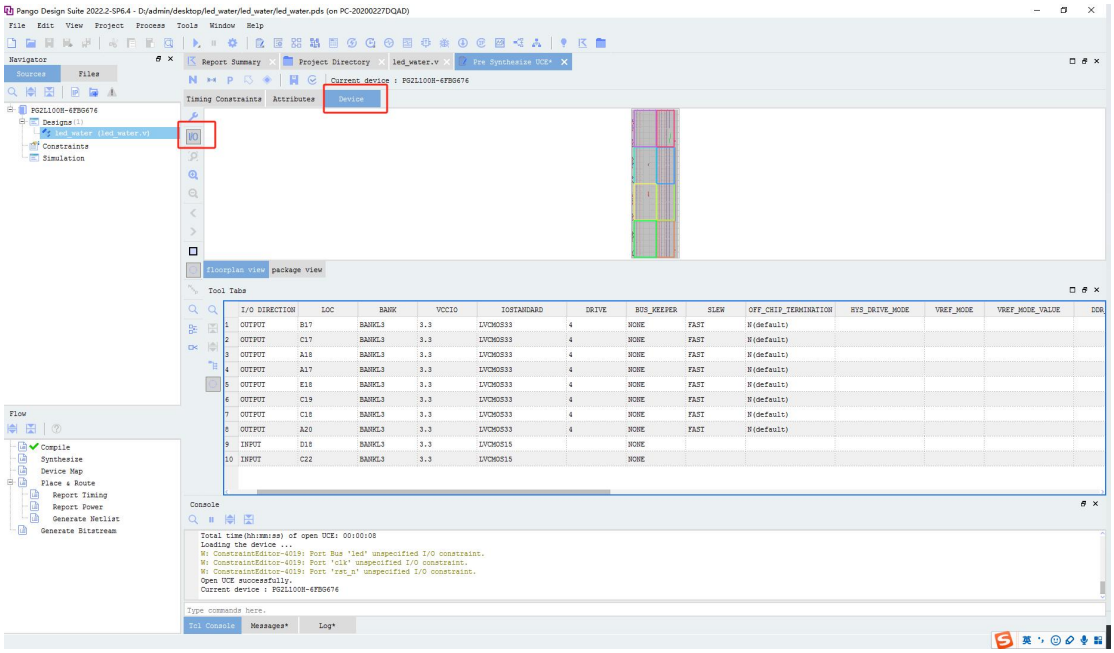


图 1.6-24

按照原理图编辑好 IO 分配后, 点击保存, 会生成.fdc 文件, 完成约束。

1.6.5.综合

运行 Synthesize 流程有以下四种方式可以实现:

(1)双击 Flow 中的 Synthesize 进行综合;

(2)右击 Synthesize 点击 Run 进行综合;

完成 Synthesize 操作后, 会看到下图所示:

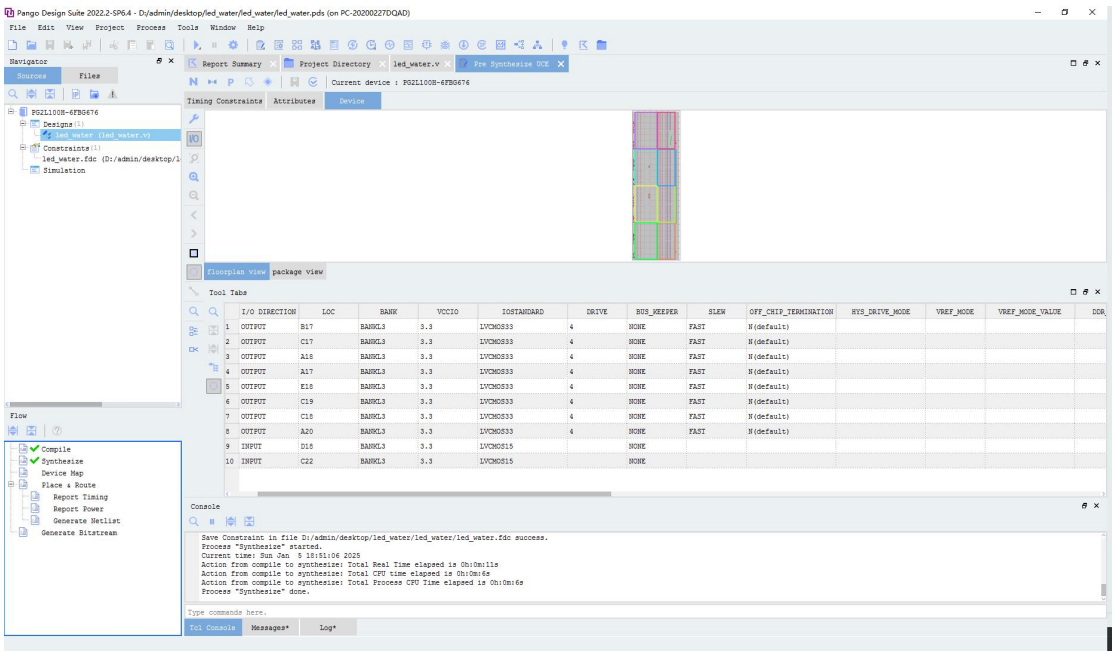


图 1.6-25

1.6.6.Device Map

Device Map 的主要作用是将设计映射到具体型号的子单元上（LUT、FF、Carry 等）。运行 Device Map 流程有以下方式可以实现：

（1）直接双击 Device Map；

（2）右击 Device Map 点击 Run；

完成 Device Map 操作后，会看到下图所示：

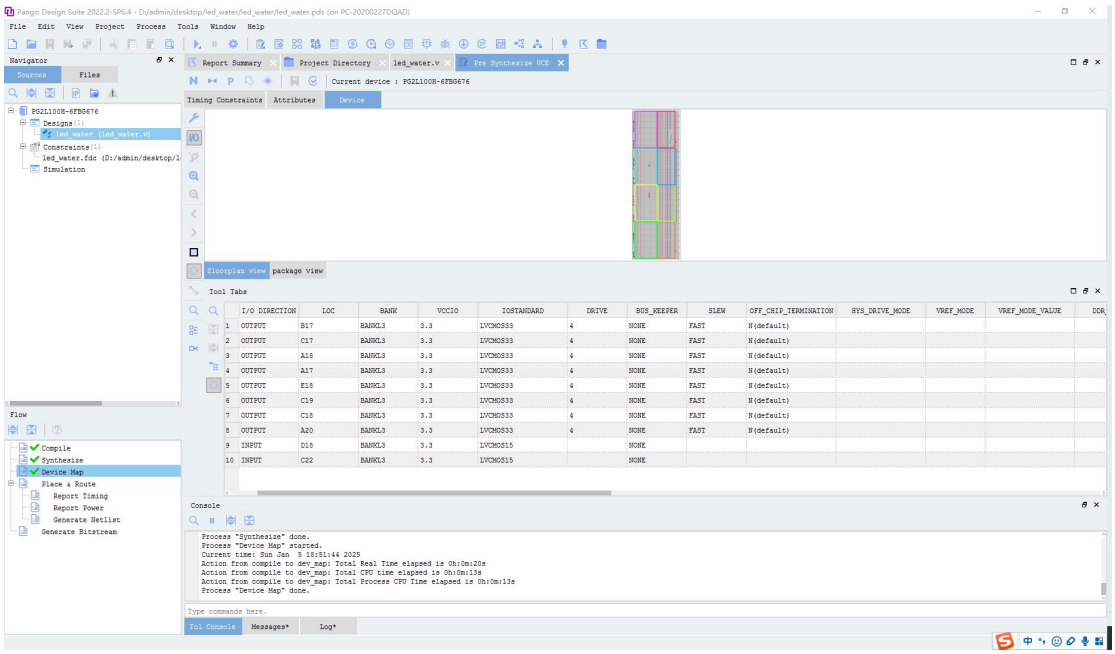


图 1.6-26

1.6.7.Place & Route

布局布线（Place & Route）根据用户约束和物理约束, 对设计模块进行实际的布局及布线。运行 Place & Route 流程有以下方式可以实现:

（1）直接双击 Place & Route;

（2）右击 Place & Route 点击 Run;

完成 Device Map 操作后, 会看到下图所示:

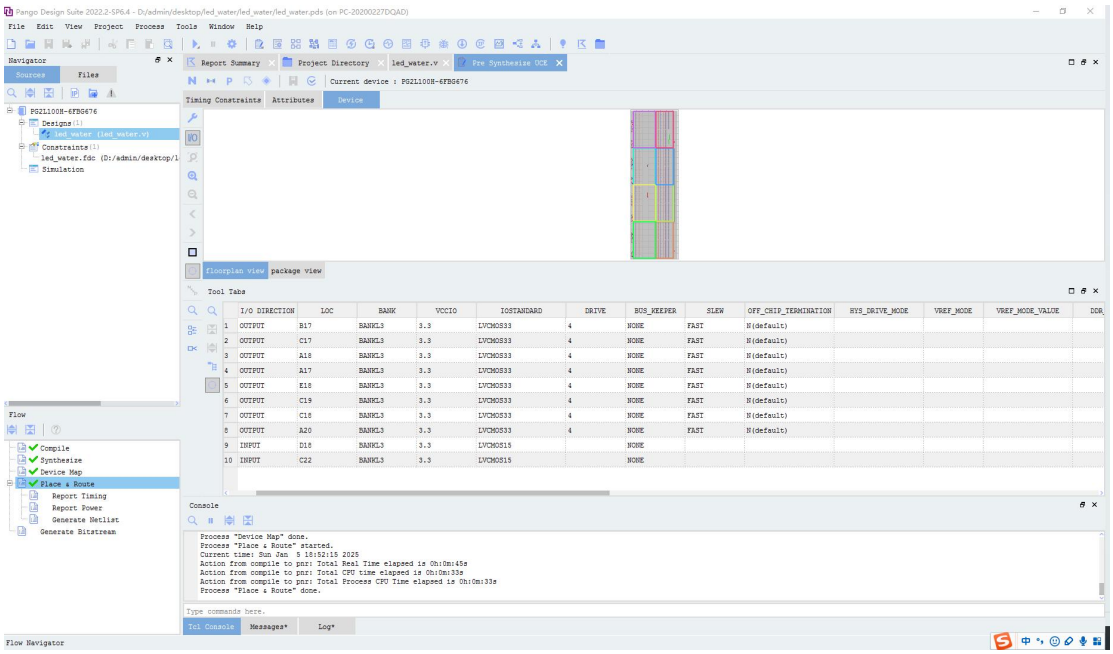


图 1.6-27

1.6.8.Generate Bitstream

Generate Bitstream 生成二进制位流文件。运行 Generate Bitstream 流程有以下方式可以实现:

（1）直接双击 Generate Bitstream;

（2）右击 Generate Bitstream 点击 Run;

完成以上操作, 将会产生位流文件。运行 Generate Bitstream, 可以看到界面如下图所示:

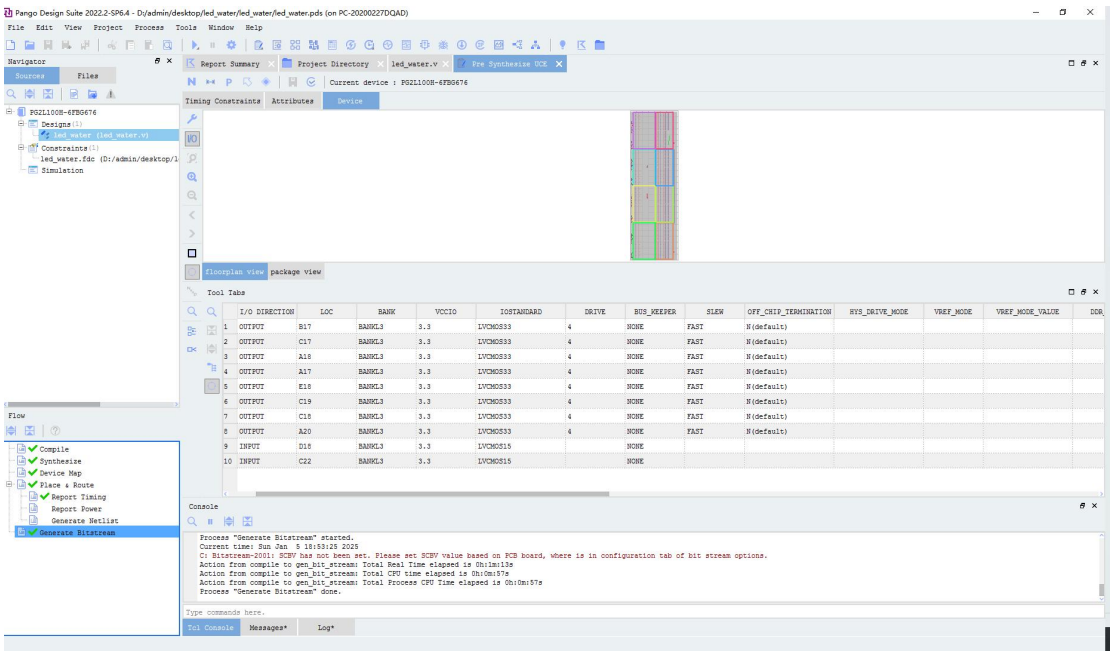


图 1.6-28

1.6.9.下载生成的位流文件

点击 Tools 选择 Configuration 或者点击工具栏图标 Configuration, 如下图所示:

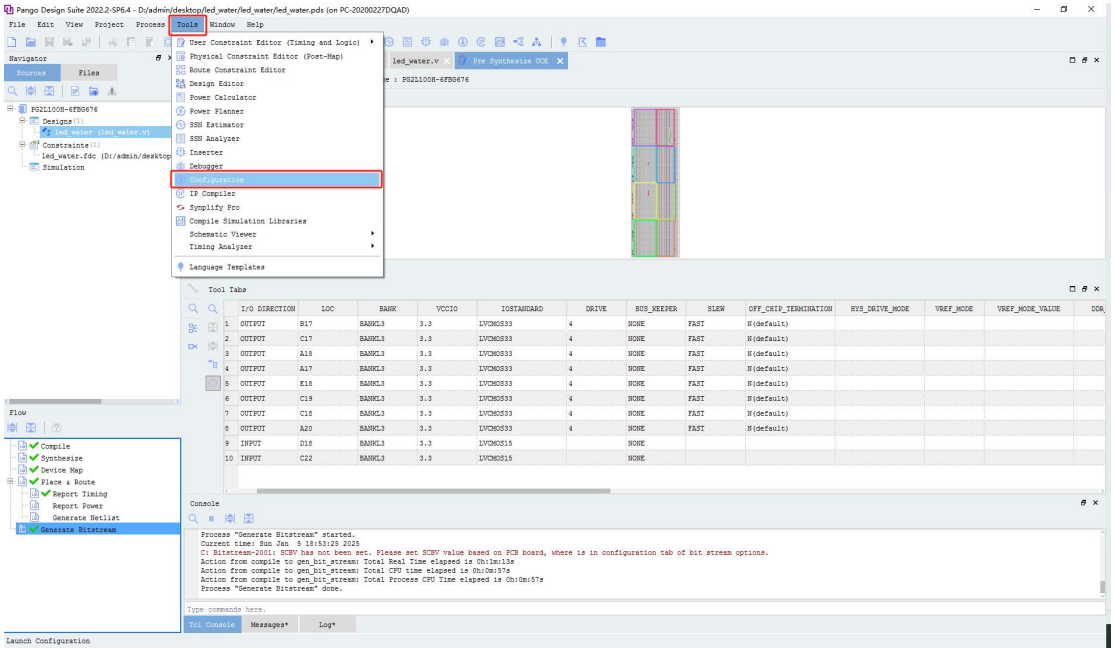


图 1.6-29

上图为 Tools 下的 Configuration 选项。

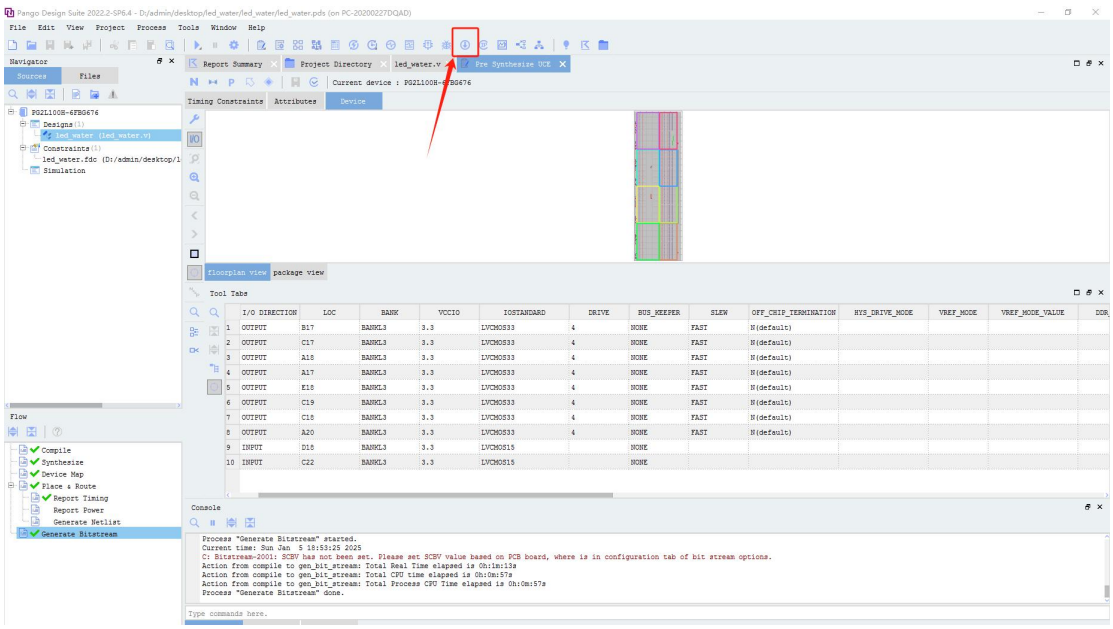


图 1.6-30

上图为工具栏 Configuration 图标。

打开 Configuration 后直接选择 Scan Device 直接进行扫描 Jtag 链操作, 初始化链成功, 会将链上扫描到的所有器件显示于工作区内, 并在器件属性窗口显示当前器件的器件信息, 并弹出对话显示能够为器件添加的配置文件;

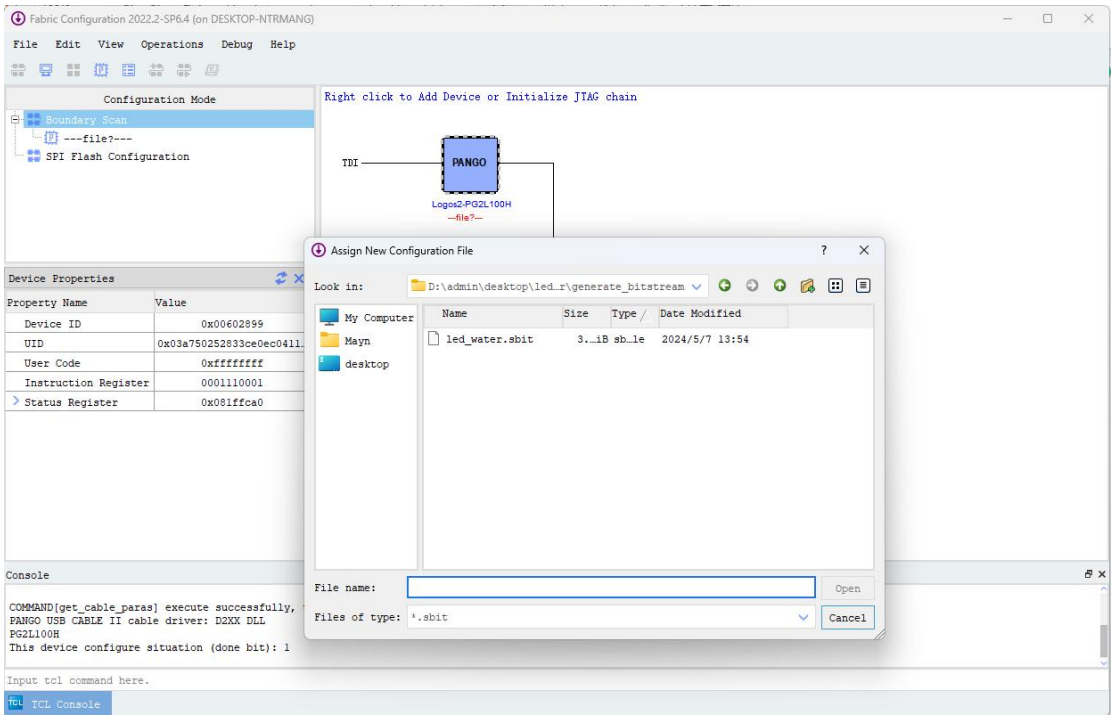


图 1.6-31

在对话框中选择位流文件, 添加该配置文件, 提示所载入文件的绝对路径并在信息栏中显示, 右键后点击 program 下载位流文件如下图所示:

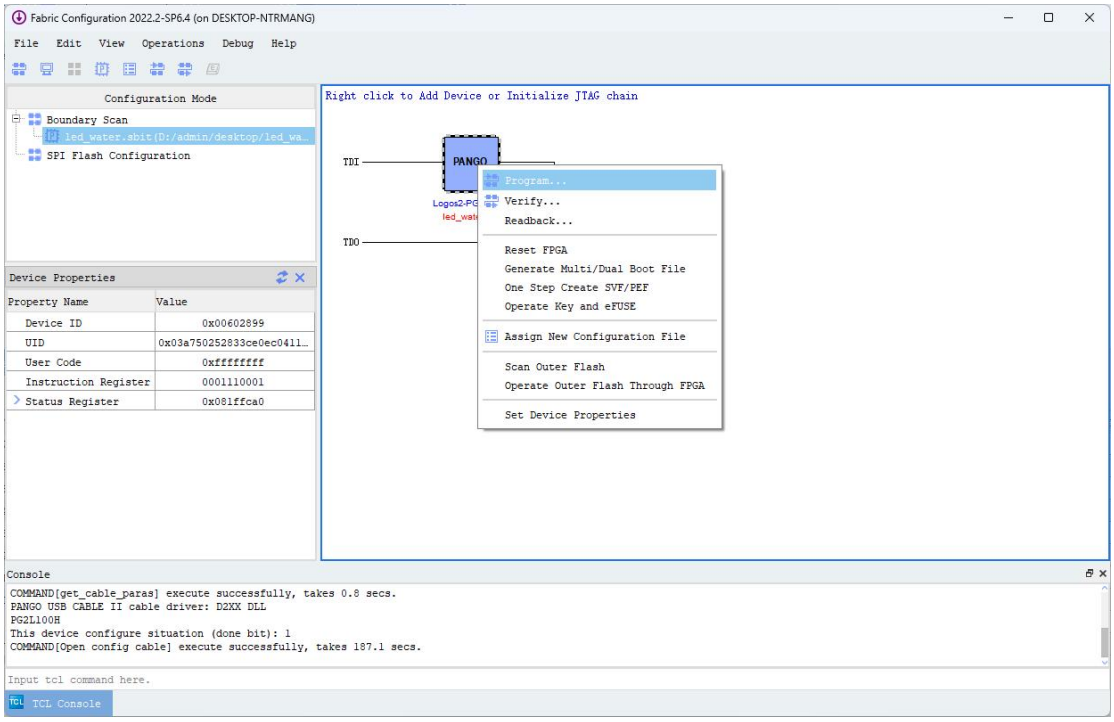


图 1.6-32

下载位流文件成功如下图所示:

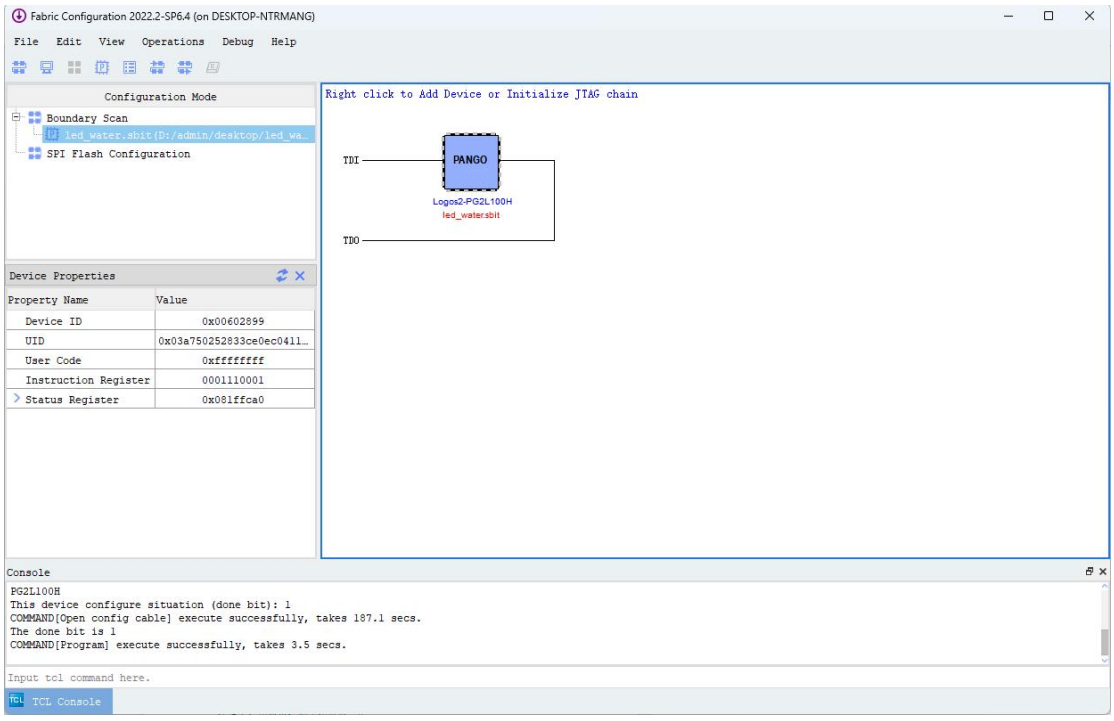


图 1.6-33

PG2L100H 板卡为 PG2L100H 的 FPGA 配置了一个外部 flash, 其中, 若需要将程序固化到板卡上需要将尾流文件转化为.sfc 文件。

首先点击 Configuration 页面的 Operations 选项的 Covert File 选项;

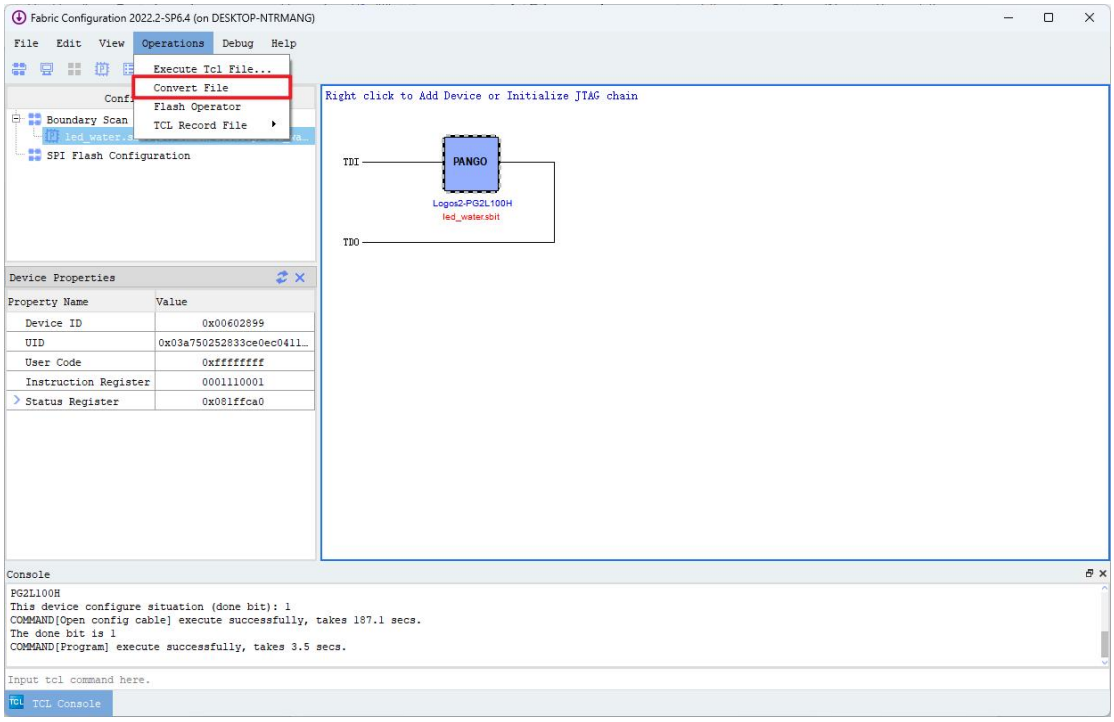


图 1.6-34

点击后会出现如下画面, 在 Generate Flash Programing File 页面选择对应的 Flash 器件的厂商名、型号、再在 BitStreamFile 位置选择位流文件的路径, 点击 OK。(若使用的 flash 器件不在可选的 flash 列表中, 需手动添加对应 flash 型号, 操作步骤请参考开发板下载与固化相说明);

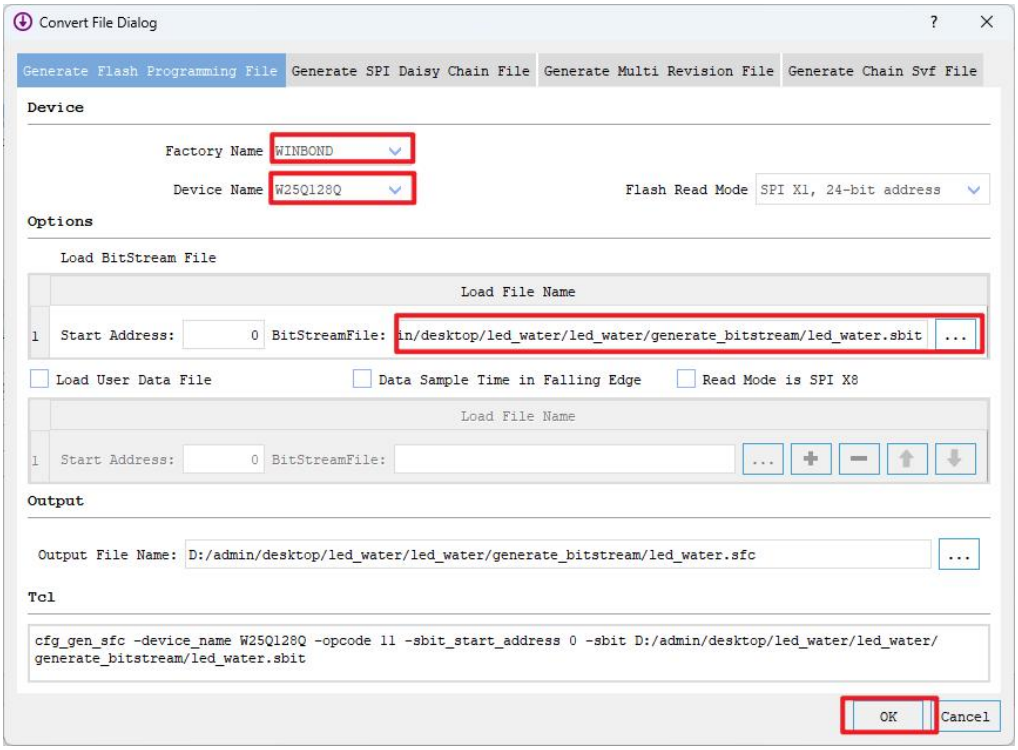


图 1.6-35

转化.sfc 文件成功后, 页面会如下图所示, 点击 OK;

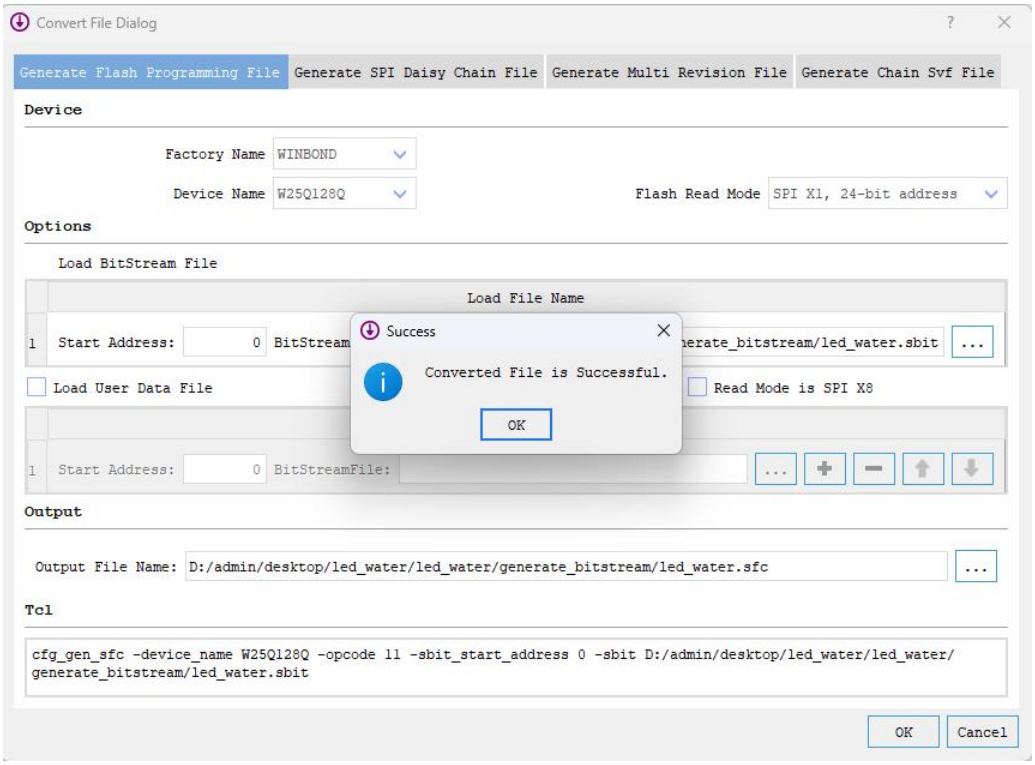


图 1.6-36

用户可通过右键下图位置, 点击 Scan Outer Flash;

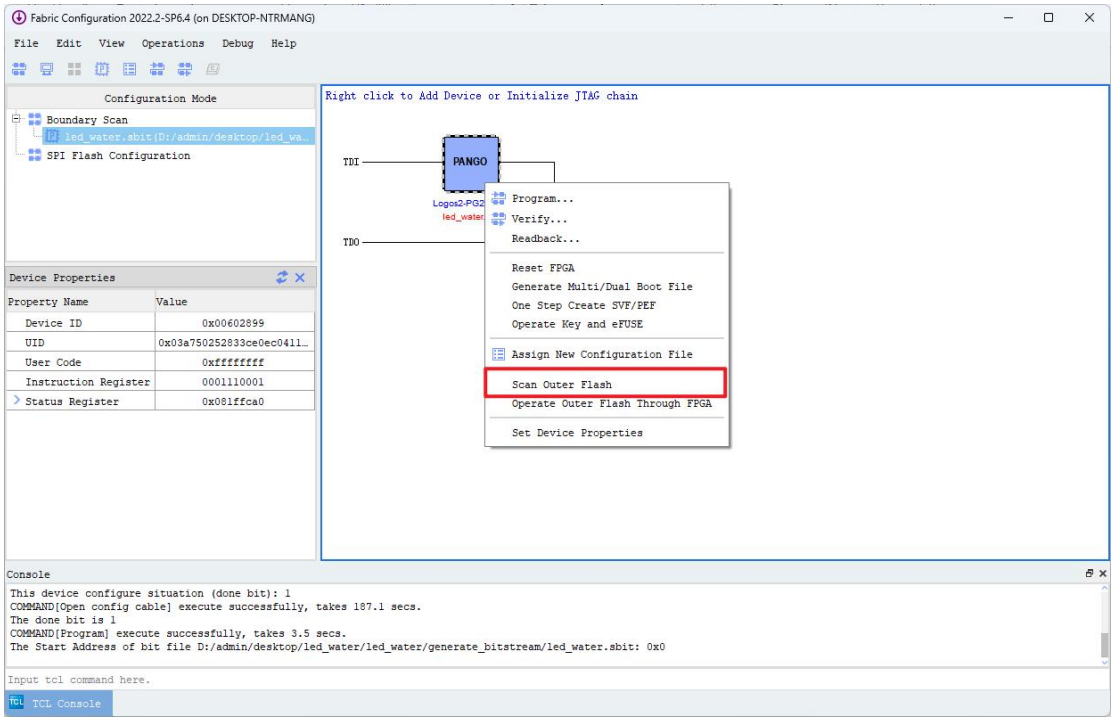


图 1.6-37

页面会显示板卡搭载的 Flash 的型号, 点击.sfc 文件, 点击 OPEN;

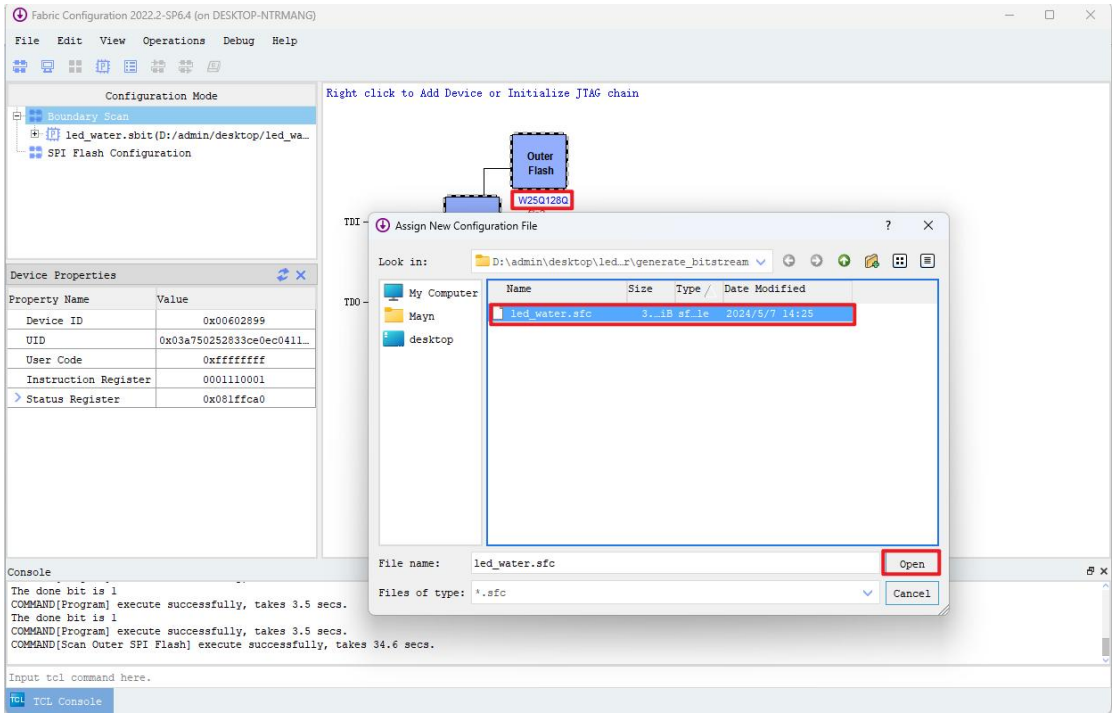


图 1.6-38

在下图位置点击鼠标右键后, 点击 Program;

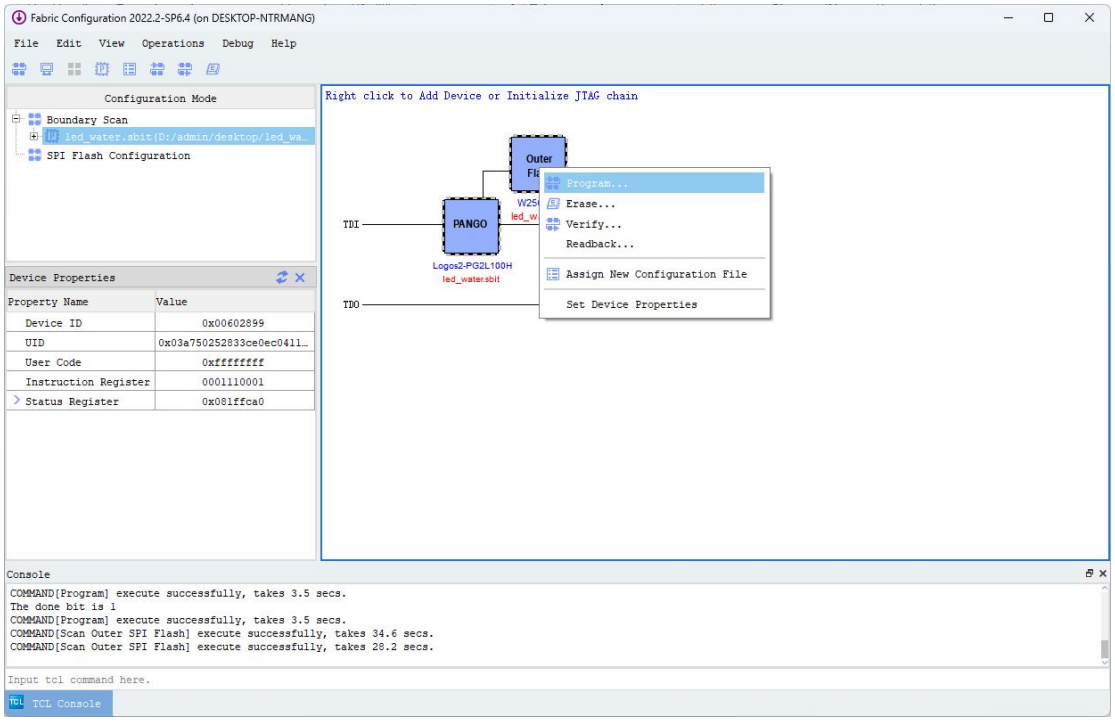


图 1.6-39

固化 Flash 成功如下图所示:

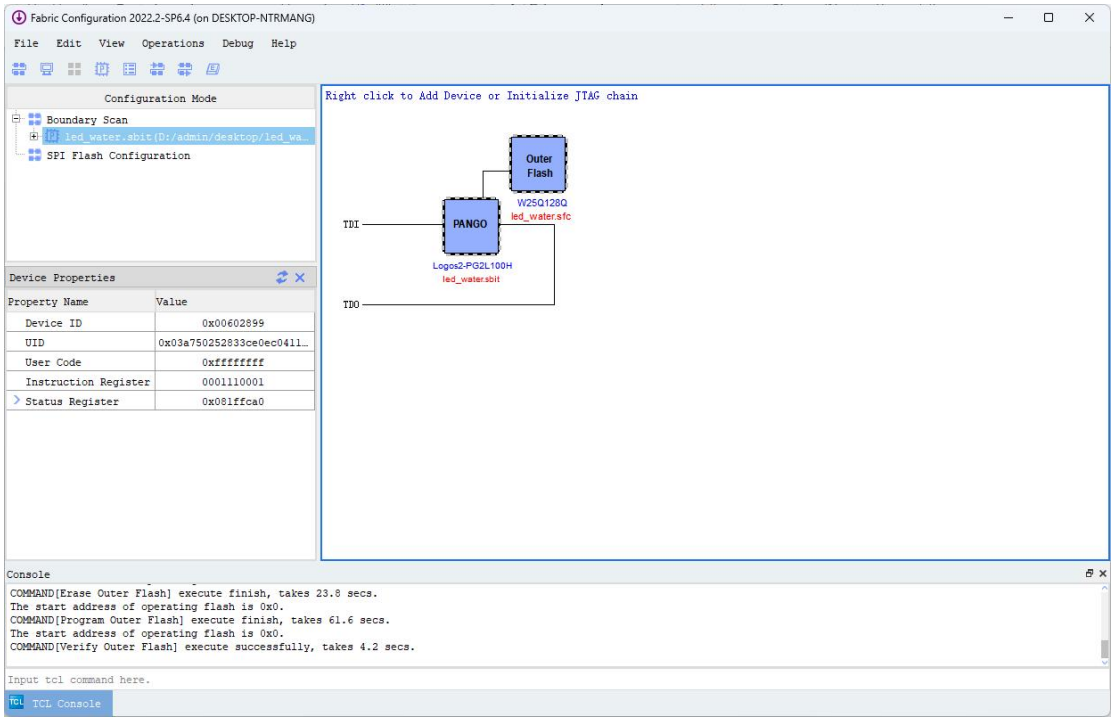

Fabric Configuration 2022.2-SP6.4 (on DESKTOP-NTRMANG)
File Edit View Operations Debug Help
Configuration Mode
Boundary Scan
led_water.sbit(D:/admin/desktop/led_wa...
SPI Flash Configuration
Right click to Add Device or Initialize JTAG chain
Outer Flash
W25Q128Q
led_water.sfc
TDI
PANGO
TDO
Logos2-PG2L100H
led_water.sbit
Device Properties
Property Name
Value
Device ID
0x00602899
UID
0x03a750252833ce0ec0411...
User Code
0xffffffff
Instruction Register
0001110001
Status Register
0x081ffca0
Console
COMMAND[Erase Outer Flash] execute finish, takes 23.8 secs.
The start address of operating flash is 0x0.
COMMAND[Program Outer Flash] execute finish, takes 61.6 secs.
The start address of operating flash is 0x0.
COMMAND[Verify Outer Flash] execute successfully, takes 4.2 secs.
Input tcl command here.
TCL Console

图 1.6-40

1.6.10. 上板验证

连接好电源和 jtag, 然后打开板卡上的电源;

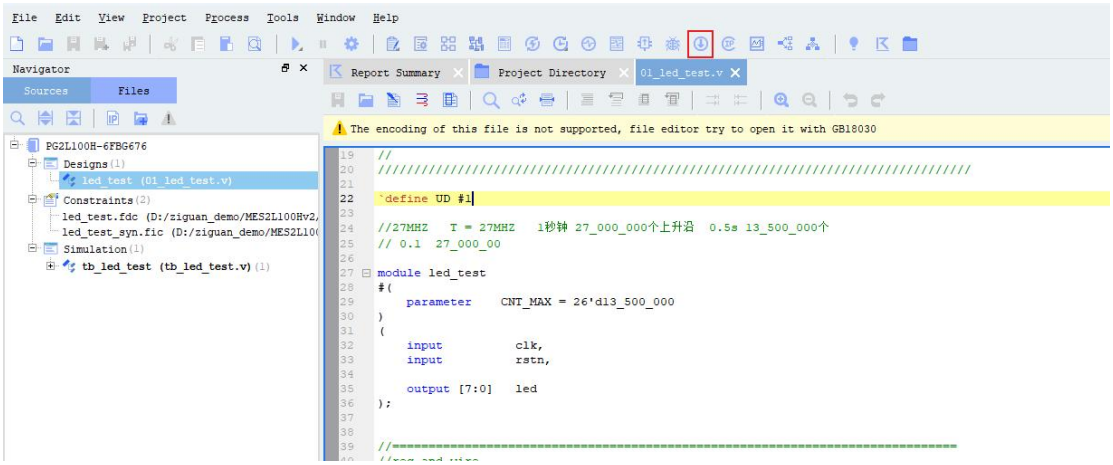


图 1.6-41

点击 PDS 软件上方的下载按钮, 红框所示部分:

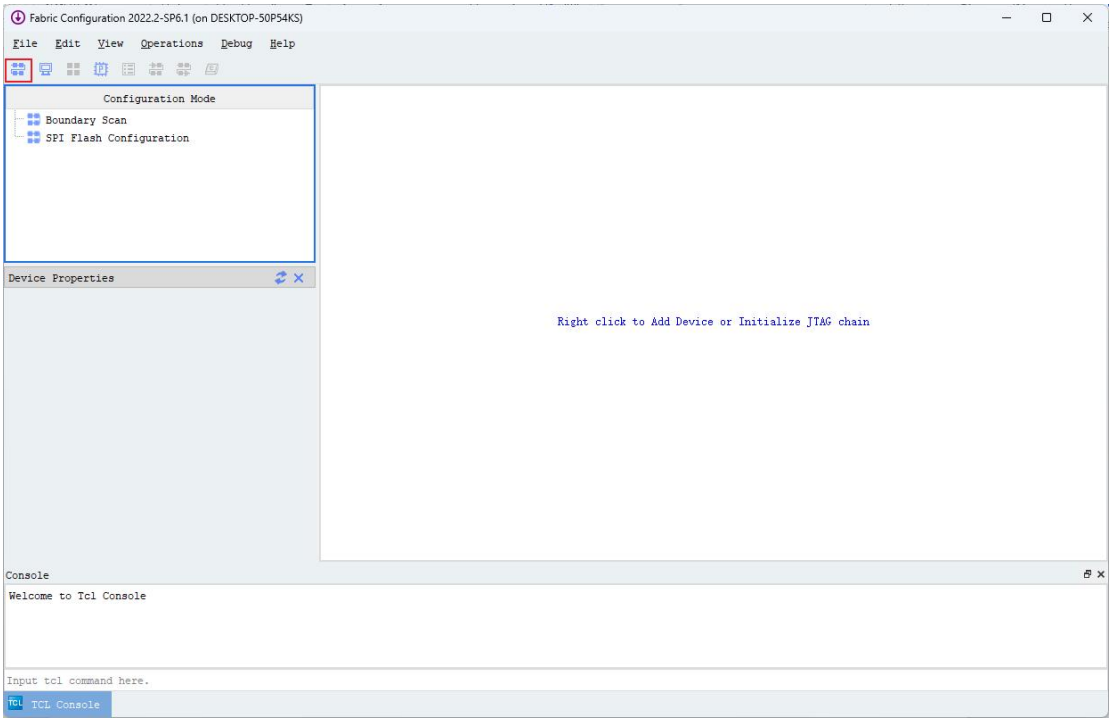


图 1.6-42

在弹出来的界面再点击如图红框部分:

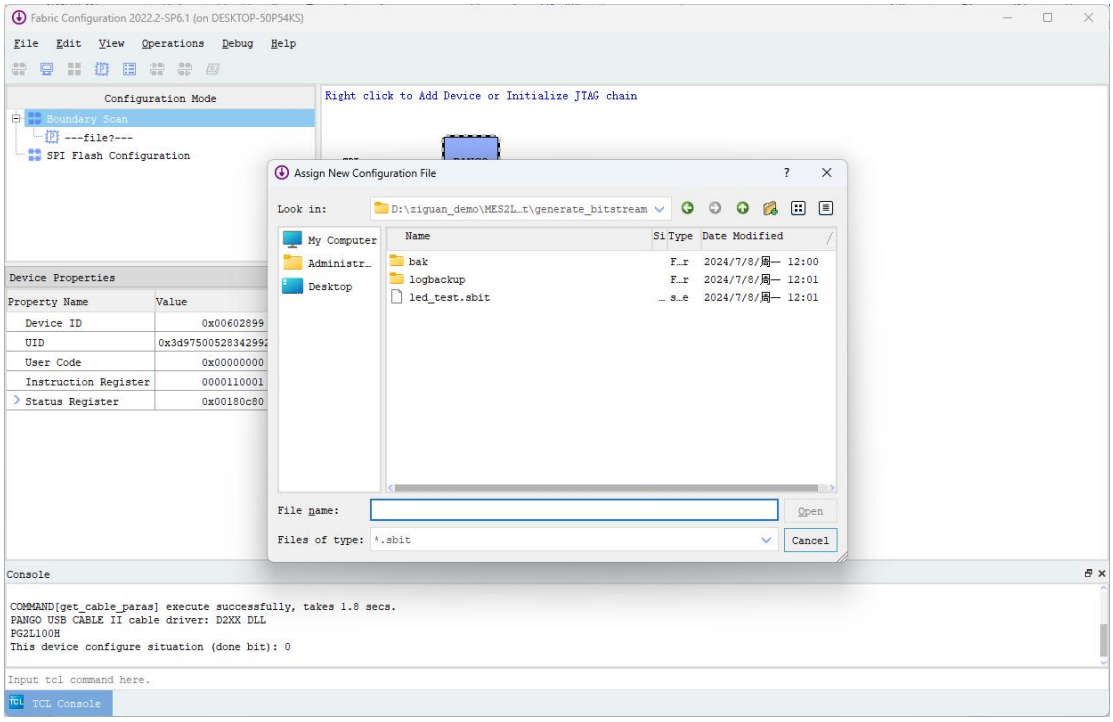


图 1.6-43

点击后会自动搜索设备，如果连线没问题的话可以看到自动弹出一个界面来选择 sbit。然后选择 led_test.sbit;

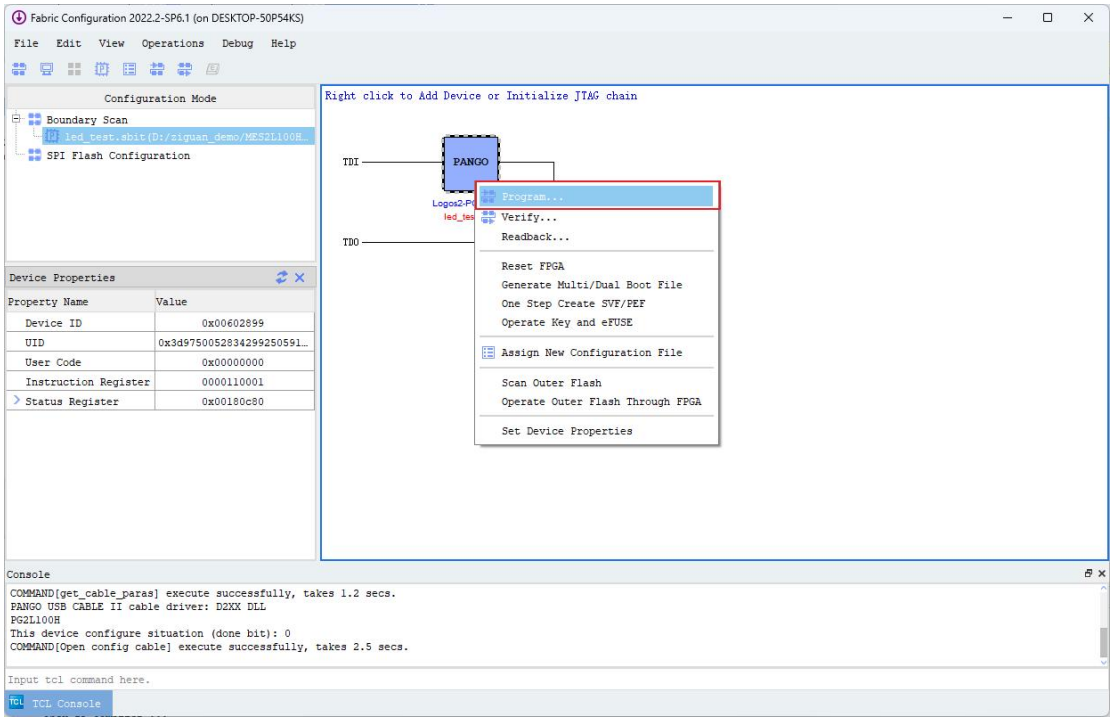


图 1.6-44

右键“芯片”，然后选择 program，等待烧录完成即可。接下来观察上板现象。

2. 基于 CLM 的单端口 RAM 读写

- **实验测试内容:**

测试模块名称: clm_single_ram_test

使用 GTP (GTP_RAM32X2SP)和 IP 核 (Distributed Single Port RAM)分别调用 CLM 资源实现单端口 RAM, 并对单端口 RAM 进行读写测试。

- 测试数据:

对单端口 RAM 进行写操作测试: 周期性的送入 00、01、10、11 四组数据, 送入 32 组数据后执行读操作。

对单端口 RAM 进行读操作测试: 读出 32 组数据后执行写操作。

- 实验结果验证:

对照对同一地址进行写操作写入的数据与进行读操作读出的数据是否一致。

2.1. 实验原理

logos2 系列的 CLM 分两种形态 CLMA、CLMS, 数量比约 3: 1 , CLMA 与 CLMS 均可实现逻辑、算数、移位寄存器以及 ROM 功能, 但仅有 CLMS 支持分布式 RAM 功能。

一般分布式 RAM 用于深度小于等于 64 的场景, 可以通过软件 Pango Design Suite 内嵌的 IP Compiler 工具或例化 GTP 原语生成。

2.1.1.Distributed Single Port RAM IP 调用

- 选择 IP

打开 IPC, 在主窗口中点击 File->Update IP 对话框, 添加对应版本的 IP 模型。选取 Module/Memory/Distributed RAM 目录下对应版本的 Distributed Single Port RAM, IP 选择路径下对应版本的 Distributed single Port RAM, IP 选择路径如图 2.1-1 所示:

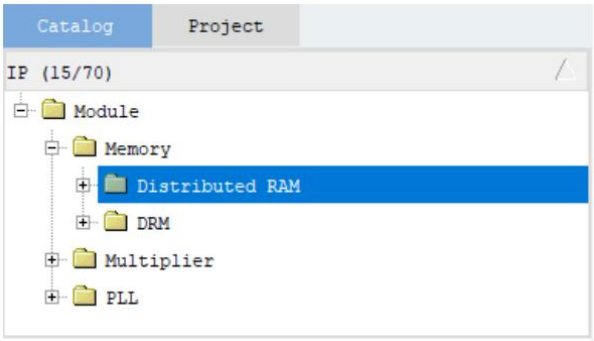


图 2.1-1 IP 选择路径

选择 IP 后, 需在右侧页面设置 Pathname 和 Instance Name 名称, 工程例化界面如图 2.1-2 所示:

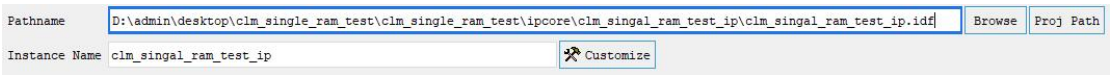


图 2.1-2 工程例化界面

➢ 配置 IP 参数

IP 选择完成后，点击<Customize>进入 IP 参数配置页面，详细参数说明请参考参数描述；

测试实验中配置的 IP 核参数如图 2.1-3 所示：

地址位宽：5；数据位宽：2；不使能输出寄存器，复位类型为异步；不使用初始化文件。

图 2.1-3 测试实验设置的 IP 核参数

参数描述如表 2.1-1 所示：

表 2.1-1 参数描述

参数/配置选项	参数说明	IP 配置界面默认值
Address Width	配置地址位宽，此参数合法范围为 4~10。	4
Data Width	配置数据位宽，此参数合法范围 1~256。	4
Enable Output Register	配置是否使用输出寄存。 勾选：使用输出寄存器； 不勾选：不使用输出寄存。	不勾选
Reset Type	配置寄存器输出时所使用的复位方式。 ASYNC：异步复位； SYNC：同步复位。	ASYNC
Init Enable	配置是否使能对当前 RAM 进行初始化。 勾选：初始化使能； 不勾选：初始化不使能。	不勾选

Init File	指定初始化文件路径, 仅在选项 “Init Enable ” 勾选时有效。 若不指定, 则生成初始值为全“0 ”的初始化文件.v 文件。	NONE
Initial Data Format Type	配置初始化文件数据格式。 BIN: 二进制; HEX: 十六进制。	BIN

- 生成 IP

参数配置完成后, 点击左上角<Generate>生成 IP。

2.1.2.GTP_RAM32X2SP 原语调用

CLMS 可以配置成不同大小的 RAM 块,使用 GTP 原语例化可以调用 CLM 实现分布式 RAM,其中 CLM 可以实现两种 RAM, 一种为 SP 单端口 RAM, 读写地址共用, 另一种为(S)DP 简单双端口 RAM, 读写地址不共用。

测试实验使用 PG2L100H 板卡, 调用深度为 32, 数据位宽为 2, 单端口 RAM 模块原语: GTP_RAM32X2SP, 此原语支持 Logos2 系列。使用 GTP 原语调用分布式单端口 RAM 的过程如下所示:

- 选择原语

打开 Tools->Language Templates, 在 Language Templates 界面下, 选取 Verilog/GTP Templates 目录下的 GTP_RAM32X2SP 进行例化, 如图 2.1-4 所示。

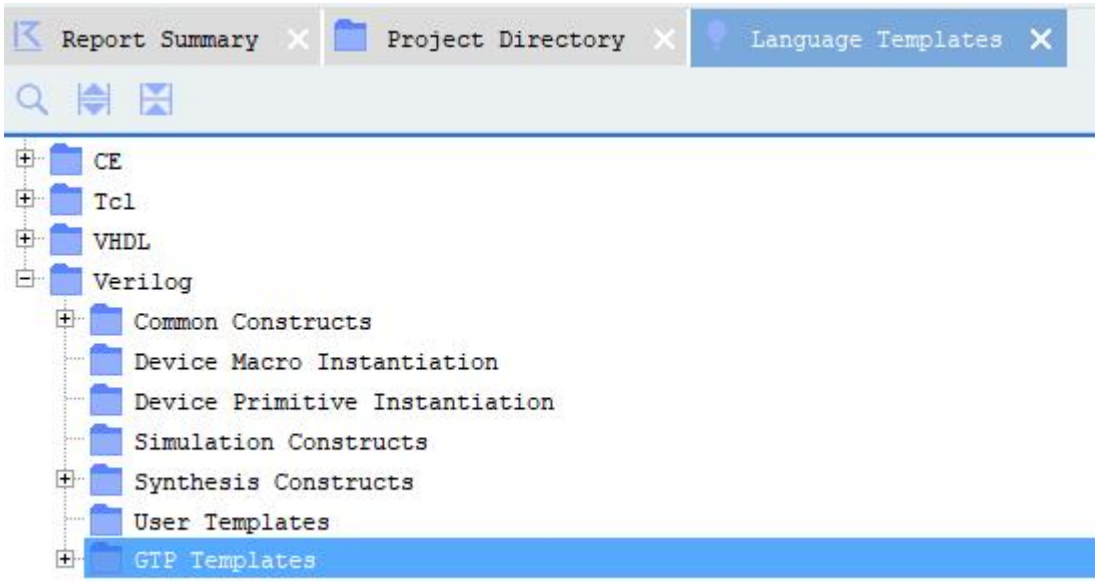


图 2.1-4 GTP 路径选择

2.2. 接口列表

- IP 接口列表

接口框图如下图:

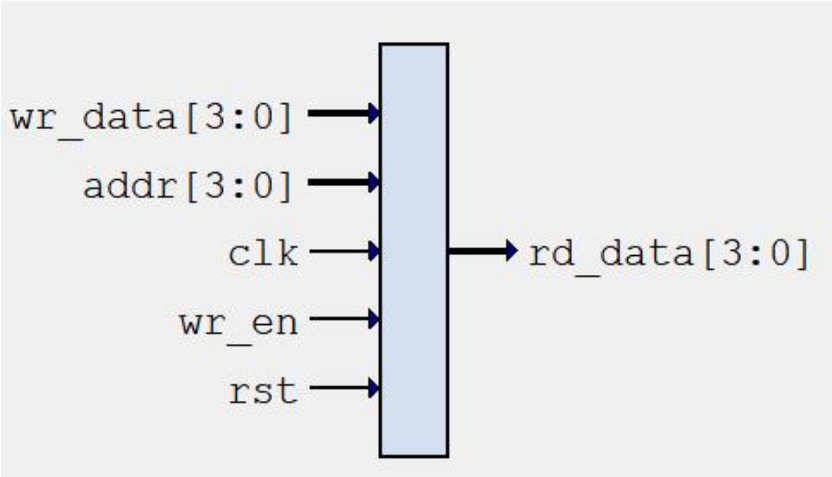


图 2.2-1 分布式单端口 RAM IP 核接口框图

接口列表如下表所示:

表 2.2-1 分布式单端口 RAM IP 核接口描述

端口	I/O	描述
wr_data	I	写数据信号。
addr	I	地址信号，读写共用一个地址信号。
clk	I	时钟信号。
wr_en	I	写使能信号, 高有效。
rst	I	复位信号，高有效。
rd_data	O	读数据信号。

- 原语接口列表

GTP_RAM32X2SP 是一个深度为 32, 数据位宽为 2 的单端口 RAM, 可以存储 64bit 的数据, 因此地址位宽为 5, 数据位宽为 2, WE 为写使能。

功能框图如图 2.2-2 所示:

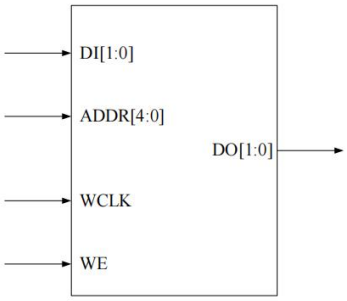


图 2.2-2 GTP_RAM32X2SP 功能框图

端口描述描述如下:

表 2.2-2 GTP_RAM32X2SP 端口功能描述

端口名称	方向	功能描述
DI[1:0]	输入	数据输入端口
ADDR	输入	读写地址
WCLK	输入	写时钟
WE	输入	写使能
DO[1:0]	输出	数据输出端口

参数描述如下:

表 2.2-3 GTP_RAM32X2SP 参数描述

参数名称	设置值	默认值	功能描述
INIT_0	32'h0~32'hffffffff	32'h0	Memory 初始化配置参数
INIT_1	32'h0~32'hffffffff	32'h0	Memory 初始化配置参数

2.3. 工程说明

2.3.1.测试参数说明

- Distributed Single RAM IP 核设置参数如下:
  - 时钟: 27MHz 。
  - 地址位宽: 5 (RAM 的适度为 32)。
  - 数据位宽: 2。
  - 输出寄存器: 不使用。

- 复位: 同步复位。
- 初始化文件: 不使用。

图 2.3-1 测试中有关分布式单端口 RAM 的 IP 核设置

- IP 核使用时钟: 27MHz。
- GTP 参数设置如下所示:
  - INIT_0 设置为全 0 (32 位)
  - INIT_1 设置位全 0 (32 位)
- GTP 使用时钟: 27MHz。
- 代码设置测试数据如下:
  - 读写间隔: 32 个时钟周期读, 32 个时钟周期写。
  - 测试数据: 在 RAM 进行写操作时, 周期性的写入 00、01、10、11 四个数据。
  - 时钟: 27MHz。

(分布式单端口 RAM IP 核的复位信号高电平有效)

- debug 核使用时钟:
  - 测试结果使用 debug 核观察,debug 工作时钟选择使用 108MHz,该时钟使用 PLL 产生。
- 该测试工程 debug 核采样深度设置为 2048。

2.3.2.使用 debug 观察测试结果

使用 debug 观察 IP_wr_en、IP_addr、IP_wr_data、IP_rd_data、GTP_we、GTP_addr、GTP_wr_data、GTP_rd_data 八个信号查看测试结果。该测试工程 debug 核采样深度设置为 2048。

Debug 使用请参考 PDS 软件目录下\doc\ Fabric_Debugger_User_Guide.pdf

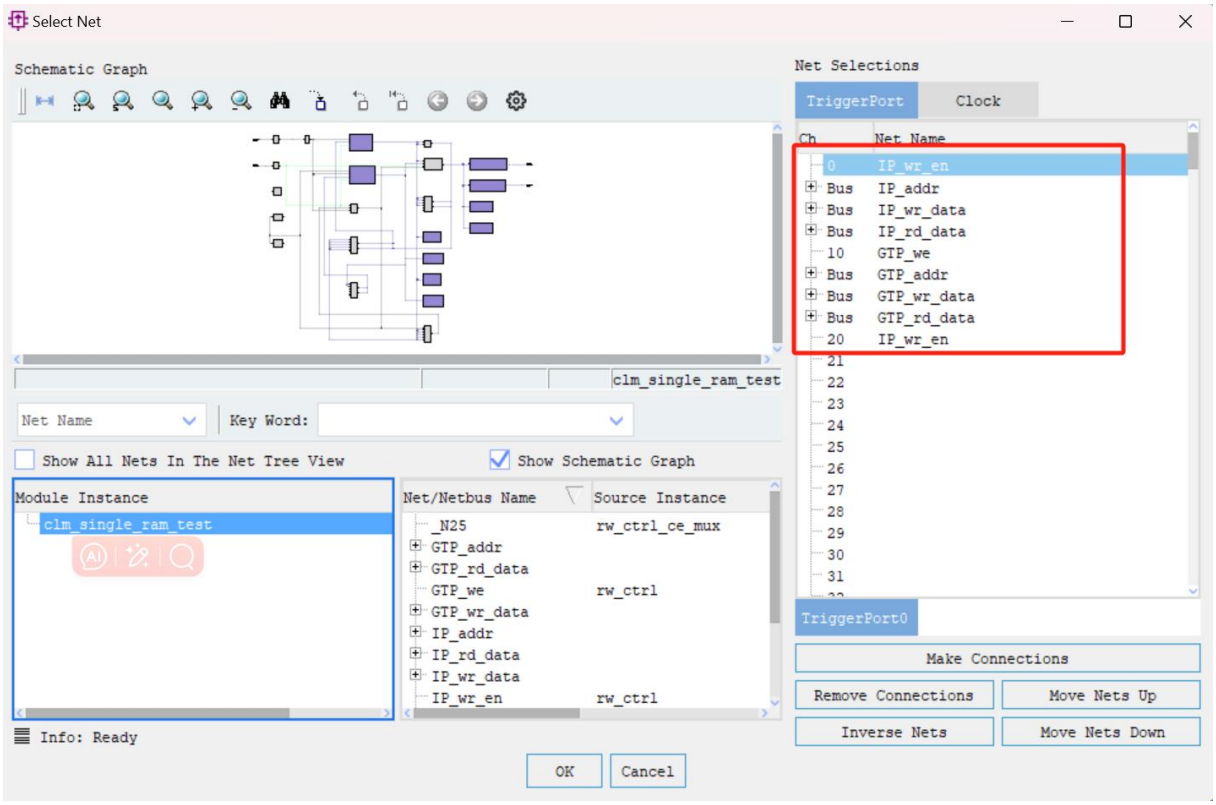


图 2.3-2 Select Net 配置

测试结果分析:

使用分布式单端口 RAM 进行写操作时, 写使能信号拉高, 将数据 00 写入地址 0, 将数据 01 写入地址 1, 将数据 10 写入地址 2, 将数据 11 写入地址 3, 将数据 00 写入地址 4, 依此规律写入。

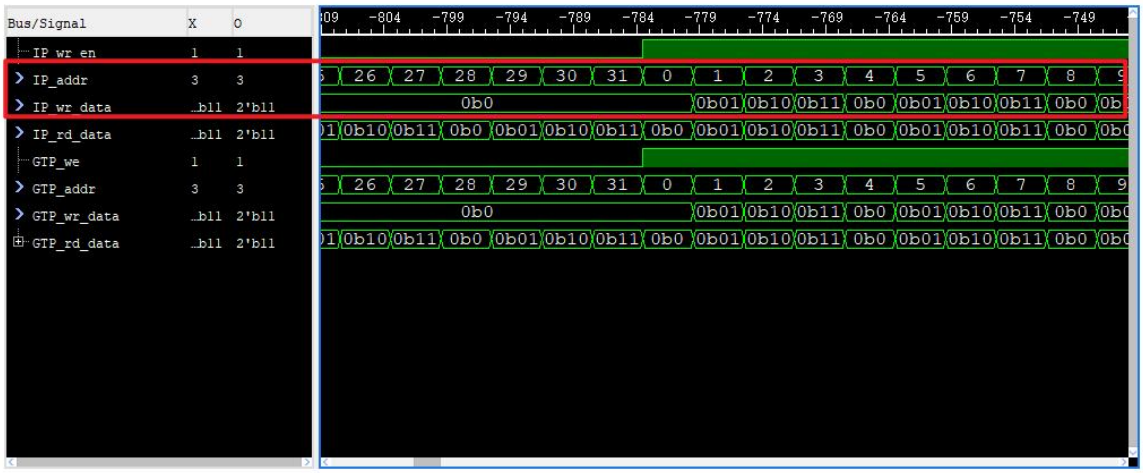


图 2.3-3 Distributed Single Port RAM 写操作测试（IP）

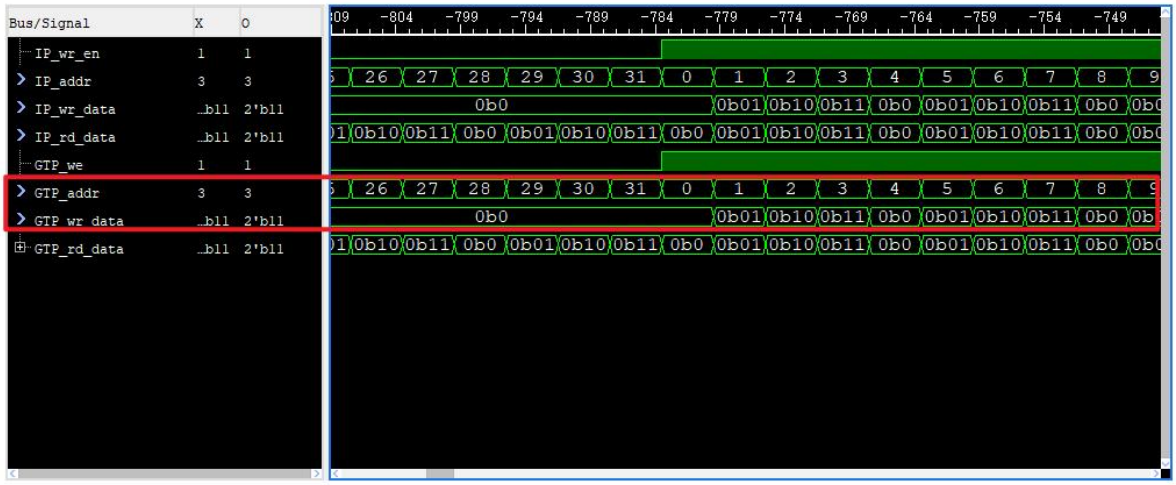


图 2.3-4 GTP_RAM32X2SP 写操作测试（GTP）

使用分布式单端口 RAM 进行读操作时，写使能拉低，地址 0 读出数据为 00，地址 1 读出的数据为 01，地址 2 读出的数据为 10，地址 3 读出的数据为 11，地址 4 读出的数据为 00 ；

图 2.3-5 Distributed Single Port RAM 读操作测试（IP）

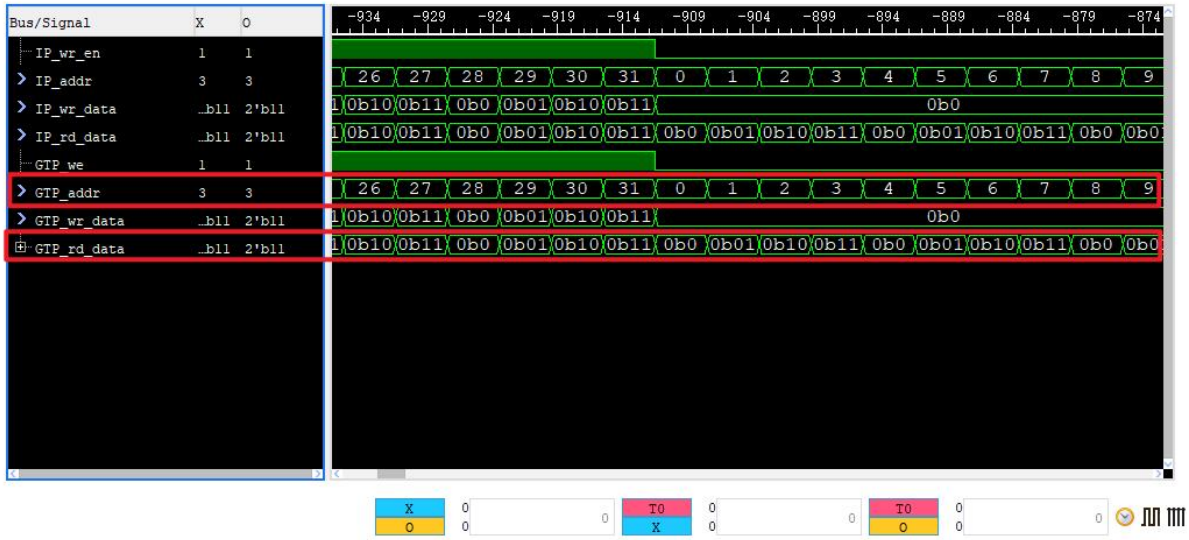


图 2.3-6 GTP_RAM32X2SP 读操作测试（GTP）

3. 基于 DRAM 的单端口 RAM 读写

- **测试实验内容:**

测试模块名称: dram_single_ram_test

使用 GTP（GTP_DRM18K_E1）和 IP（DRM Based Single Port RAM）核分别调用 DRAM 资源实现单端口 RAM, 并对单端口 RAM 进行读写测试。

- 测试数据:

对单端口 RAM 进行写操作测试: 周期性的送入 00、01、10、11 四组数据, 送入 32 组数据后执行读操作。

对单端口 RAM 进行读操作测试: 读出 32 组数据后执行写操作。

- 实验结果验证:

对照对同一地址进行写操作写入的数据与进行读操作读出的数据是否一致。

3.1. 实验原理

Logos2 系列 FPGA 的 DRAM 存储单元为 36kbit, 每个 DRAM 都可被独立的配置为 2 个 18kbit 或 1 个 36kbit, 每个 DRAM 都支持 DP 双端口 RAM、SP 单端口 RAM、SDP 简单双端口 RAM、ROM、FIFO 模式。可通过 IP 核或原语调用 DRAM。

DRM Based Single Port RAM(IP)与 GTP_DRM18K_E1(GTP)均具有三种写模式:Normal Write、Transparent Write、Read-before-Write、支持使用二进制或十六进制初始化文件对 RAM 进行初始化。

- 在 Normal Write 模式下进行写操作时, rd_data 不会输出数据。
- 在 Transparent Write 模式下进行写操作时, rd_data 会写入的数据。
- 在 Read-before-Write 模式下进行写操作时, rd_data 会输出写入数据对应地址在写入之前的数据。

3.1.1.DRM Based Single Port RAM IP 调用

- 选择 IP

打开 IPC, 在主窗口中点击 File->Update IP 对话框, 添加对应版本的 IP 模型。选取 Module/Memory/DRAM 目录下对应版本的 DRAM Based Single Port RAM, IP 选择路径下对应版本的 DRAM Based Single Port RAM, IP 选择路径如下图所示:

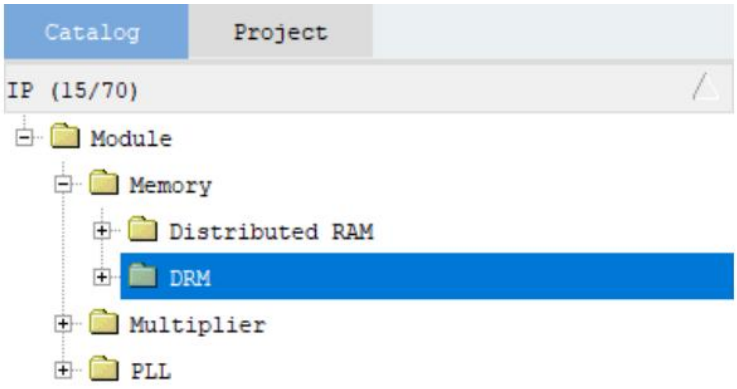


图 3.1-1 IP 选择路径

选择 IP 后, 需在右侧页面设置 Pathname 和 Instance Name 名称, 工程例化界面如下图所示:

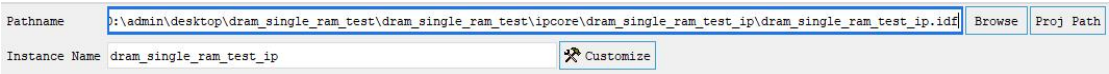


图 3.1-2 工程例化界面

- 配置参数

IP 选择完成后, 点击<Customize>进入 IP 参数配置页面, 进行参数的配置。

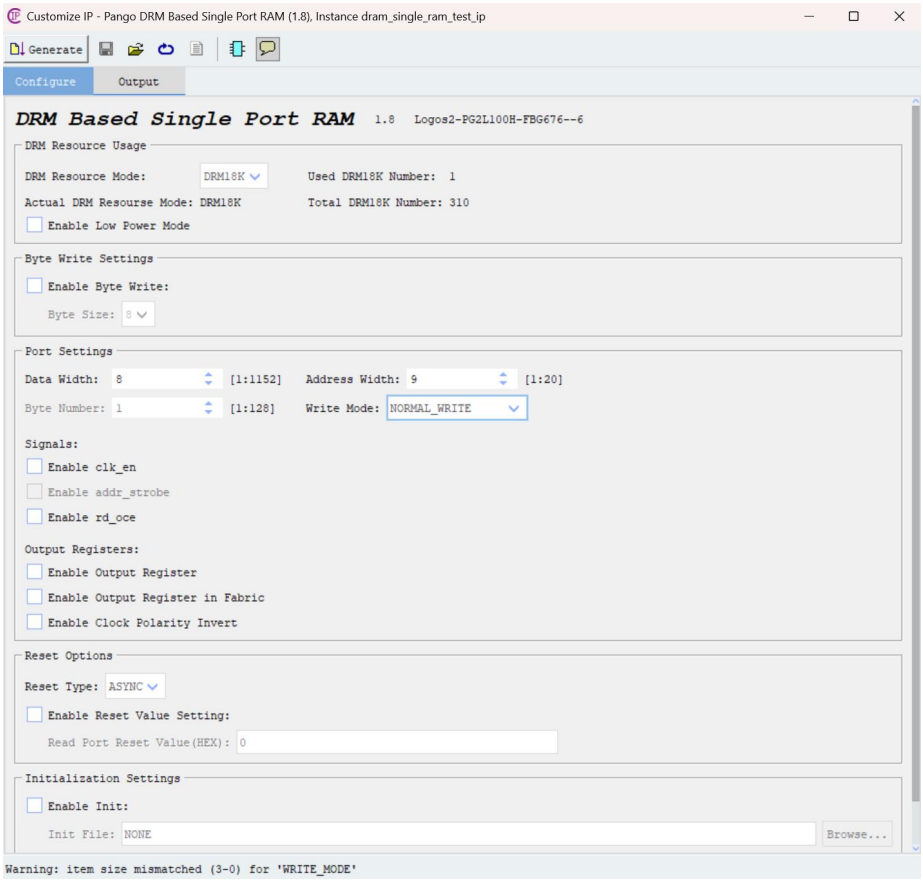


图 3.1-3 IP 参数配置

- 生成 IP

参数配置完成后, 点击左上角<Generate>生成 IP。

3.1.2.GTP_DRM18K_E1 原语调用

- 选择原语

打开 Tools->Language Templates, 在 Language Templates 界面下, 选取 Verilog/GTP Templates 目录下的 GTP_DRM18K_E1;

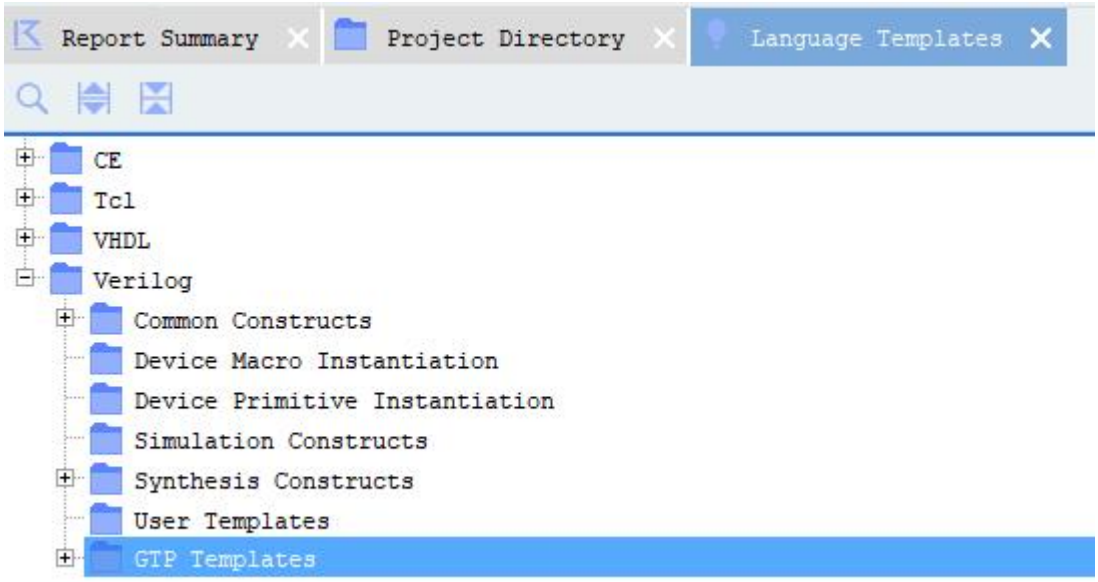


图 3.1-4 路径选择

- 例化原语

在右侧页面复制 Instantiation Template 例化到工程中。

```
// Module Instantiation Template
// Insert the following codes into your Verilog file.
//   * Change <InstanceName> to your own instance name.
//   * Fill in your own signal names in the port connections.

GTP_DRM18K_E1 #(
    .GRS_EN("TRUE"),
    .DATA_WIDTH_A(18),
    .DATA_WIDTH_B(18),
    .DOA_REG(0),
    .DOB_REG(0),
    .DOA_REG_CLKINV(0),
    .DOB_REG_CLKINV(0),
    .RSTA_VAL('b000000000000000000),
    .RSTB_VAL('b000000000000000000),
    .RST_TYPE("SYNC"),
    .RAM_MODE("TRUE_DUAL_PORT"),
    .WRITE_MODE_A("NORMAL_WRITE"),
    .WRITE_MODE_B("NORMAL_WRITE"),
    .INIT_00('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_01('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_02('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_03('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_04('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_05('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_06('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_07('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_08('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_09('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_0A('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_0B('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_0C('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_0D('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_0E('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_0F('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_10('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_11('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_12('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_13('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_14('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_15('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_16('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_17('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_18('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_19('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_1A('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_1B('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    .INIT_1C('b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
```

图 3.1-5 GTP 例化模板

3.2. 接口列表

3.2.1.IP 接口列表

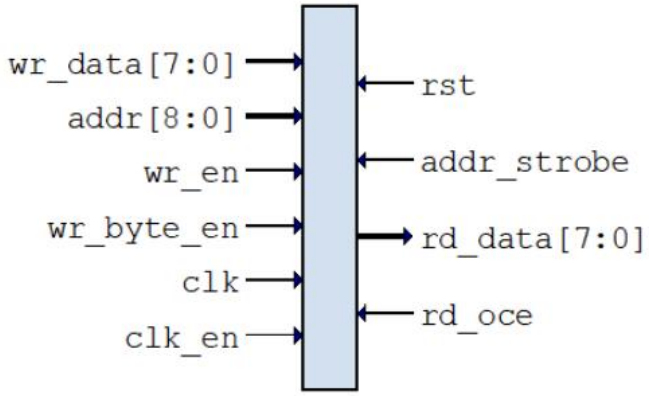


图 3.2-1 DRM Based Single Port RAM 接口框图

接口列表及描述如下:

表 3.2-1 DRM Based Single Port RAM 接口描述

端口	I/O	描述
wr_data	I	写数据信号, 位宽范围 1~1152。
wr_addr	I	写地址信号, 位宽范围 6~20。
wr_en	I	写使能信号。 1: 写使能; 0: 读使能。
wr_clk	I	写时钟信号。
wr_clk_en	I	写时钟使能信号。 1: 对应地址有效; 0: 对应地址无效。
wr_rst	I	写端口复位信号。 1: 复位; 0: 复位释放。
wr_byte_en	I	Byte Write 使能信号, 位宽范围 1~128。 1: 对应 Byte 值有效; 0: 对应 Byte 值无效。

wr_addr_strobe	I	写地址锁存信号。 1: 对应地址无效， 地址被保持; 0: 对应地址有效。
rd_data	O	读数据信号, 位宽范围 1~1152。
rd_addr	I	读地址信号, 位宽范围 6~20。

3.2.2.原语接口列表

GTP_DRM18K_E1 是调用 DRAM 的原语之一, 有 18K bit 存储单元, 支持双端口 RAM、简单双端口 RAM、单端口 RAM、ROM 模式, 且在双端口 RAM、简单双端口 RAM 模式下支持双端口混合数据位宽。

功能框架如下图所示:

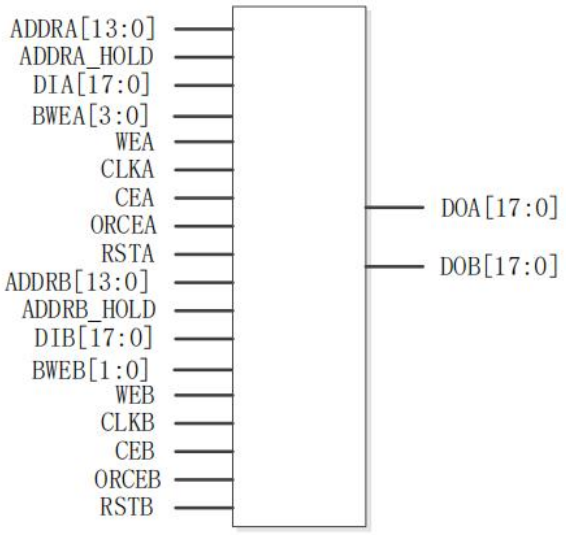


图 3.2-2 GTP_DRM18K_E1 功能框架

端口描述如下所示:

表 3.2-2 GTP_DRM18K_E1 端口描述

端口命名	输入/输出	描述
ADDRA[13:0]	输入	A 端口输入地址
ADDRA_HOLD	输入	A 端口地址输入选择
DIA[17:0]	输入	A 端口数据输入
WEA	输入	A 端口写使能
BWEA[3:0]	输入	A 端口字节写使能
CLKA	输入	A 端口时钟

CEA	输入	A 端口输入寄存器时钟使能
ORCEA	输入	A 端口输出寄存器时钟使能
RSTA	输入	A 端口数据寄存器复位
DOA[17:0]	输出	A 端口数据输出
ADDRB[13:0]	输入	B 端口输入地址
ADDRB_HOLD	输入	B 端口地址输入选择
DIB[17:0]	输入	B 端口数据输入
WEB	输入	B 端口写使能
BWEB[1:0]	输入	B 端口字节写使能
CLKB	输入	B 端口时钟
CEB	输入	B 端口输入寄存器时钟使能
ORCEB	输入	B 端口输出寄存器时钟使能
RSTB	输入	B 端口数据寄存器复位
DOB[17:0]	输出	B 端口数据输出

3.3. 工程说明

3.3.1.测试参数说明

- 测试中有关基于 DRAM 的单端口 ram IP 核参数设置如下:
  - DRAM 资源模式: DRAM18K ；
  - 数据位宽: 8 ；
  - 地址位宽: 9 ；

其他参数保持默认。不使用输出寄存器。

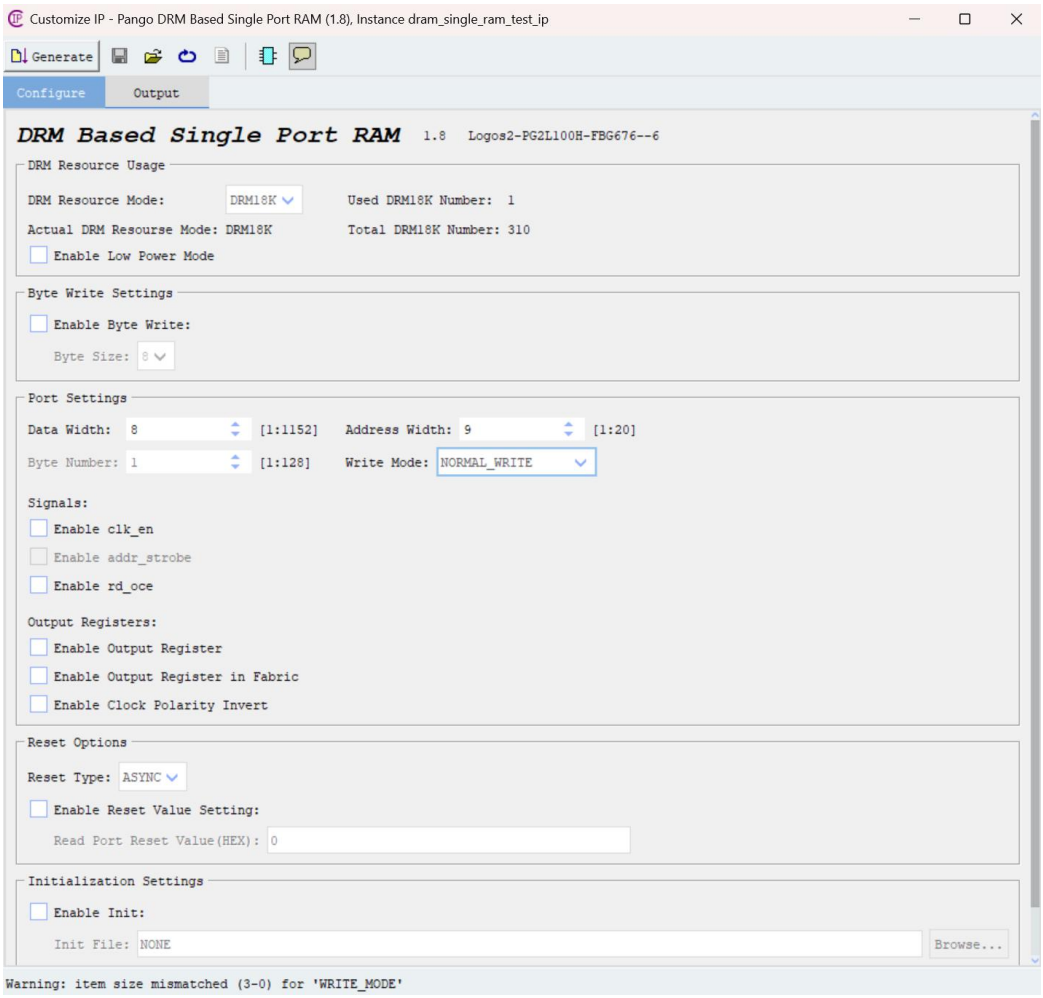


图 3.3-1 基于 DRAM 的单端口 ram IP 核参数设置

- IP 核使用时钟: 27MHz。
- 使用 GTP_DRM18K_E1 原语的 A 端口进行单端口（SINGLE PORT RAM）读写测试
- GTP 参数设置设置如下:
  - DATA_WIDTH_A ：8
  - RAM_MODE: SINGLE PORT(单端口 RAM)
  - WRITE_MODE_A ：NORMAL_WRITE
  - 其他参数保持默认。不使用输出寄存器。
- GTP 使用时钟: 27MHz。
- 代码设置测试数据如下:
  - 读写间隔: 512 个时钟周期读, 512 个时钟周期写;
  - 测试数据: 在 RAM 进行写操作时, 数据进行自加, 数据范围为 0~255, 数据加到 255 时, 重新返回 0 值继续进行自加。

（DRM Based Single Port RAM IP Core 的复位为高电平有效）

- debug 核使用时钟:

测试结果使用 debug 核观察, debug 工作时钟选择使用 108MHz, 该时钟使用 PLL 产生。

- 该测试工程 debug 核采样深度设置为 4096。

3.3.2.使用 debug 观察测试结果

使用 debug 观察 IP_wr_en、IP_addr、IP_wr_data、IP_rd_data、GTP_wr_en、GTP_addr、GTP_wr_data、GTP_rd_data 八个信号查看测试结果。该测试工程 debug 核采样深度设置为 4096。

Debug 使用请参考 PDS 软件目录下\doc\ Fabric_Debugger_User_Guide.pdf

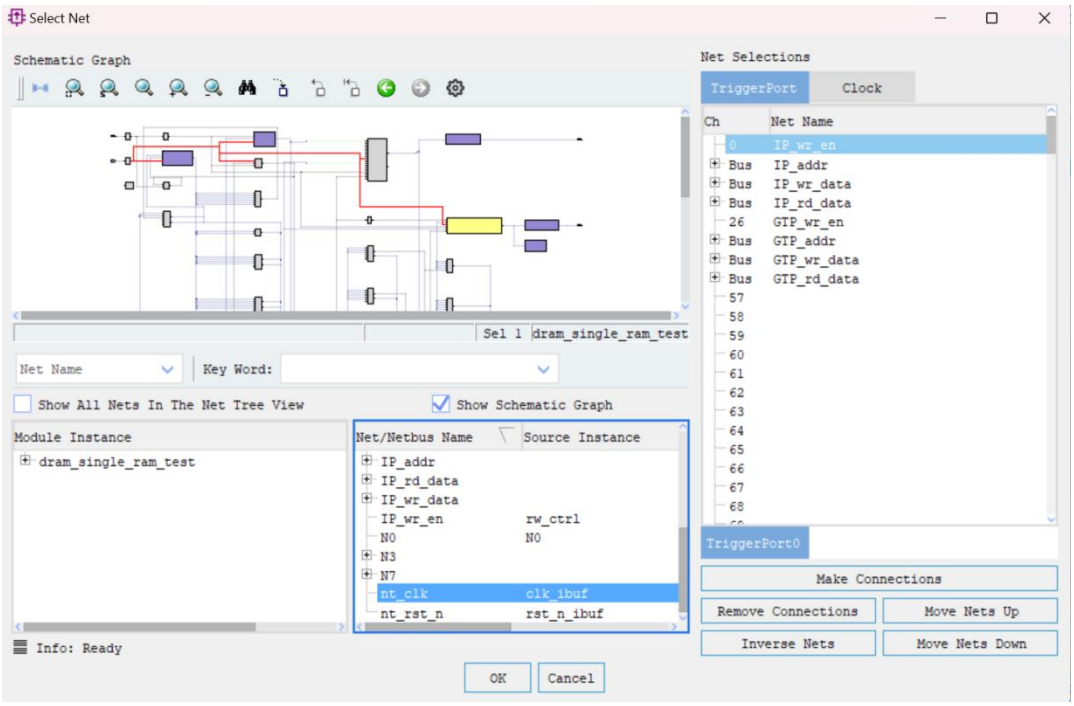


图 3.3-2 Select Net 配置

测试结果分析:

使用分布式单端口 RAM 进行写操作时, 写使能信号拉高, 在地址 0 处写入数据 0, 在地址 1 处写入数据 1, 在地址 2 处写入数据 2, 在地址 3 处写入数据 3, 在地址 4 处写入数据 4, 以此类推, 在地址 255 处写入数据 255, 在地址 256 处写入数据 0, 在地址 257 处写入数据 1, 等等。

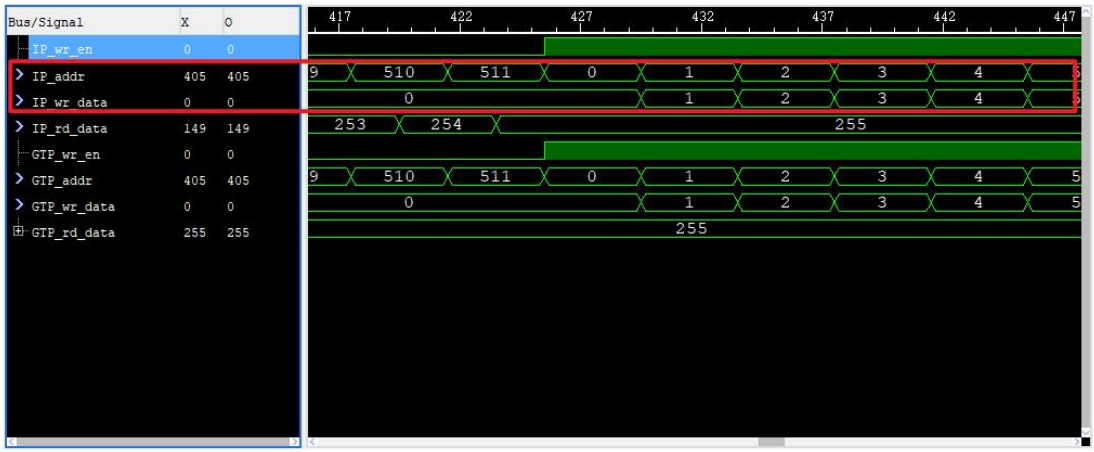


图 3.3-3 DRM Based Single Port RAM 写操作测试 1

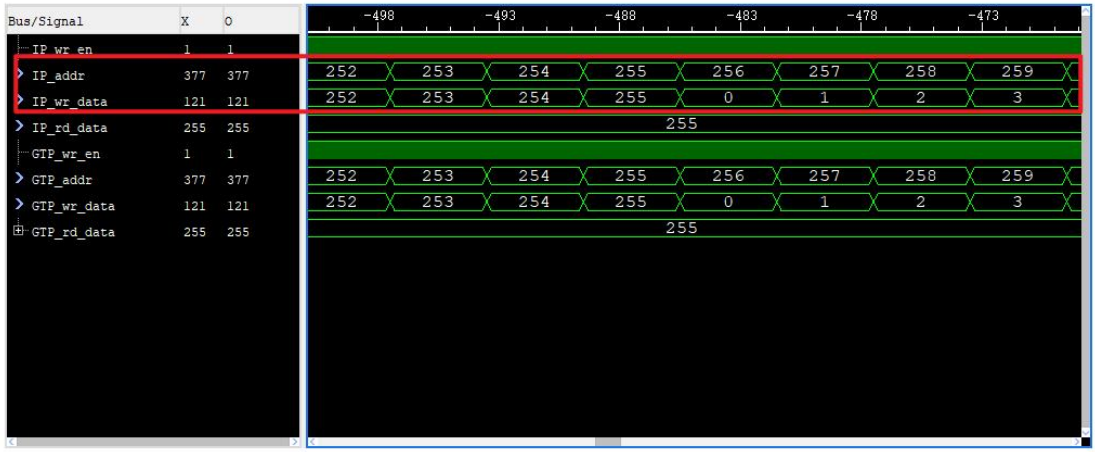


图 3.3-4 DRM Based Single Port RAM 写操作测试 2

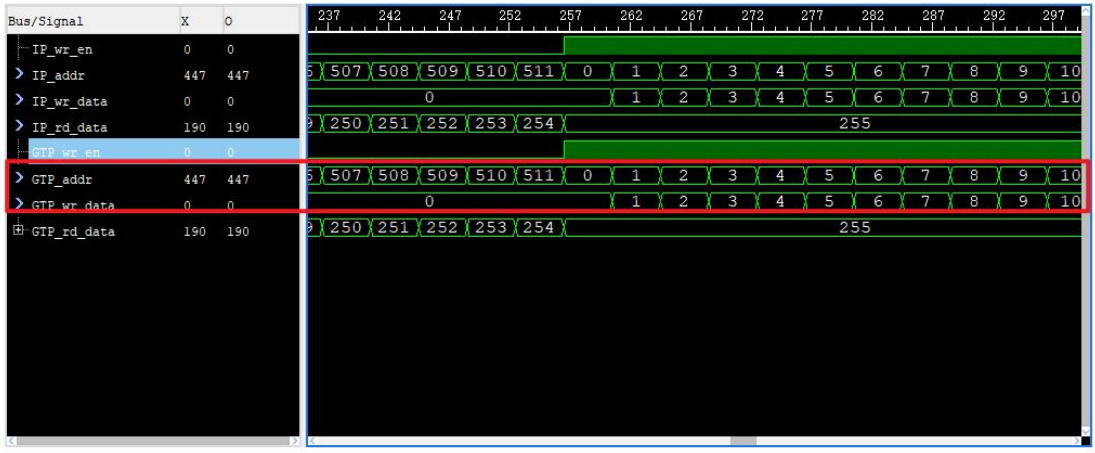


图 3.3-5 GTP_DRM18K_E1 写操作测试 1

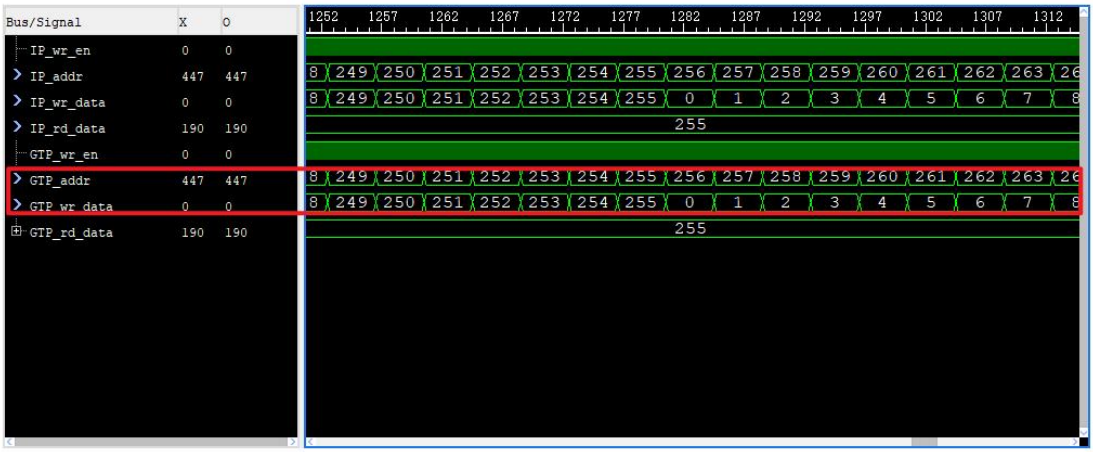


图 3.3-6 GTP_DRM18K_E1 写操作测试 2

使用分布式单端口 RAM 进行读操作时，写使能信号拉低，地址信号输入 IP 核后，数据隔一个时钟周期输出，可以观察到地址 0 输出数据 0，地址 1 输出数据 1，地址 2 输出数据 2，地址 3 输出数据 3，地址 4 输出数据 4，地址 255 输出数据 255，地址 256 输出数据 256，地址 257 输出数据 257。

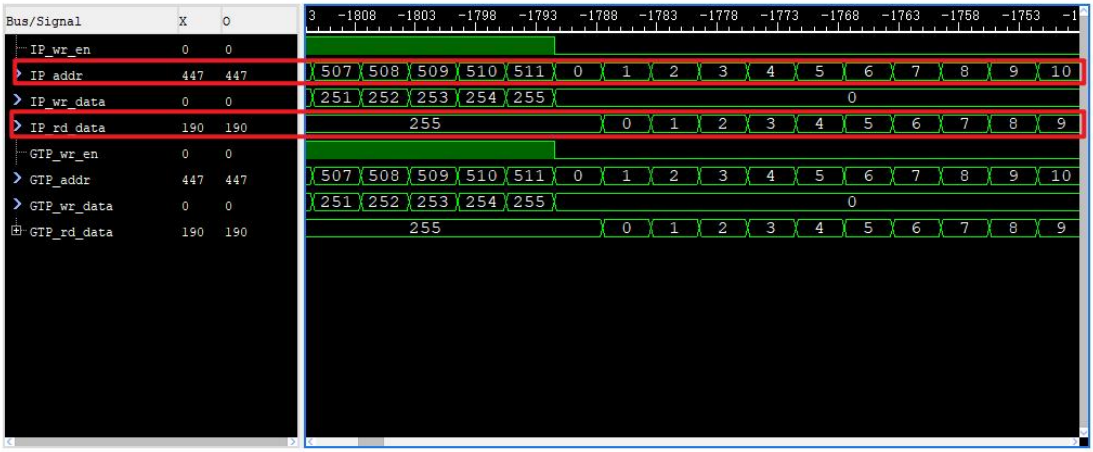


图 3.3-7 DRM Based Single Port RAM 读操作测试 1

图 3.3-8 DRM Based Single Port RAM 读操作测试 2

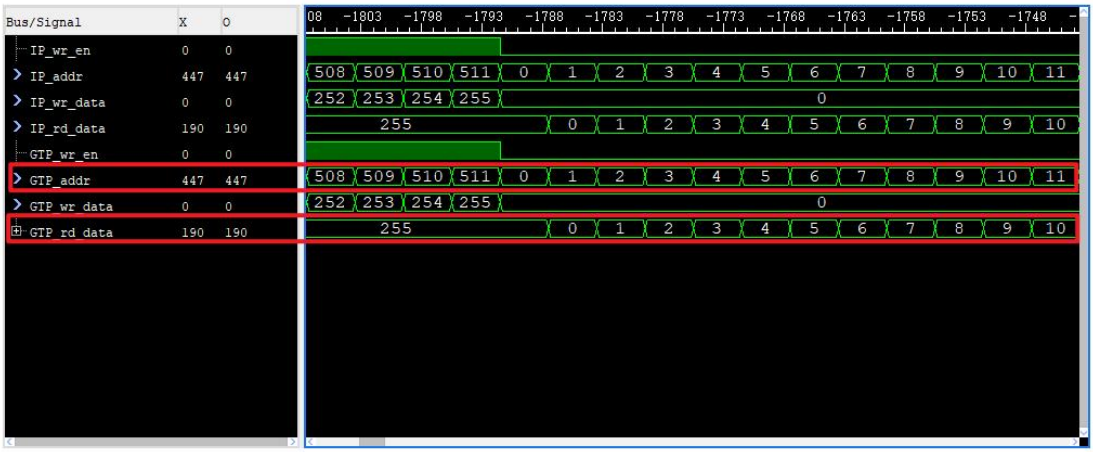


图 3.3-9 GTP_DRM18K_E1 读操作测试 1

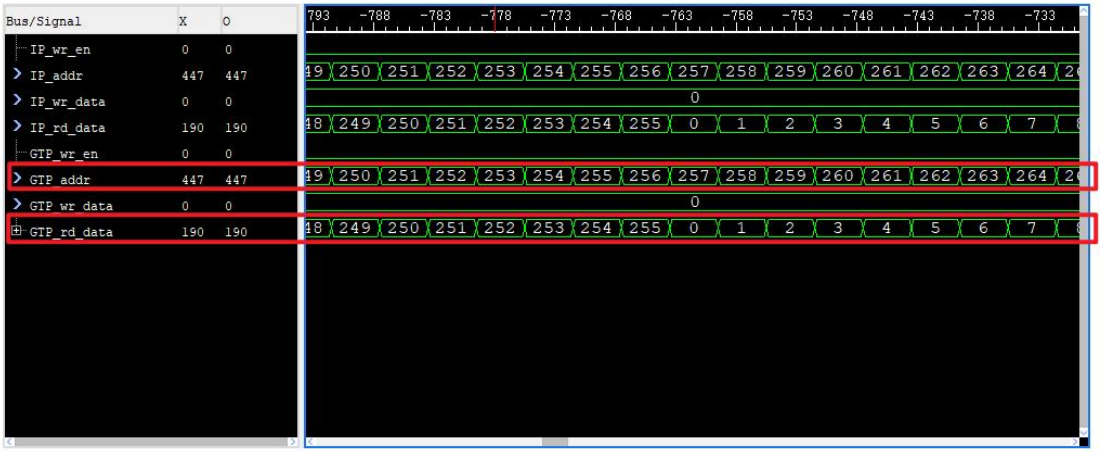


图 3.3-10 GTP_DRM18K_E1 读操作测试 2

4. 基于 APM 的乘法 DSP_mult 模块

实现 P = X*Y 乘法功能, PCOUT 输出用于与其他 DSP 模块级联。

4.1. 实验原理

通过配置 APM 功能寄存器调用内部 25 x 18 的乘法器实现

MODEIN (5'b00110)// MODEIN[3:0]=4'b0110;X_MULT=X2+XB, 预加器为加法（未使能 PREADD 功能）, 故 X_MULT=X2, MODEIN[4]=1'b0,故选择 Y2, 可通过 X_REG、Y_REG 寄存器设置输入寄存器个数, 如未使能 X/Y 端口寄存器, X1/X2 与 X 相当, Y1/Y2 与 Y 相当;

MODEY(3'b001) //MODEY 控制 POSTADD 功能, MODEY[1:0]=2'd1,选择为乘法器输出, MODEY[2]=1'b0,不对 Ymux 输出取反;

MODEZ (4'b0010) //MODEZ[3]=1'B0,输出不取反, MODEZ[2:0]=3'd2,Zmux 选择为 Z 输入端

- 端口配置

X_REG

当 X_REG 为 0 时, X 端口输入无寄存器

当 X_REG 为 1 时, X 端口输入 1 寄存器

当 X_REG 为 2 时, X 端口输入 2 寄存器

Y_REG

当 X_REG 为 0 时, Y 端口输入无寄存器

当 X_REG 为 1 时, Y 端口输入 1 寄存器

当 X_REG 为 2 时, Y 端口输入 2 寄存器

延迟计算

AMP 原语中, 默认 MULT_REG、P_REG 寄存器有效。列表中为延迟

X\Y	M	P	Latancy
0	1	1	2
1	1	1	3
2	1	1	4

4.2. 接口列表

端口	I/O	位宽	描述
X_reg	parameter	1	X 寄存器延迟配置
Y_reg	parameter	1	Y 寄存器延迟配置

clk	input	1	系统时钟
rst	input	1	系统复位, 0 复位, 1 复位释放;
X	Input	30	X 数据输入
Y	Input	18	Y 数据输入
P	output	48	乘法输出数据 P

4.3. 工程说明

打开工程 DSP_mult, Sources 栏中右击 tb_DSP_mult, 点击 Run Behavior Simulation 调用出第三方仿真（提前编译库并设置仿真路径）, 如下:

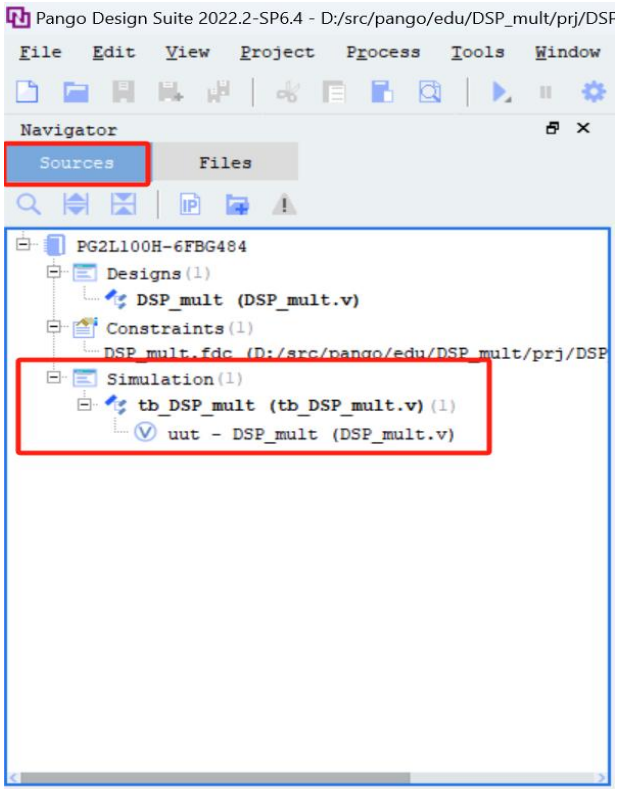


图 4.3-1 仿真文件界面

仿真文件中 X 依次累加 100 产生乘数, Y 依次累加 33 产生乘数, 得到乘法运算结果如下:

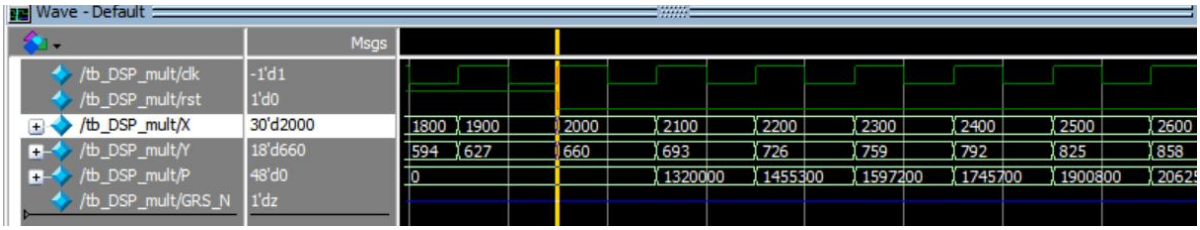


图 4.3-2 仿真波形结果

5. 基于 APM 乘累加 DSP_mult_as_cas 模块

实现输入数据 X\Y 的乘累加, 即 P = P + X*Y。

5.1. 实验原理

通过配置 APM 功能寄存器实现乘法,乘累加的 GTP 例化模板如下:通过 USE_POSTADD=1 使能累加功能, 不使能 PREADD 功能, 通过 X_REG 控制 X 寄存器使能、通过 Y_REG 控制 Y 寄存器使能, 使能 P_REG ；

MODEIN(5'b00010)//MODEIN [3:0]=4'b0010, X_MULT=X2, MODEN[4]=1'b0,Y=Y2,预加器为加法（未使能 PREADD 功能）; MODEY(3'b001)//MODEY [1:0]=3'd1, 选择为乘法器输出; MODEY[2]=1'b0,不对 Ymux 输出取反（取反 结果变为 P=P-X*Y）;

MODEZ(4'b0001)//MODEZ [3]=1'b0,不对 Zmux 输出取反(取反结果变为 P=-P+X*Y),MODEZ [2:0]=3'b001, 选择累加反馈;

ROUNDMODE_SEL=0, P_INIT0 =48'd0, P_INIT1=48'd0, rounding 功能设置为 Round-floor ；

在乘累加模式中, 需注意先将 MODEZ 设为 4'b0000, 将 APM 内部的累加值初始为零, 再将 MODEZ 值设为 4'b0001 进行乘累加计算

- 端口配置

X_REG

当 X_REG 为 0 时, X 端口输入无寄存器

当 X_REG 为 1 时, X 端口输入 1 寄存器

当 X_REG 为 2 时, X 端口输入 2 寄存器

Y_REG

当 X_REG 为 0 时, Y 端口输入无寄存器

当 X_REG 为 1 时, Y 端口输入 1 寄存器

当 X_REG 为 2 时, Y 端口输入 2 寄存器

延迟计算

AMP 原语中, 默认 MULT_REG、P_REG 寄存器有效。列表中为延迟

X\Y	M	P	Latancy
0	1	1	2
1	1	1	3
2	1	1	4

5.2. 接口列表

端口	I/O	位宽	描述

X_reg	parameter	1	X 寄存器延迟配置
Y_reg	parameter	1	Y 寄存器延迟配置
clk	input	1	系统时钟
rst	input	1	系统复位, 0 复位, 1 复位释放;
X	Input	30	X 数据输入
Y	Input	18	Y 数据输入
P	output	48	乘法输出数据 P
CPO	output	48	级联 P 输出

5.3. 工程说明

打开工程 DSP_mult, Sources 栏中右击 tb_DSP_mult, 点击 Run Behavior Simulation 调用出第三方仿真（提前编译库并设置仿真路径）, 如下:

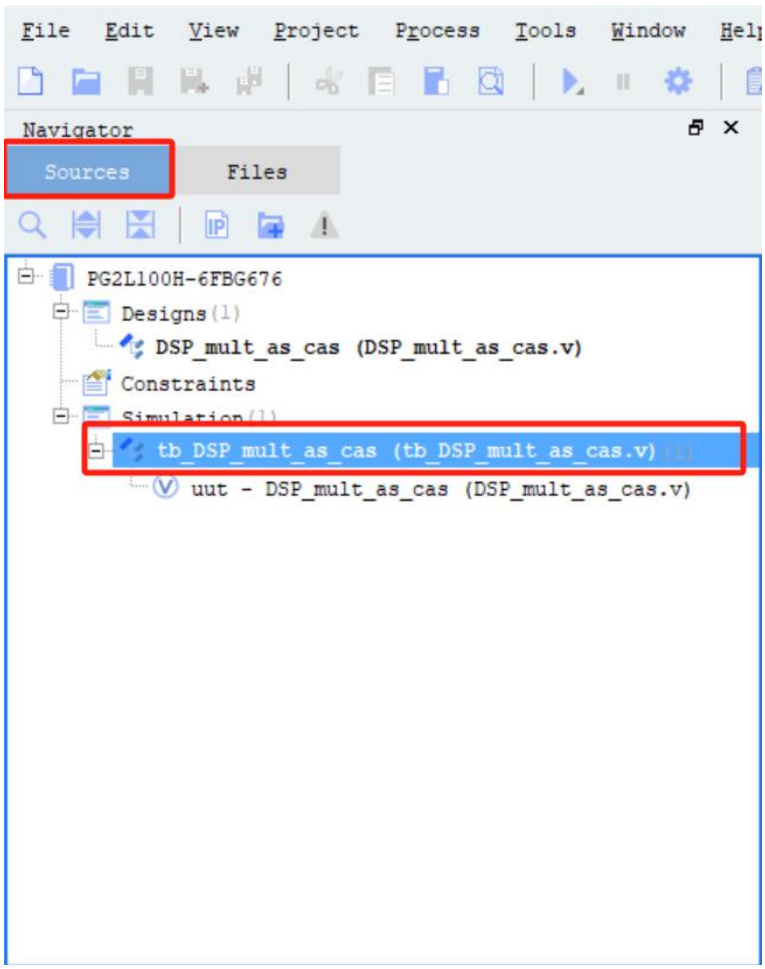


图 5.3-1 仿真文件界面

仿真文件中 X 依次累加 100 产生乘数, Y 依次累加 33 产生乘数, 得到乘累加运算结果 P 及可用于级联的 CPO 如下:

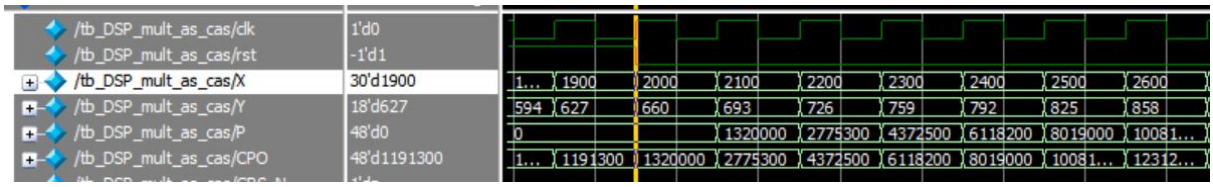


图 5.3-2 仿真波形结果

5.4. 实验内容

利用 MES2L676-100HP-MINI 开发板实现 HDMI 输出彩条显示，并且可通过按键切换输出分辨率，支持分辨率如表 5.4-1 所示：

表 5.4-1 支持分辨率列表

按键次数	分辨率
默认	1920*1080P60
1	1280*800P60
2	1360*768P60
3	1400*1050P60
4	1440*900P60
5	1600*900P60
6	1600*1200P60
7	1680*1050P60
8	1280*720P60
9	1920*1080P60
10	1920*1200P60

5.5. 实验原理

MES2L676-100HP-MINI 开发板的 HDMI 接收和发送接口采用 MS7200 和 MS7210 实现，芯片兼容 HDMI1.4b 及以下标准视频的 3D 传输格式,配置接口采用 IIC 接口,最高分辨率高达 4K@30Hz，最高采样率达到 300MHz，数字接口支持 YUV 及 RGB 格式。

在本实验中，实现 HDMI 彩条显示且通过按键控制分辨率切换，切换过程中图像输出信号时序调整的同时像素时钟频率也需要做对应的调整，因此可通过 PLL 动态调整实现像素时钟调整。实现框架如下图所示，FPGA 实现对 MS7200 的 IIC 接口配置及对按键按下次数进行计数，根据计数结果对应的分辨率调整图像输出信号时序及配置对应的像素时钟频率，实现多种分辨率的切换。

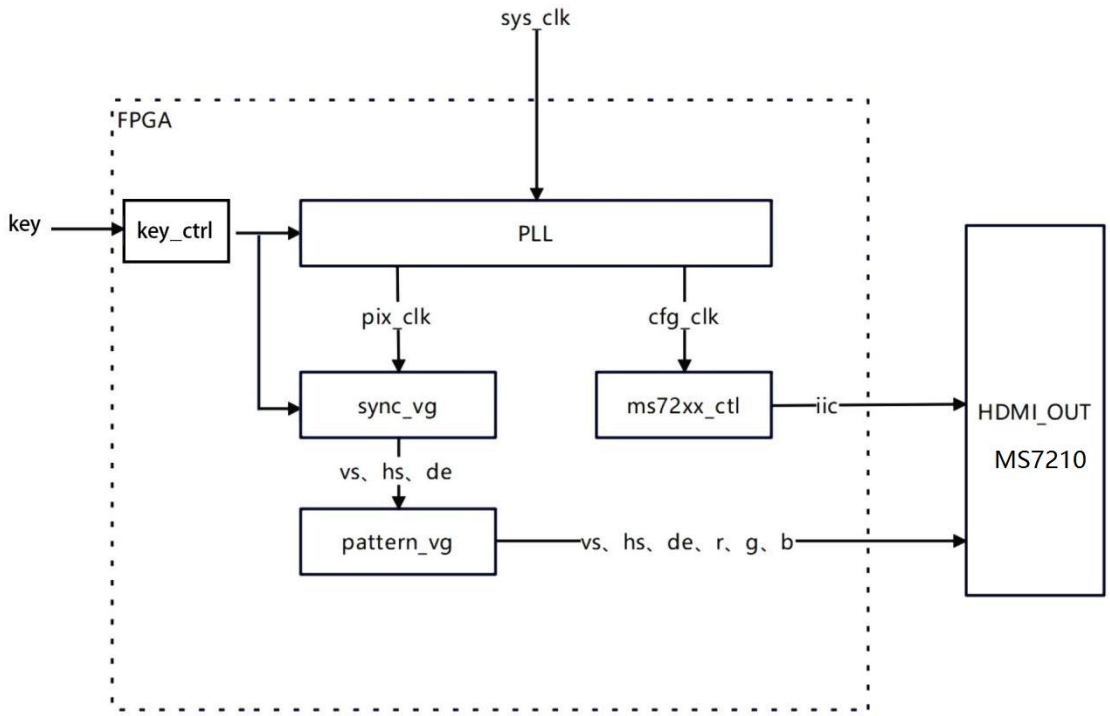


图 5.5-1 键控 HDMI 分辨率切换实现框图

5.6. PLL 动态调整输出频率

PLL IP 是紫光同创基于 PLL 及时钟网络资源设计的 IP,通过不同的参数配置,可实现时钟信号的调频、调相、同步、频率综合等功能。

Logos2 系列器件的 PLL 支持 APB 动态重配功能,实现 HDMI 像素时钟频率的调整可通过 APB 接口动态配置 PLL 输出时钟频率(时钟相位和占空比不做调整)。

PLL 的使用可选择 Basic 和 Advanced 两种模式,两种模式均支持 APB 接口动态配置实现时钟频率的动态调整,Advanced 模式下 PLL 的内部参数配置完全开放。Basic 和 Advanced 两种模式下均可选择使用 GPLL 或 PPLL,GPLL 支持分数分频可实现更精确的输出频率,GPLL 与 PPLL 详细描述请参考《UG040004_Logos2 系列 FPGA 时钟资源(Clock)用户指南_V1.2.pdf》

5.6.1.IP 配置操作流程

1、点击菜单项 Tools->IP Compiler 或直接点击工具栏中的 IP Compiler 按钮 , 打开 IP compiler 工具

2、选中 PLL IP,可点击 view datasheet 工具查看对应 IP 的用户指南

3、命名及路径设置

4、点击右侧 Customize 对 IP 进行配置

以上操作如图 5.6-1 所示

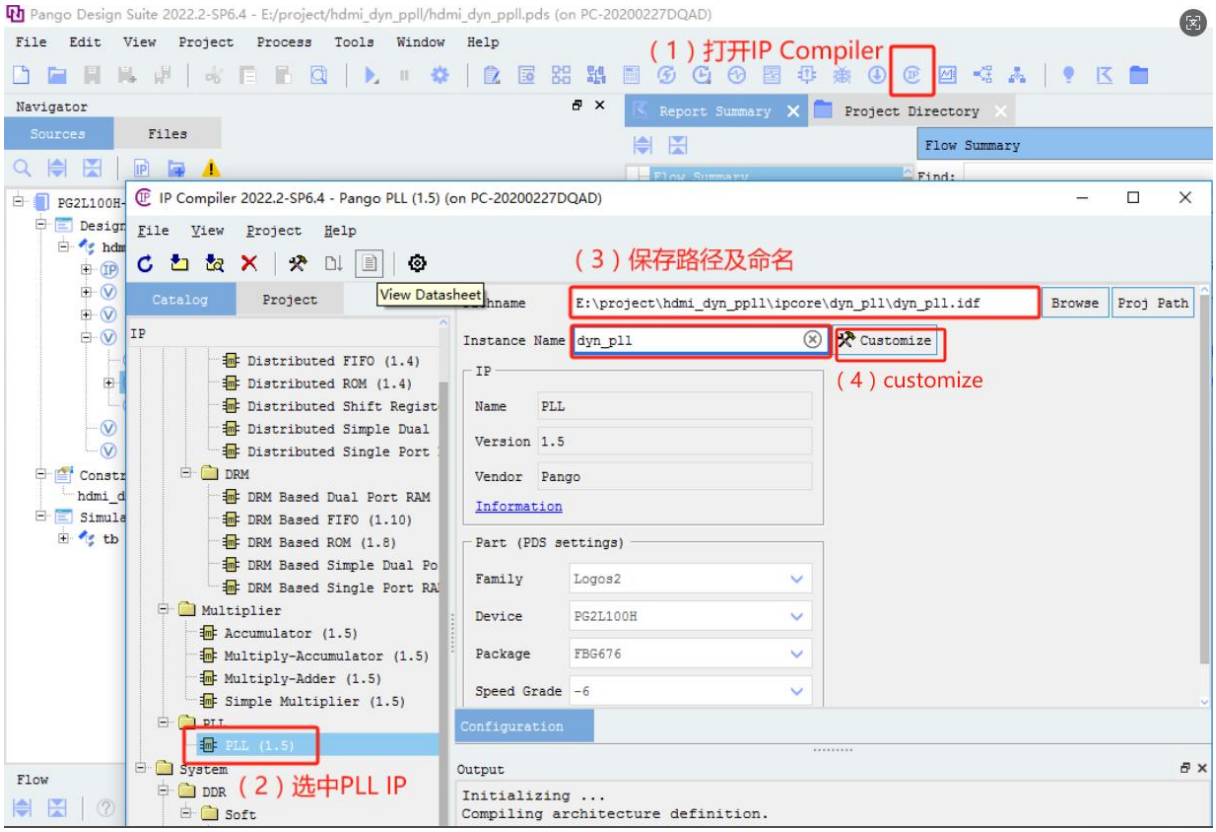


图 5.6-1 IP Compiler 操作流程图

5、Mode Selection 选择 Basic Configurations 模式，如图 5.6-2 所示

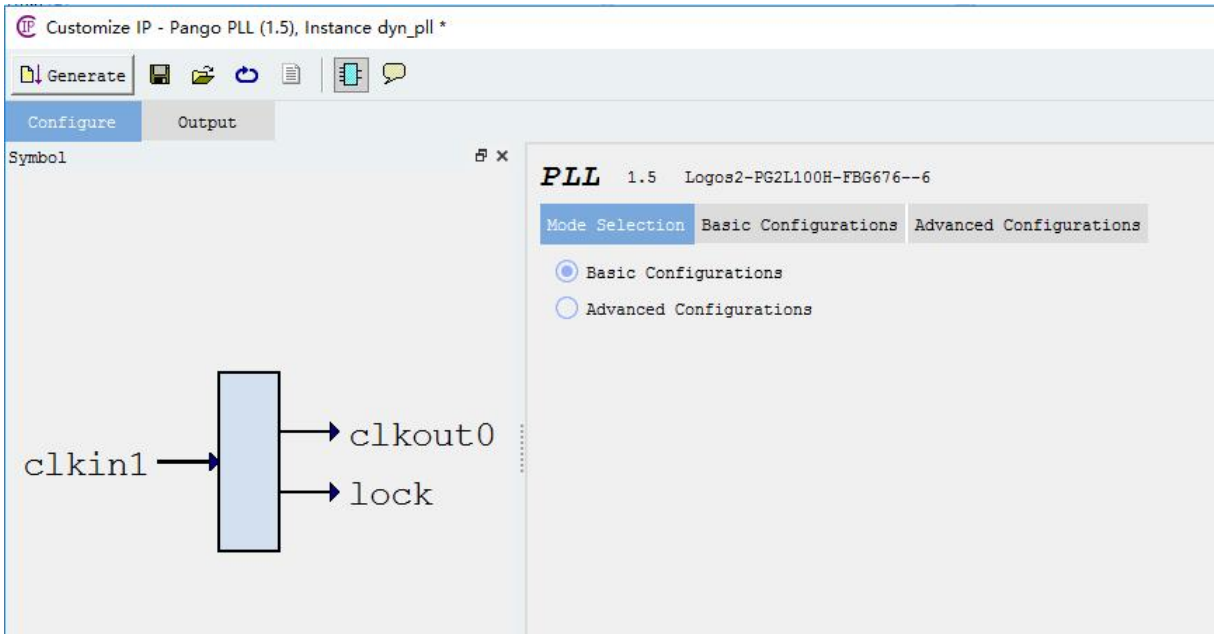


图 5.6-2 PLL 配置 Mode Selection

6、点击 Basic Configurations 界面

7、PLL Mode Configurations 选择 PPLL，若需更精确的输出时钟频率可选择支持分数分频的 GPLL

8、Public Configurations

（1）选择内部反馈模式，反馈时钟选择 CLKOUTF

（2）Input Clock clkin Frequency 填写输入时钟频率

（3）勾选 Enable Port rst，动态配置完成后需对 PLL 重新复位

（4）勾选 Enable APB Bus，动态配置使用 APB 接口

9、Clockout0 Configurations：勾选 Enable clkout0，设置默认输出时钟频率为 148.5MHz，相位不作调整，占空比设置为 50%

以上操作如图 5.6-3 所示

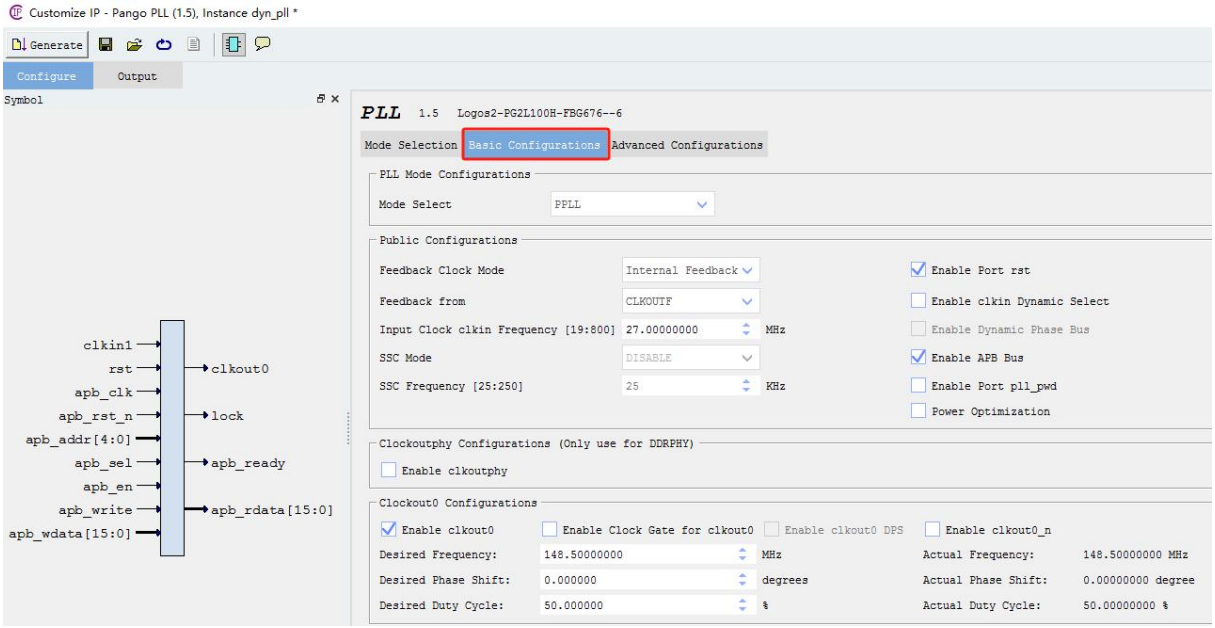


图 5.6-3 Basic Configurations 配置

10、若需查看 IP 内部分频参数等数值，可在界面下方勾选 show Internal Setting of PLL，如图 5.6-4 所示

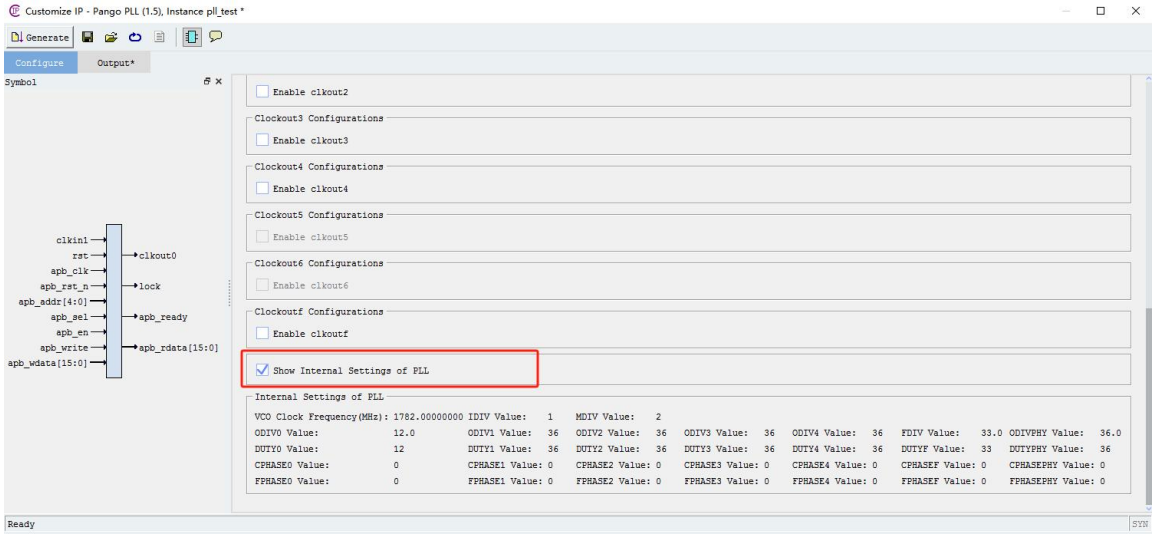


图 5.6-4 show Internal Setting of PLL 设置

11、IP 配置完成后点击 Generate，如图 5.6-5 所示，完成后可对 IP 进行调用

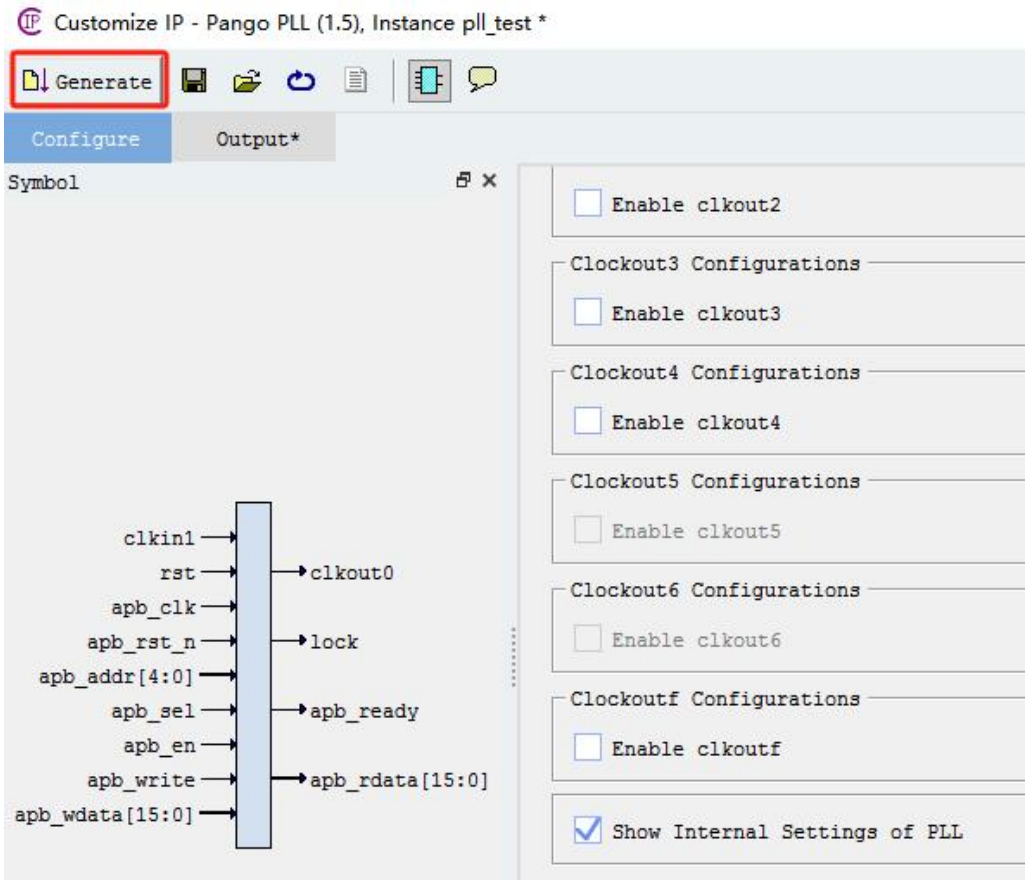


图 5.6-5 generate 生成 IP

配置完成后 PLL 端口如图 5.6-6 所示，其中 clkin1 为 PLL 输入时钟，clkout0 为 PLL 输出时钟，rst 为 PLL 复位信号,lock 信号为 PLL 频率锁定指示信号,指示 PLL 反馈时钟信号已锁定到参考时钟，以及 APB 配置接口。

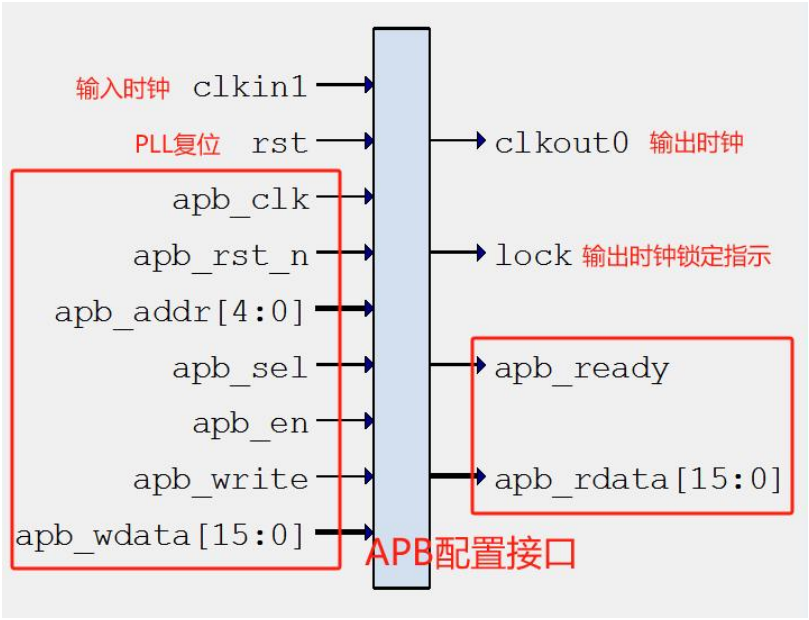


图 5.6-6 PLL 端口

基于以上配置，PLL 使用情况如图 5.6-7 所示，配置一路时钟输入，clkoutf 作为 PLL 内部反馈，并输出一路时钟 clkout0

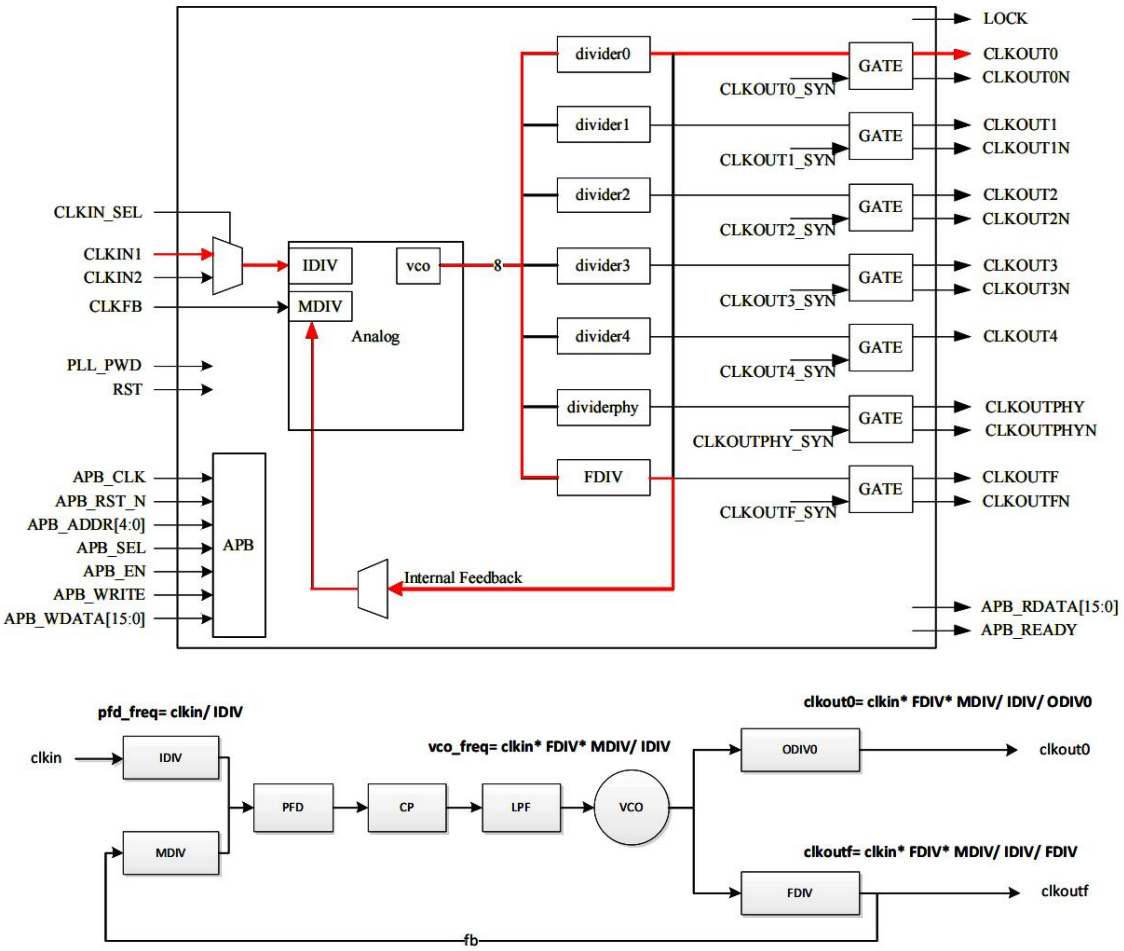


图 5.6-7 PLL 应用模式

5.6.2.输出频率计算

PLL 输出时钟频率由输入时钟频率、配置模式及相关配置参数决定，计算公式如下：

$$F_{CLKOUTx} = \frac{F_{IN} \times MDIV \times FBDIV}{IDIV \times ODIVx}$$

FCLKOUTx 为输出时钟频率；

FIN 为输入时钟频率；

MDIV 为反馈分频参数；

FBDIV 为 PLL 反馈时钟的输出分频参数，clkoutf 作为反馈时钟时 FBDIV = ODIVf；

IDIV 为输入分频器分频参数；

ODIVx 为输出时钟的输出分频参数，本实验使能 clkout0，所以 x=0 ，ODIVx = ODIV0；

根据像素时钟频率确定时钟配置参数，每次切换分辨率时 PLL 动态配置通过 APB 接口修改相关参数的寄存器数值实现输出频率调整。

5.6.3.APB 配置接口

APB 接口作为 PLL 的动态配置接口，若需要动态配置 PLL，PLL IP 需使能的 APB 接口，APB 接口

端口如表 5.6-1 所示:

表 5.6-1 APB 接口端口列表

序号	端口名称	I/O	描述
1	APB_CLK	I	时钟, 上升沿采样;
2	APB_RST_N	I	异步复位, 低有效; 只复位 Data Bus (不复位寄存器);
3	APB_ADDR[4:0]	I	地址总线;
4	APB_SEL	I	选择, 表示从设备被选择;
5	APB_EN	I	使能, 总线使能信号;
6	APB_WRITE	I	方向, 0 :读, 1: 写;
7	APB_WDATA[15:0]	I	数据总线输入;
8	APB_RDATA[15:0]	O	数据总线输出;
9	APB_READY	O	标志信号;

APB 接口写时序如图 5.6-8 所示

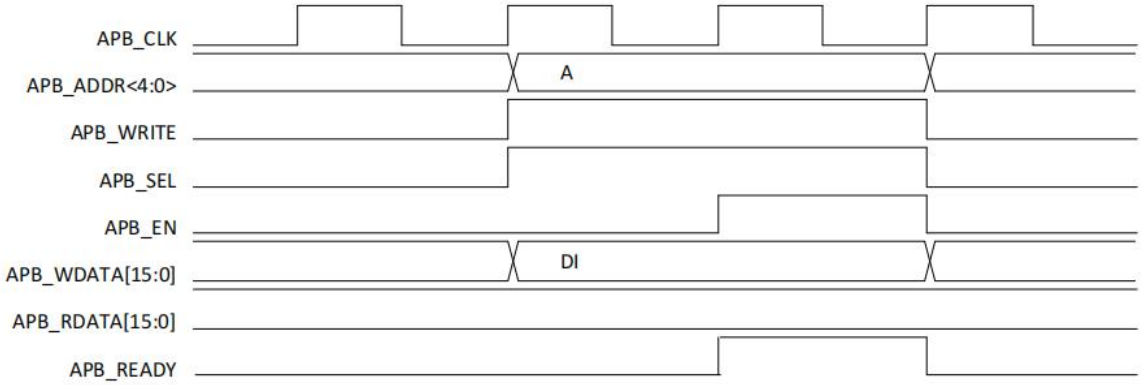


图 5.6-8 APB 写时序

APB 接口读时序如图 5.6-9 所示

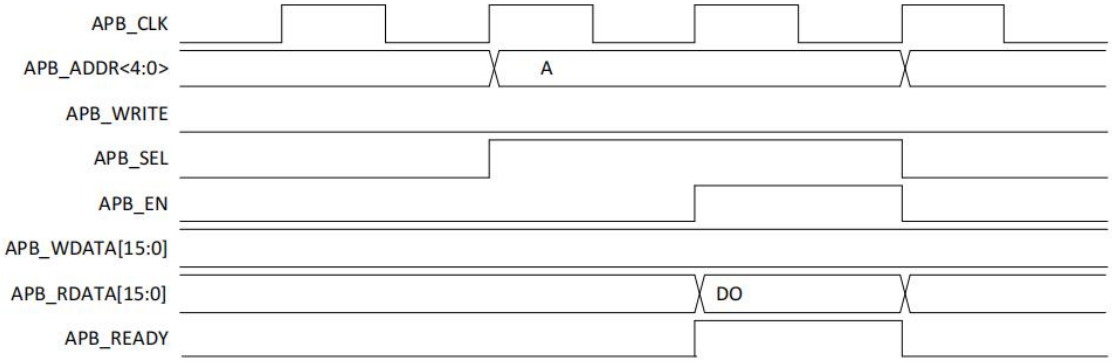


图 5.6-9 APB 读时序

5.6.4.APB 寄存器列表

通过 APB 接口修改相关寄存器数值重新配置输出时钟频率, 详细寄存器地址及其定义如表 5.6-2 所示:

表 5.6-2 APB 读时序

Address	Bit	寄存器	描述
0	[15:8]	REG_ODIV0_RATIO<7:0>,	ODIV0 Register1: 分频比设置;
	[7:0]	REG_ODIV0_DUTY<7:0>	ODIV0 Register1: 占空比设置;
1	[15:11]	RESERVED	
	[10]	REG_ODIV0_MUXSEL_EN	ODIV0 Register2: 输入时钟 MUX 使能;
	[9:7]	REG_ODIV0_FPHASE<2:0>	ODIV0 Register2: 相位细调设置;
	[6:0]	REG_ODIV0_CPHASE<6:0>	ODIV0 Register2: 相位粗调设置;
2	[15:8]	REG_ODIV1_RATIO<7:0>	ODIV1 Register1: 分频比设置;
	[7:0]	REG_ODIV1_DUTY<7:0>	ODIV1 Register1: 占空比设置;
3	[15:11]	RESERVED	
	[10]	REG_ODIV1_MUXSEL_EN	ODIV1 Register2: 输入时钟 MUX 使能;
	[9:7]	REG_ODIV1_FPHASE<2:0>	ODIV1 Register2: 相位细调设置;
	[6:0]	REG_ODIV1_CPHASE<6:0>	ODIV1 Register2: 相位粗调设置;
4	[15:8]	REG_ODIV2_RATIO<7:0>	ODIV2 Register1: 分频比设置;
	[7:0]	REG_ODIV2_DUTY<7:0>	ODIV2 Register1: 占空比设置;
5	[15:11]	RESERVED	
	[10]	REG_ODIV2_MUXSEL_EN	ODIV2 Register2: 输入时钟 MUX 使能;
	[9:7]	REG_ODIV2_FPHASE<2:0>	ODIV2 Register2: 相位细调设置;
	[6:0]	REG_ODIV2_CPHASE<6:0>	ODIV2 Register2: 相位粗调设置;
6	[15:8]	REG_ODIV3_RATIO<7:0>	ODIV3 Register1: 分频比设置;
	[7:0]	REG_ODIV3_DUTY<7:0>	ODIV3 Register1: 占空比设置;
7	[15:11]	RESERVED	
	[10]	REG_ODIV3_MUXSEL_EN	ODIV3 Register2: 输入时钟 MUX 使能;
	[9:7]	REG_ODIV3_FPHASE<2:0>	ODIV3 Register2: 相位细调设置;
	[6:0]	REG_ODIV3_CPHASE<6:0>	ODIV3 Register2: 相位粗调设置;
8	[15:8]	REG_ODIV4_RATIO<7:0>	ODIV4 Register1: 分频比设置;
	[7:0]	REG_ODIV4_DUTY<7:0>	ODIV4 Register1: 占空比设置;
9	[15:11]	RESERVED	
	[10]	REG_ODIV4_MUXSEL_EN	ODIV4 Register2: 输入时钟 MUX 使能;
	[9:7]	REG_ODIV4_FPHASE<2:0>	ODIV4 Register2: 相位细调设置;
	[6:0]	REG_ODIV4_CPHASE<6:0>	ODIV4 Register2: 相位粗调设置;
A	[15:8]	REG_ODIVPHY_RATIO<7:0>	ODIVPHY Register1: 分频比设置;
	[7:0]	REG_ODIVPHY_DUTY<7:0>	ODIVPHY Register1: 占空比设置;
B	[15:11]	RESERVED	
	[10]	REG_ODIVPHY_MUXSEL_EN	ODIVPHY Register2: 输入时钟 MUX 使能;
	[9:7]	REG_ODIVPHY_FPHASE<2:0>	ODIVPHY Register2: 相位细调设置;

	[6:0]	REG_ODIVPHY_CPHASE<6:0>	ODIVPHY Register2: 相位粗调设置;
C	[15:8]	REG_FDIV_RATIO<7:0>	FDIV Register1: 分频比设置;
	[7:0]	REG_FDIV_DUTY<7:0>	FDIV Register1: 占空比设置;
D	[15:11]	RESERVED	
	[10]	REG_FDIV_MUXSEL_EN	FDIV Register2: 输入时钟 MUX 使能;
	[9:7]	REG_FDIV_FPHASE<2:0>	FDIV Register2: 相位细调设置;
	[6:0]	REG_FDIV_CPHASE<6:0>	FDIV Register2: 相位粗调设置;
E	[15:7]	RESERVED	
	[6:0]	REG_IDIV_RATIO<7:0>	IDIV Register: 分频比设置;
F	[15:7]	RESERVED	
	[6:0]	REG_MDIV_RATIO<7:0>	MDIV Register: 分频比设置;
10	[15:14]	RESERVED	BANDWIDTH Register
	[13:12]	REG_VCTRL_INIT<1:0>	
	[11:10]	REG_CP_SELFBIAS_SEL<1:0>	
	[9:8]	REG_ICP_BASE_SEL<1:0>	
	[7:4]	REG_CP_CUR_SEL<3:0>	
	[3:1]	REG_LPF_R<2:0>	
	[0]	REG_LPF_C	
11	[15:6]	RESERVED	LOCK Register1
	[5]	REG_LOCK_FILTER_PD	
	[4:0]	REG_FREQ_LOCKDET_SET<4:0>	

5.6.4.1.1. PPLL 寄存器设置

占空比计算公式为 duty cycle = (50%/odiv)*duty, 若保持占空比为 50%不变, 需设置分频参数与占空比参数相同:

REG_ODIV0_DUTY=REG_ODIV0_RATIO

REG_FDIV_DUTY=REG_FDIV_RATIO

输出时钟相位不做调整:

REG_ODIV0_FPHASE=0

REG_ODIV0_CPHASE=0

REG_FDIV_FPHASE=0

REG_FDIV_CPHASE=0

根据 PLL 输出频率计算公式, 相关分频参数如下, 需根据实际输出频率确定分频参数数值:

REG_ODIV0_RATIO<7:0>

REG_FDIV_RATIO<7:0>

REG_IDIV_RATIO<7:0>

REG_MDIV_RATIO<7:0>

动态配置参数需符合以下要求:

- 选择输入频率必须在可配范围内
- 输出时钟频率必须在可配范围内
- PFD 频率必须在可配范围内
- 输入分频参数 IDIV、输出分频参数 ODIV, 反馈分频参数 MDIV、FBDIV 必须在可配范围内
- VCO 频率范围必须在规定范围内

$$F_{PFD} = \frac{F_{IN}}{IDIV}$$

$$F_{VCO} = \frac{F_{IN} \times MDIV \times FBDIV}{IDIV}$$

$$F_{CLKOUTx} = \frac{F_{IN} \times MDIV \times FBDIV}{IDIV \times ODIVx}$$

PPLL 具体要求如表 5.6-3 所示,此外需要注意,APB 改写内部寄存器后需复位 PLL 保证其正常工作,RST 最小脉冲宽度 10ns,详细可参考《DS04001_Logos2 系列 FPGA 器件数据手册_V2.1.pdf》

表 5.6-3 PPLL 交流特性

指标	最小值	最大值	单位	说明
FIN	19	800	MHz	输入时钟频率
FINJIT	200		ps	输入时钟抖动（FIN＜200MHz）
	0.04		UI	输入时钟抖动（FIN ≥200MHz）
FINDT	10-49MHz: 25% 50-199MHz: 30% 200-399MHz: 35% 400-499MHz: 40% 500-800MHz: 45%			输入时钟占空比
FPFD	19	450	MHz	PFD 工作频率范围
FVCO	1330	2133	MHz	VCO 工作频率范围
FOUT	10.39	2133	MHz	输出时钟频率范围
TPHO	0.12		ns	静态相位失调
TOUTJT	180		ps	输出时钟抖动（FOUT ≥100MHz）
T	0.018		UI	输出时钟抖动（FOUT＜100MHz）
TOUTDT	50%±5%			输出时钟占空比精度（50%情形）
FBW	1	4	MHz	带宽
TLOCK	---	120	us	锁定时间
TRST	10	---	ns	复位信号宽度

输出时钟频率及对应 APB 寄存器数值的确定,用户可通过计算的方式确定置参数配数值,也可通

过 IP 配置界面查看生成的对应参数数值，如图 5.6-10 所示

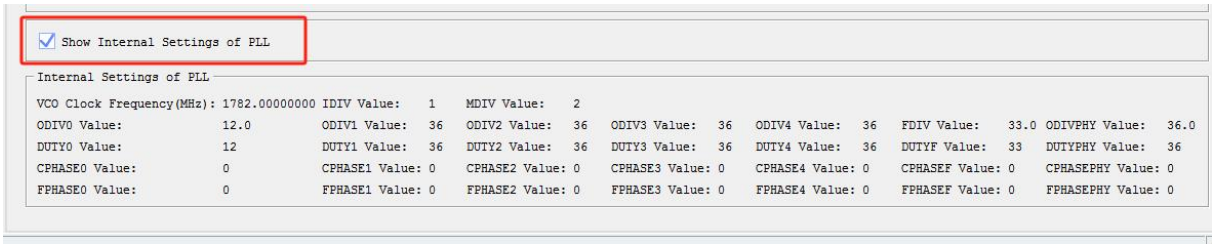


图 5.6-10 IP 参数数值查看

5.7. 工程介绍

代码设计框架如图 5.7-1 所示，按功能划分，主要包含以下模块：

ms72xx_ctrl: MS7210 配置；

key_ctrl: 按键计数，控制分辨率切换；

dyn_pll: 锁相环 PLL IP，动态调整时钟频率；

apb_cfg: APB 接口实现 pll 输出时钟频率动态配置；

sync_vg: 实现显示时序，图像行场同步等控制信号输出；

pattern_vg: 彩条显示像素值 RGB888 数据输出。

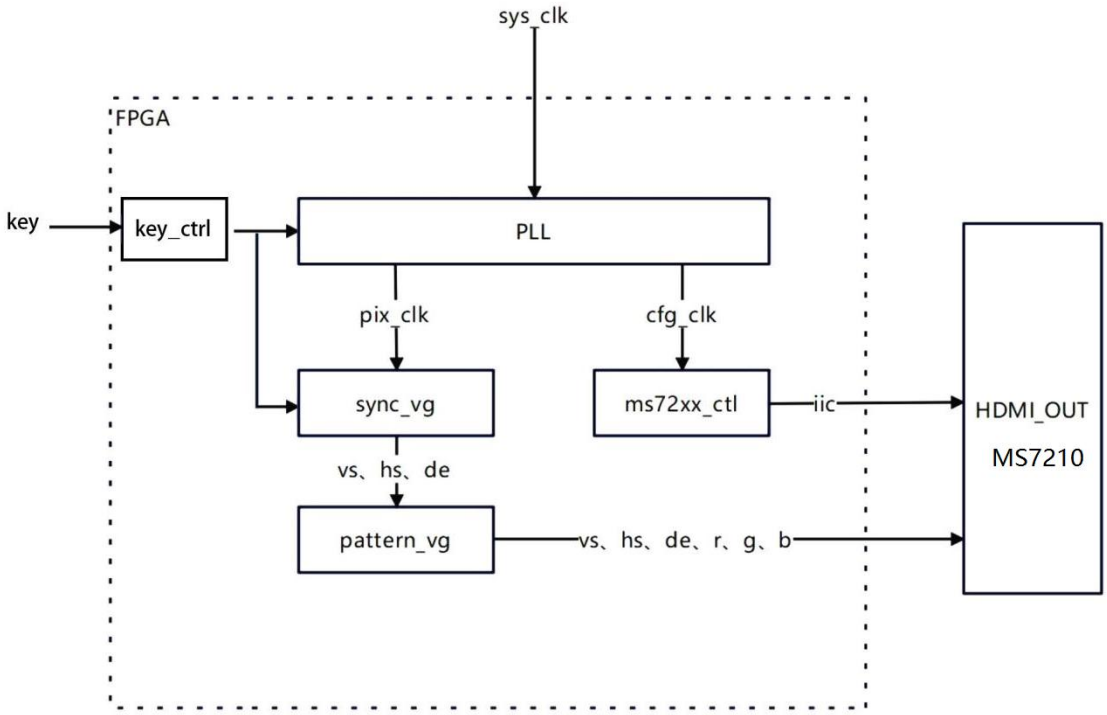


图 5.7-1 代码设计框架

5.7.1.ms72xx_ctrl 模块

使用 IIC 接口配置 MS7210 芯片,MES2L100HP 开发板硬件上设置 MS7210 的 ID 地址为 0xB2，配置 MS7210 数字接口为 RGB888，IIC 接口写操作流程如图 5.7-2 所示，MS7210 寄存器的配置请参考《MS7210 Register Map.pdf》。

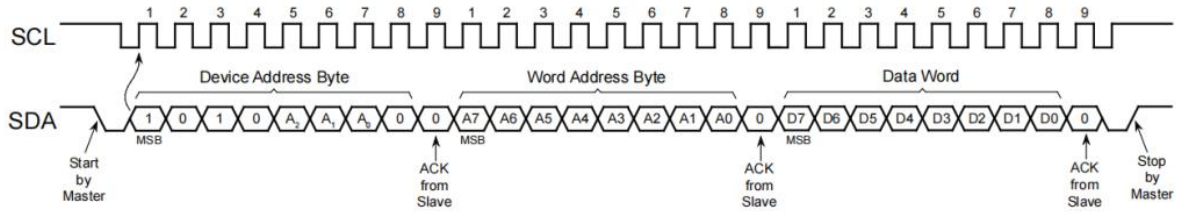


图 5.7-2 IIC 写操作

5.7.2.key_cnt 模块

实现按键信号消抖处理，再进行按键计数，通过按键按下次数控制 HDMI 显示分辨率。

5.7.3.sync_vg 模块

例如 1920*1080P60 分辨率为 1920*1080，帧率为 60。该分辨率下每一副画面都由行 1920 个像素点，列 1080 个像素点构成，按照“Z 字形”从左到右、从上到下的顺序快速点亮每一个像素点显示一幅画面如图 5.7-3 所示，帧率 60 表示每秒刷新 60 个画面。

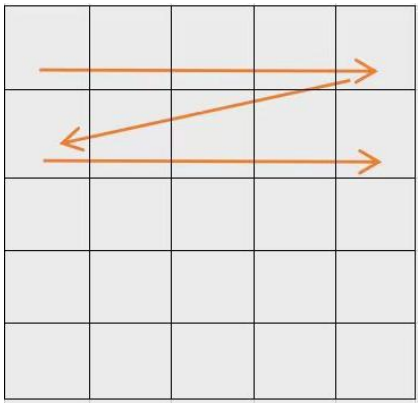


图 5.7-3 图像显示像素刷新顺序

像素数据源源不断输送进来，行、场的切换通过行场同步信号来控制，即 hsync（行同步）和 vsync（场同步信号）。

如图 5.7-4 所示，Addressable 部分内容是在显示器中可看到的区域，像素点是否有效通过 DE 信号标识；Border 可理解为显示黑边或者显示边框，通常 Border 显示的像素值是 0（黑色）。行、场切换过程都是在用户感受不到的区域进行的，这个区域就是 Blanking 部分，称为消隐区间。行同步信号 Hsync 有效表示新的一行开始，场同步信号 Vsync 有效表示新的一场开始。

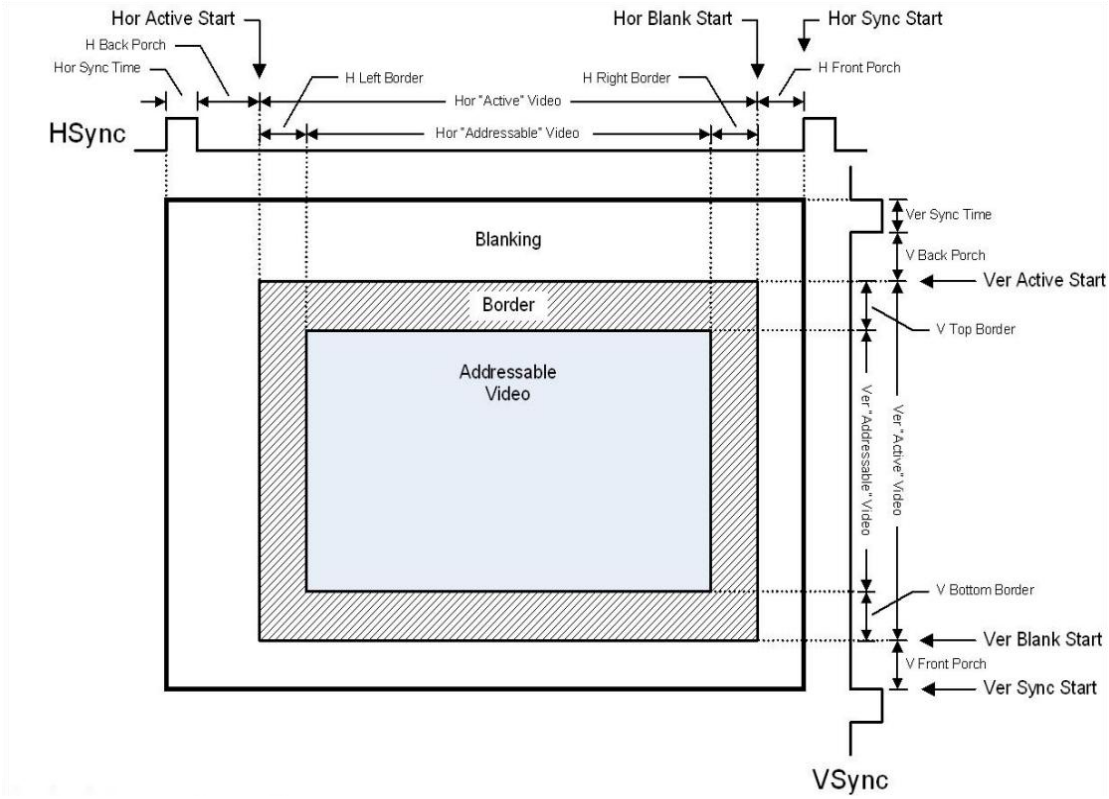


图 5.7-4 HDMI 图像显示时序定义

图像信号波形如图 5.7-5 所示, vsyn 信号有效之后开始一帧中第一行数据的传输, 每一行数据传输由 HSYNC 同步, 且 DE 信号有效时像素数据（RGB）得到传输。

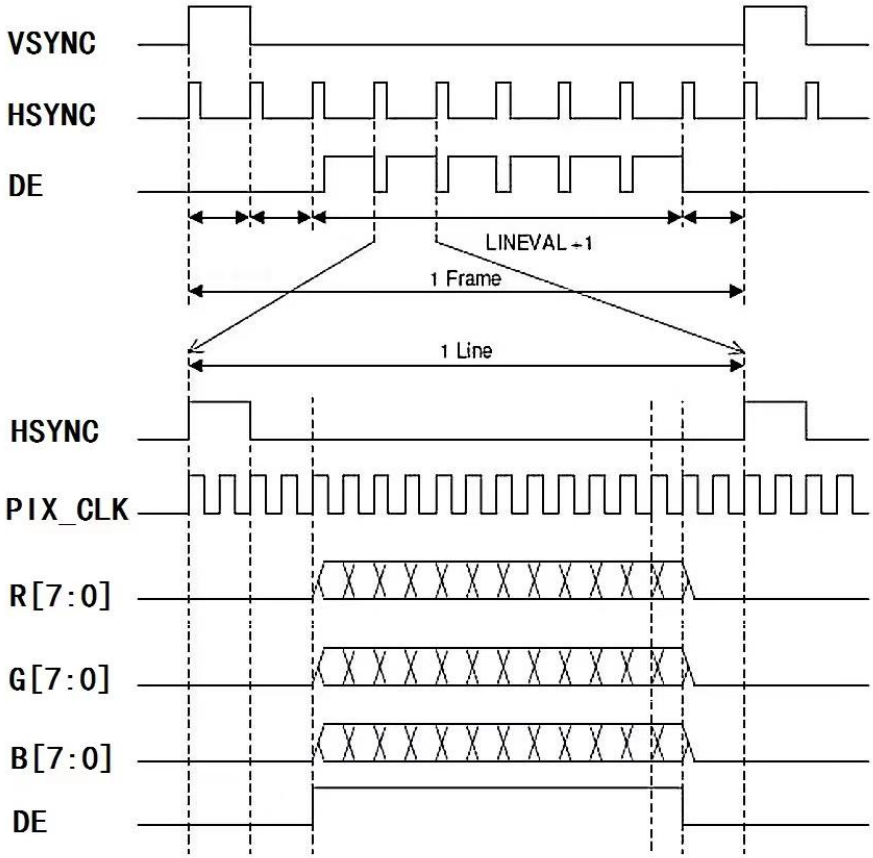


图 5.7-5 图像传输信号波形图

根据 key_cnt 模块的按键计数结果确定 HDMI 分辨率，调整行同步、场同步及使能信号时序（vs、hs、de），并为 pattern_vg 模块提供行有效像素点参数以便彩条显示像素数据生成，详细时序参数如图 5.7-6 所示，根据实现的 HDMI 显示格式，可查阅 vesa 时序标准及对应时钟频率。

VESA MONITOR TIMING STANDARD

Adopted: 11/17/08
Resolution: 1920 x 1080 at 60 Hz (non-interlaced)
EDID ID: DMT ID: 52h; STD 2 Byte Code: (D1, C0)h; CVT 3 Byte Code: n/a
Method: ***** NOT CVT COMPLIANT *****
Per CEA-861 --- 1080p (Code 16) Timing Definition

Detailed Timing Parameters

Timing Name	= **1920 x 1080 @ 60Hz;**					
Hor Pixels	= **1920;**	// Pixels				
Ver Pixels	= **1080;**	// Lines				
Hor Frequency	= 67.500;	// kHz	=	14.8 usec	/ line	
Ver Frequency	= 60.000;	// Hz	=	16.7 msec	/ frame	
Pixel Clock	= **148.500;**	// MHz	=	6.7 nsec	± 0.5%	
Character Width	= **4;**	// Pixels	=	26.9 nsec		
Scan Type	= **NONINTERLACED;**			// H Phase =	1.4 %	
Hor Sync Polarity	= **POSITIVE**	// HBlank	=	12.7% of HTotal		
Ver Sync Polarity	= **POSITIVE**	// VBlank	=	4.0% of VTotal		
Hor Total Time	= 14.815;	// (usec)	=	550 chars =	2200 Pixels	
Hor Addr Time	= 12.929;	// (usec)	=	480 chars =	1920 Pixels	
Hor Blank Start	= 12.929;	// (usec)	=	480 chars =	1920 Pixels	
Hor Blank Time	= 1.886;	// (usec)	=	70 chars =	280 Pixels	
Hor Sync Start	= 13.522;	// (usec)	=	502 chars =	2008 Pixels	
// H Right Border	= 0.000;	// (usec)	=	**0** chars =	0 Pixels	
// H Front Porch	= 0.593;	// (usec)	=	**22** chars =	88 Pixels	
Hor Sync Time	= 0.296;	// (usec)	=	**11** chars =	44 Pixels	
// H Back Porch	= 0.997;	// (usec)	=	**37** chars =	148 Pixels	
// H Left Border	= 0.000;	// (usec)	=	**0** chars =	0 Pixels	
Ver Total Time	= 16.667;	// (msec)	=	1125 lines	HT – (1.06xHA)	
Ver Addr Time	= 16.000;	// (msec)	=	1080 lines	= 1.11	
Ver Blank Start	= 16.000;	// (msec)	=	1080 lines		
Ver Blank Time	= 0.667;	// (msec)	=	45 lines		
Ver Sync Start	= 16.059;	// (msec)	=	1084 lines		
// V Bottom Border	= 0.000;	// (msec)	=	**0** lines		
// V Front Porch	= 0.059;	// (msec)	=	**4** lines		
Ver Sync Time	= 0.074;	// (msec)	=	**5** lines		
// V Back Porch	= 0.533;	// (msec)	=	**36** lines		
// V Top Border	= 0.000;	// (msec)	=	**0** lines		

图 5.7-6 vesa 时序参数示例

按实验分辨率要求, 对应像素时钟频率数值如表 5.7-1 所示

表 5.7-1 像素时钟频率表

key_cnt	分辨率帧率	像素时钟频率（MHz）
1	1280*800P60	83.5
2	1360*768P60	85.5
3	1400*1050P60	101
4	1440*900P60	88.75
5	1600*900P60	108
6	1600*1200P60	162
7	1680*1050P60	119
8	1280*720P60	74.25
9	1920*1080P60	148.5
10	1920*1200P60	154

5.7.4.pattern_vg 模块

每个像素点的像素值数据, 对应每个像素点的颜色。常见的像素值表示格式比如: RGB888, RGB 分别代表: 红 R, 绿 G, 蓝 B, 888 是指 R、G、B 分别有 8bit, 即 R、G、B 每一色光有 28=256 级阶调, 通过 RGB 三色光的不同组合, 每个像素点最多可显示 24 位的 256*256*256=16,777,216 色。

根据行有效像素点参数，在 de 信号有效时输出彩条图像数据（RGB888），将一行数据分 8 等份，通过计数的方式在每个行像素位置范围输出固定的 RGB888 数据实现彩条的显示，如图 5.7-7 所示。

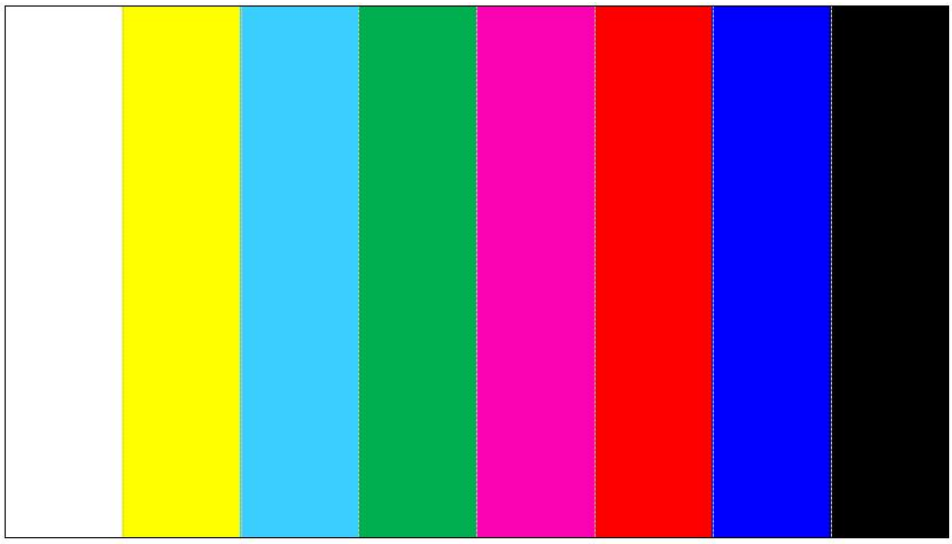


图 5.7-7 彩条显示效果示意图

5.7.5.pll 模块

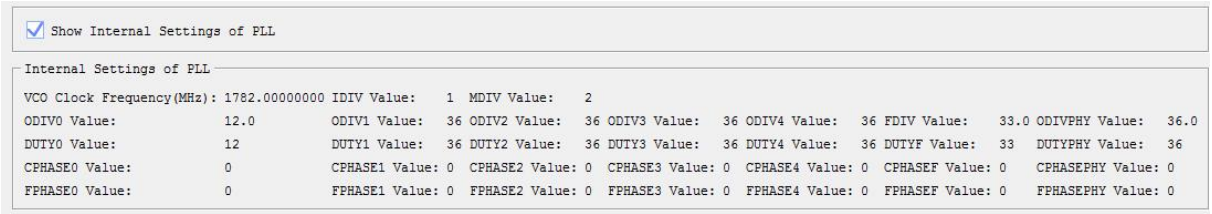
调用 PLL IP，根据 key_cnt 模块的按键计数结果，通过 APB 接口对 PLL 输出频率进行配置，输出对应频率的像素时钟，PLL 输入时钟采用 27MHz 晶振时钟，实现每种分辨率的像素时钟配置参数如表 5.7-2 所示

表 5.7-2 时钟配置参数表

key_cnt	分辨率	频率（MHz）	IDIV	MDIV	ODIV	FDIV
1	1280*800P60	83.5	1	2	22	34
2	1360*768P60	85.5	1	2	24	38
3	1400*1050P60	101	1	1	19	71
4	1440*900P60	88.75	1	1	21	69
5	1600*900P60	108	1	2	19	38
6	1600*1200P60	162	1	2	13	39
7	1680*1050P60	119	1	1	17	75
8	1280*720P60	74.25	1	2	24	36
9	1920*1080P60	148.5	1	2	12	36
10	1920*1200P60	154	1	1	10	57

5.7.6.apb_cfg 模块

实现 APB 接口时序，根据 key_cnt 模块的按键计数结果对应的显示分辨率，动态配置 PLL 输出时钟频率，且保持输出时钟信号的占空比和相位不变。具体时钟频率参数的设置，可在 IP 界面的设置参数，如输入时钟为 27MHz，输出时钟频率 148.5MHz 对应的参数如下：



通过 APB 接口对 PLL 的 IDIV_RATIO、MDIV_RATIO、ODIV0_RATIO、FDIV_RATIO、ODIV0_DUTY 和 FDIV_DUTY 寄存器进行写数据操作，完成对输出时钟信号的配置。

5.8. 实验效果

MES2L676-100HP-MINI 开发板 HDMI-OUT 接口连接显示屏,按下按键 key1 可切换彩条显示分辨率。

6. 基于 ADC 硬核读取内部电压及温度

6.1. 实验原理

Logos2 系列 FPGA 产品具有一个包括 2 个 12bit SAR-ADC 资源的模数转换模块,用户可以通过 ADC IP 根据需求配置 ADC, 调用 ADC 硬核资源, 读取 FPGA 内部电压及温度。

6.2. 接口列表

6.2.1.选择 IP

打开 IPC, 选择 ADC IP 路径, 在右侧页面设置 Instance Name 名称, 点击 Customize, 进入 ADC IP 参数配置页面, 如图 6.2-1 所示。

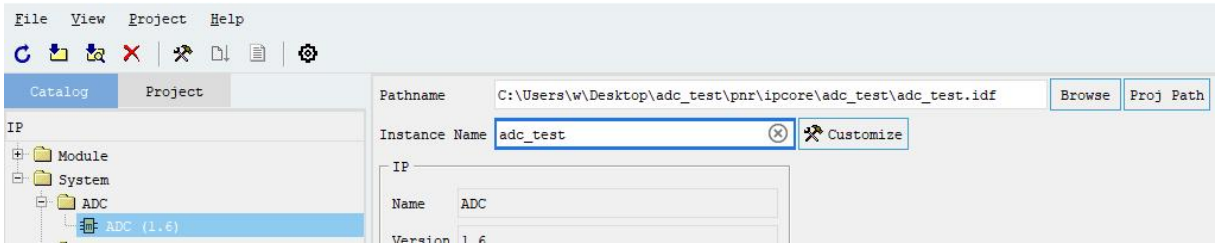


图 6.2-1 ADC IP 选择

6.2.2.配置 IP 参数

ADC IP 参数配置有三个页面, demo 中配置参数如下, 详情可查看 demo 例化 ADC IP, 详细参数配置描述请查看《UG041005_ADC_IP》。

ADC Config 是 ADC 参数配置页面, demo 参数页面如图 6.2-2 所示:

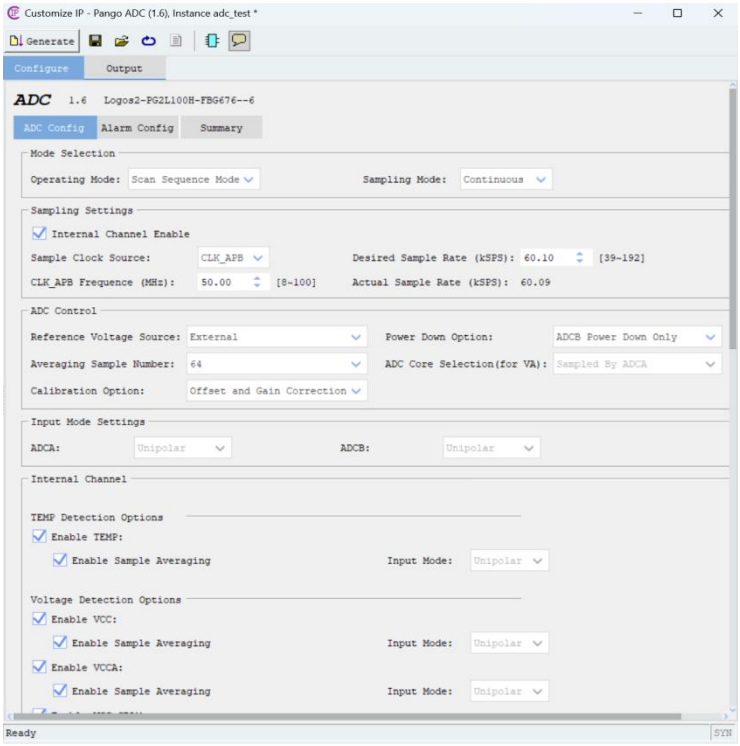


图 6.2-2 ADC IP 参数配置页面

Alarm Config 是温度、电压检测告警配置页面，demo 中未使用告警内容，页面如图 6.2-3 所示。

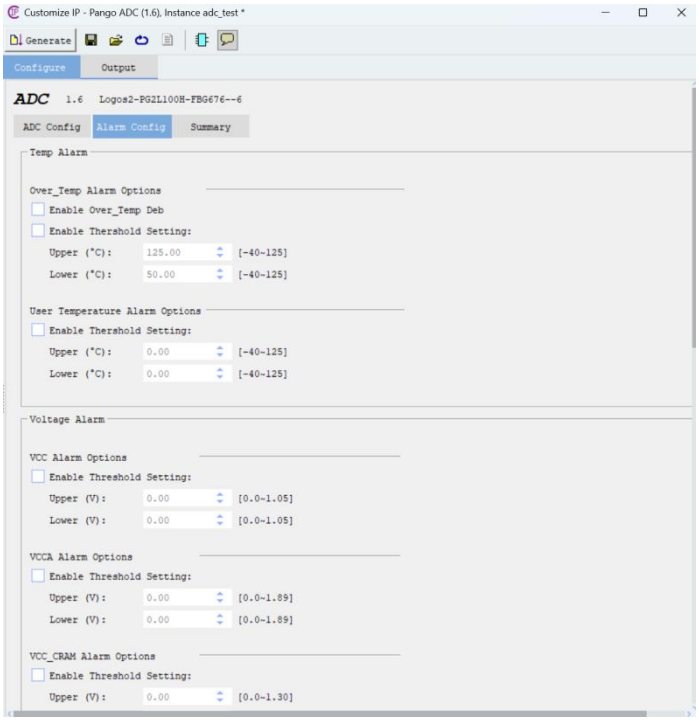


图 6.2-3 alarm config 页面

Summary 页面为打印配置信息，无需配置，页面如图 6.2-4 所示。

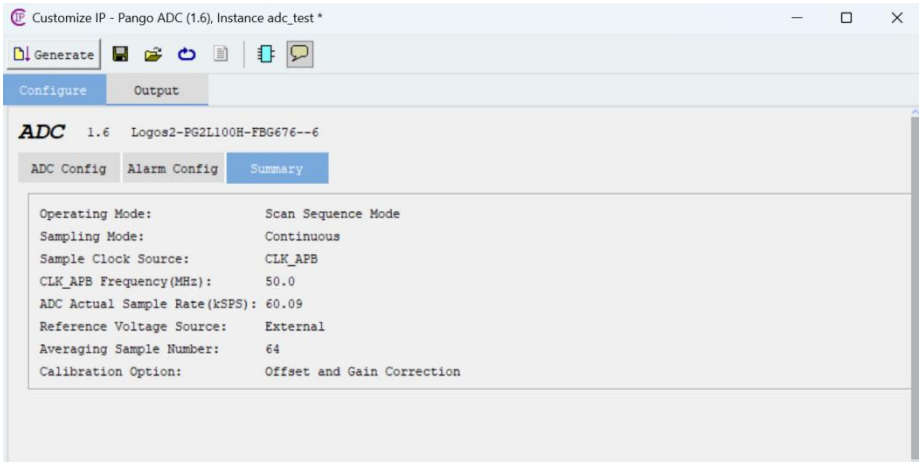


图 6.2-4 IP summary 界面

6.2.3.IP 接口描述

点击左上角 Generate, 会生成相应配置的 ADC IP, IP 接口如图 6.2-5 所示。

```
adc_test the_instance_name (
  .i_rst_n(i_rst_n),                    // input
  .i_apb_pclk(i_apb_pclk),              // input
  .i_apb_paddr(i_apb_paddr),            // input [7:0]
  .i_apb_psel(i_apb_psel),              // input
  .i_apb_enable(i_apb_enable),          // input
  .i_apb_pwrite(i_apb_pwrite),          // input
  .i_apb_pwdata(i_apb_pwdata),          // input [15:0]
  .o_apb_prdata(o_apb_prdata),          // output [15:0]
  .o_apb_pready(o_apb_pready),          // output
  .i_adc_loadsc_n(i_adc_loadsc_n),      // input
  .o_logic_done_a(o_logic_done_a),      // output
  .o_logic_done_b(o_logic_done_b),      // output
  .o_adc_clk_out(o_adc_clk_out),        // output
  .o_adc_dmodified(o_adc_dmodified)     // output
);
```

图 6.2-5 ADC IP 接口

demo 使用 ADC IP 接口说明如表 6.2-1 所示, 详细参数配置描述请查看《UG041005_ADC_IP》。

表 6.2-1 ADC IP 接口说明

端口	I/O	位宽	描述
i_rst_n	input	1	系统复位, 0 复位, 1 复位释放;
i_apb_pclk	input	1	APB 时钟, demo 输入 50MHz;
i_apb_paddr	input	8	apb 接口, 读写地址;
i_apb_psel	input	1	apb 接口, 片选信号, 0 未选, 1 选中;
i_apb_enable	input	1	apb 接口, 访问使能, 0 不使能, 1 使能;
i_apb_pwrite	input	1	apb 接口, 写操作使能, 0 读数据, 1 写数据;

i_apb_pwdata	input	16	apb 接口, 写数据;
o_apb_prdata	output	16	apb 接口, 读数据;
o_apb_pready	output	1	apb 接口, 读写 ready 信号; 写操作 0: 数据还未成功写入寄存器 1: 数据已成功写入寄存器 读操作 0: 读数据还未准备好 1: 读数据已经准备好
i_adc_loadsc_n	input	1	控制寄存器加载静态配置值使能信号 0 使能, 1 不使能;
o_logic_done_a	output	1	ADCA 状态寄存器更新指示信号, 0 未更新, 1 更新;
o_logic_done_b	output	1	ADCB 状态寄存器更新指示信号, 0 未更新, 1 更新;
o_adc_clk_out	output	1	传输给 SRB 的 ADC 工作时钟;
o_adc_dmodified	output	1	控制寄存器改动标志信号 0: APB 已操作过控制寄存器 1: Jtag-APB 写过后, 还未进行 APB 操作

6.3. 工程说明

6.3.1.模块设计

实验例程可用过串口转 APB 接口读取 ADC 输出码, 检测芯片温度, 检测芯片 VCC/VCCA/VCC_CRAM/VCC_DRM 电压。模块设计框图, 如图 6.3-1 所示, 主要包括 uart2apb、rst_n_sync、pll、adc_ip 模块。

- uart2apb 模块主要实现 uart 和 apb 接口之间的数据转换;
- rst_n_sync 模块主要实现复位信号的去抖、同步处理;
- pll 实现 50MHz 时钟输出;
- adc_ip 实现片内 ADC 模块的相关控制及状态访问;

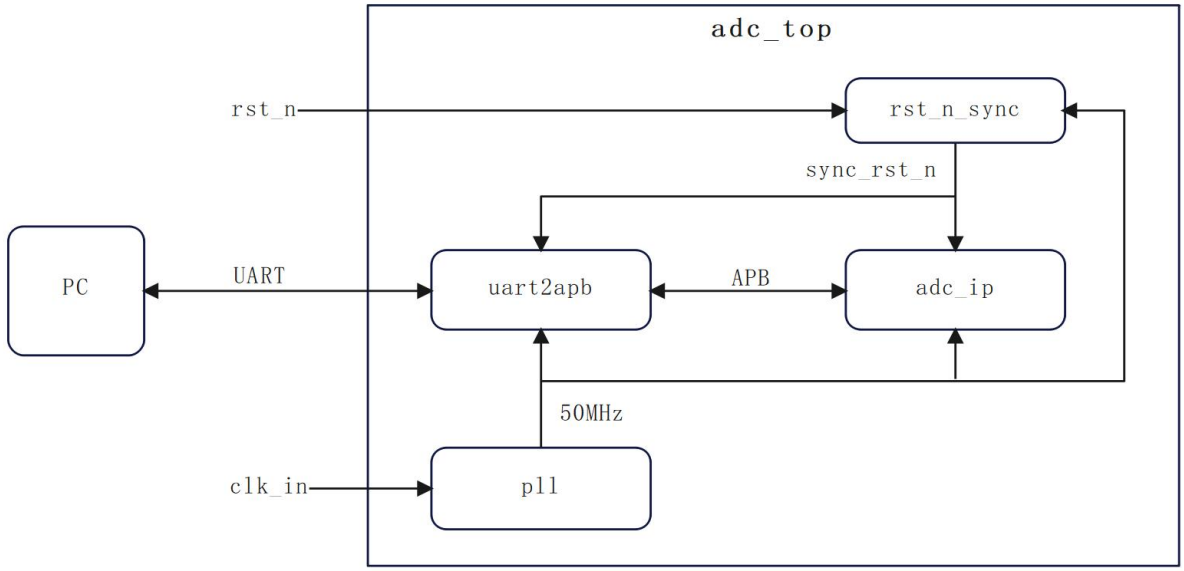


图 6.3-1 模块设计框图

顶层模块接口如图 6.3-2 所示, 相关描述见表 4-6-2。

```
module adc_top(
    input    wire        clk_in      ,
    input    wire        rst_n       ,

    output   reg         led         ,

    output   wire        cfg_uart_txd,
    input    wire        cfg_uart_rxd
);
```

图 6.3-2 顶层模块接口

表 6.3-1 图 顶层模块接口描述

端口	I/O	位宽	描述
clk_in	input	1	系统时钟输入, 27MHz;
rst_n	input	1	系统复位输入, 0 复位, 1 复位释放;
led	output	1	adc 状态指示灯;
cfg_uart_txd	output	1	向 pc 端发送串口数据端口;
cfg_uart_rxd	input	1	接受 pc 端串口数据端口;

6.3.2.状态寄存器说明

状态寄存器用来存放各通道的转换结果和用来校准的 Offset 和 Gain 的值。所有状态寄存器对用户只可读取, 不能进行写操作。

以下为 demo 使用的状态寄存器地址位定义,其他地址详情请见《UG040009_Logos2 系列 FPGA 模数转换模块（ADC）用户指南》。

表 6.3-2 demo 使用的状态寄存器

地址	默认值	描述
0x40	16'h0000	转换后的 temperature;
0x41	16'h0000	转换后的 VCC;
0x42	16'h0000	转换后的 VCCA;
0x43	16'h0000	转换后的 VCC_CRAM;
0x44	16'h0000	转换后的 VCC_DRM;

下图为 demo 使用的状态寄存器中有效数据说明:

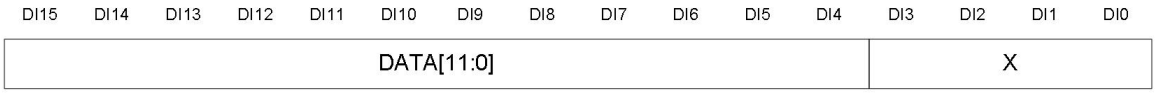


图 6.3-3 状态寄存器中有效数据

6.3.3.ADC 输出码转化公式说明

ADC 默认工作在 Unipolar 模式下, 温度和电压转换公式如下, 详情请查阅《UG040009_Logos2 系列 FPGA 模数转换模块 (ADC)用户指南》。

(1)ADC 输出码对应的片内温度转化公式为:

Temperature(℃)= ADC Code*0.1219-273.15

(2)ADC 输出码对应的电压转化公式为:

Voltage (V)=ADC Code/4096*3

6.3.4.实验现象

1) 将 demo 工程的 sbit 下载到 MES100HP 开发板中, LED1 闪烁, KEY1 为复位键;

2) 用 Type-C 连接开发板与电脑, 打开串口调试工具, 选择对应端口号、波特率设置为 115200, 数据格式为起始位 1、数据位 8、结束位 1、无校验位, HEX 显示、HEX 发送。

读操作格式为: "0x72" + "地址", 详情请查阅《UG040009_Logos2 系列 FPGA 模数转换模块 (ADC)用户指南》。

读取片内温度: 即读取地址 0x40 数据: "0x72" + "0x40", 为 0x7240;

读取 VCC: 即读取地址 0x41 数据: "0x72" + "0x41", 为 0x7241;

读取 VCCA: 即读取地址 0x42 数据: "0x72" + "0x42", 为 0x7242;

读取 VCC_CRAM: 即读取地址 0x43 数据: "0x72" + "0x43", 为 0x7243;

读取 VCC_DRM: 即读取地址 0x44 数据: "0x72" + "0x44", 为 0x7244;

实验现象如下图所示:

图 6.3-4 实验现象图

3） 将串口接收的数据，取有效位，转成十进制，如下表所示：

串口发送数据（16 进制）	串口接收数据（低位在前，16 进制）	寄存器有效数据（16 进制）	转成十进制
7240	A170	A17	2583
7241	55C0	55C	1372
7242	99A0	99A	2458
7243	6CB0	6CB	1739
7244	55C0	55C	1372

将转化的十进制值，根据转化公式计算，可得到对应温度或电压，也可填到 excel 表格中，即可得到对应温度或电压：

温度：	ADC 输出码对应的片内温度为：Temperature(℃)= ADC Code*0.1219–273.15		
寄存器地址	ADC 输出码	Temperature(℃)	备注
40h	2583	41.7177	
电压：	ADC 输出码对应的电压为：Voltage=ADC Code/4096*3V		
寄存器地址	ADC 输出码	Voltage（V）	备注
41h	1372	1.004882813	VCC
42h	2458	1.800292969	VCCA
43h	1739	1.273681641	VCC_CRAM
44h	1372	1.004882813	VCC_DRM

图 6.3-5 寄存器对应温度或电压

7. 基于 FPGA 双启动的远程升级模块

7.1. 实验原理

Master SPI 模式下，FPGA 与 Flash 之间总线为 SPI 接口，当 FPGA 处于配置模式时，该 SPI 接口由 FPGA 配置系统（Configuration Control System,CCS）控制；当 FPGA 进入用户模式后，该 SPI 接口默认为用户 IO，远程升级模块将这组信号重新配置为 SPI 接口，可更新 Flash 内位流文件，实现远程升级。

7.2. 接口列表

上位机可通过串口配置相关寄存器，下发控制命令及位流文件，实现远程升级。远程升级 remote_updata_top 模块系统框图如图 7.2-1 所示，右侧虚线框内为通信模块的主要功能为：与上位机通信、数据缓存、命令解析等。左侧红色实线框内为通用模块，主要控制读/写 flash 和热启动。

若远程升级方案中通信接口换为以太网等其它接口，左侧实线框内模块可以通用，右侧虚线框内模块可根据通信协议适配修改。

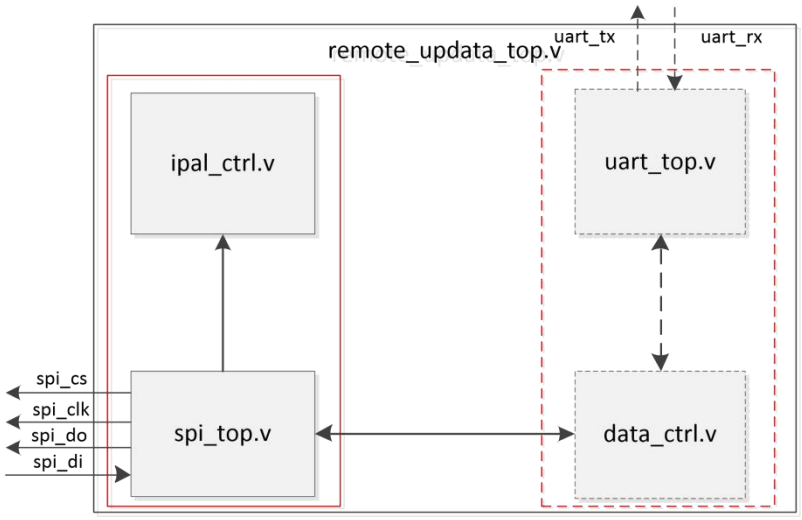


图 7.2-1 模块系统框图

(1) 串口模块(uart_top.v)

串口模块主要负责与上位机之间的串口通信，默认波特率为 115200bps。串口模块框图如图 7.2-2 所示：

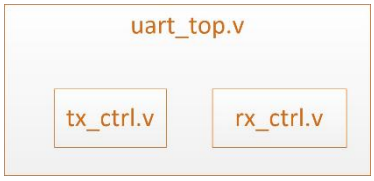


图 7.2-2 串口模块框图

(2) 数据控制模块(data_ctrl.v)

数据控制模块主要有两个功能：解析上位机下发的命令及读写寄存器控制；缓存位流数据，并填充数据至 4KB 对齐，方便后续处理。数据控制模块框图如图 7.2-3 所示：

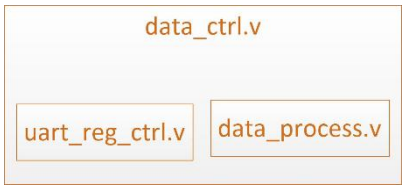


图 7.2-3 数据控制模块框图

(3) spi 控制模块(spi_top.v)

spi 控制模块完成 spi 总线控制，将位流数据写入外部 flash 芯片。spi_driver.v 模块负责完成单条指令(读配置寄存器、擦除扇区、页编程等)的执行。spi_top.v 模块将读/写位流命令分解成多个单条指令，写入 FIFO，再由 spi_driver.v 模块依次执行。spi 控制模块框图如图 7.2-4 所示：

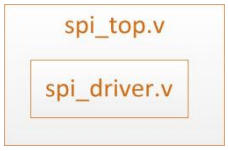


图 7.2-4 spi 控制模块框图

(4) IPAL 控制模块(ipal_ctrl.v)

IPAL 控制模块主要实现 FPGA 器件的热启动。位流数据写入完成后，IPAL 控制模块会收到一个指示信号，若 hotreset_en 为 1，则立即热启动；若 hotreset_en 为 0，则不会立即热启动，可以通过上位机控制将此寄存器写为 1，然后热启动。

7.2.1.顶层模块参数定义及接口信号列表

顶层模块的参数定义如表 5-4-1 所示，接口定义如表 5-4-2 所示：

表 7.2-1 参数定义

信号名	描述
FPGA_VESION	FPGA 版本信息，用户可自定义，默认为年、月、日、时、分。
DEVICE	器件型号，用于判断位流占用 flash 空间大小。定义需与实际器件匹配，否则会导致功能异常。若后续推出新器件，用户可扩展。
USER_BITSTREAM_CNT	应用位流个数，可支持 1/2/3 个应用位流。定义需与实际情况匹配，否则会导致功能异常。
USER_BITSTREAM1_ADDR	应用位流 1 的起始地址，在黄金位流之后，且 4KB 对齐。
USER_BITSTREAM2_ADDR	应用位流 2 的起始地址，在应用位流 1 之后，且 4KB 对齐。
USER_BITSTREAM3_ADDR	应用位流 3 的起始地址，在应用位流 2 之后，且 4KB 对齐。

表 7.2-2 顶层模块接口信号列表

信号名	方向	位宽	描述
pin_clk_in	input	1	晶振输入时钟；
sys_rst_n	input	1	系统复位，低有效；

uart_rx	input	1	与上位机通信, 串口数据接收端口;
uart_tx	output	1	与上位机通信, 串口数据发送端口;
spi_cs	output	1	spi flash 芯片片选管脚;
spi_dq1	input	1	spi flash 芯片数据输入管脚
spi_dq0	output	1	spi flash 芯片数据/指令输出管脚
led	output	8	led 灯

7.2.2.通用模块接口列表

通用模块(spi_top.v 和 ipal_ctrl.v)与前级模块(data_ctrl.v)的接口信号主要有使能控制信号、反馈指示信号、读/写数据交互信号。前级模块根据流程和反馈指示信号, 对使能信号进行控制。写数据接口需要使用包长 256 字节的包 FIFO 存储数据,回读数据接口使用普通 FIFO 缓存数据。通用模块接口列表如下表 5-4-3 所示:

表 7.2-3 通用模块接口列表

信号名	方向	位宽	描述
时钟、复位信号(spi_top.v 和ipal_ctrl.v)			
sys_clk	input	1	系统时钟。
sys_rst_n	input	1	系统复位, 低有效。
外部管脚信号(spi_top.v)			
spi_cs	output	1	spi flash 芯片片选管脚。
spi_clk	output	1	spi flash 芯片时钟管脚。
spi_dq1	input	1	spi flash 芯片数据输入管脚。
spi_dq0	output	1	spi flash 芯片数据/指令输出管脚。
使能控制信号(spi_top.v)			
flash_wr_en	input	1	写位流数据使能, 上升沿有效。
flash_rd_en	input	1	读位流数据使能, 上升沿有效。
bitstream_wr_num	input	2	写位流序号, 用于指定更新的应用位流。可支持 1/2/3 号应用位流, 且不超过参数 USER_BITSTREAM_CNT。
bitstream_rd_num	input	2	读位流序号, 用于指定读取的应用位流。可支持 1/2/3 号应用位流, 且不超过参数 USER_BITSTREAM_CNT。
bitstream_up2cpu_en	input	1	位流上传上位机使能,高有效。使能后,回读校验时上传位流。
crc_check_en	input	1	CRC32 校验使能, 高有效。若不使能, 则不进行回读校验。

clear_sw_en	input	1	单独擦除开关使能, 上升沿有效。
bs_crc32_ok	input	2	[1]:为1 则表示校验结果有效 [0]:校验结果, 1'b0:校验正确, 1'b1:校验错误。
write_sw_code_en	input	1	写开关使能, 上升沿有效。
反馈指示信号(spi_top.v)			
bs_readback_crc	output	32	读位流校验 CRC 结果。
bs_readback_crc_valid	output	1	为1 表示读位流校验 CRC 结果有效。
clear_sw_done	output	1	单独擦除开关完成指示, 高有效。
clear_bs_done	output	1	擦除应用位流完成指示, 高有效。
bitstream_wr_done	output	1	写位流文件完成, 高有效。
bitstream_rd_done	output	1	读位流文件完成, 高有效。
open_sw_code_done	output	1	写开关程序完成, 高有效。
time_out_reg	output	1	擦除flash 超时指示, 高有效。
回读数据接口(spi_top.v)			
flash_rd_data	output	8	读位流数据。
flash_rd_valid	output	1	读位流数据有效。
flash_rd_data_fifo_afull	input	1	读位流数据缓存 FIFO 将满。
写flash 数据接口(前级数据缓存需要用包 FIFO)(spi_top.v)			
bitstream_fifo_rd_req	output	1	写入flash 位流文件缓存 FIFO 读请求。
bitstream_data	input	8	写入flash 位流文件缓存 FIFO 读出数据。
bitstream_valid	input	1	写入flash 位流文件缓存 FIFO 读出数据有效。
bitstream_eop	input	1	写入flash 位流文件缓存 FIFO 数据包尾标识。每个数据包 256 字节(1 个 page), 方便后续处理。
bitstream_fifo_rd_rdy	input	1	写入flash 位流文件缓存 FIFO 非空。
ipal 控制模块接口(ipal_ctrl.v)			
hotreset_en	input	1	热启动使能, 上升沿有效。
open_sw_num	input	2	热启动应用位流序号, 可支持 1/2/3 号应用位流, 且不超过参数 USER_BITSTREAM_CNT。

crc_check_en	input	1	CRC32 校验使能, 高有效。若不使能, 则不判断 CRC 校验结果。
bs_crc32_ok	input	2	[1]:为1 则表示校验结果有效 [0]:校验结果, 1'b0:校验正确, 1'b1:校验错误。
open_sw_code_done	input	1	写开关程序完成, 高有效。
ipal_busy	output	1	ipal 忙指示信号, 送到寄存器模块, 防止本模块被综合工具优化。

7.3. 工程说明

7.3.1.寄存器说明

本参考工程使用 MES100HP 开发板, 上位机使用串口助手工具, 完成远程升级功能。由于上位机与开发板之间用串口通信的, 没有区分寄存器地址与数据, 所以自定义一套命令码。

读/写操作都是上位机下发的命令, 读/写时上位机下发 32'he7e7_e7e7+地址命令码。地址命令码为一个字节, 低 7 位为地址, 以最高位区分读/写, 1'b1 表示读, 1'b0 表示写, 如表 5-4-4 所示。示例: 读寄存器 1, 32'he7e7_e7e7+8'h81; 写寄存器 1, 32'he7e7_e7e7+8'h01+写入的数据(1 字节)。

表 7.3-1 命令码

上位机下发命令码	上传上位机命令码	写数据流完成命令码	读地址命令码	写地址命令码
32'he7e7_e7e7	8'h55	32'h7e7e_7e7e	{1'b1,addr[6:0]}	{1'b0,addr[6:0]}

读寄存器命令下发, FPGA 会返回对应的寄存器值。擦除完成、写位流文件完成和 CRC 校验完成的信息, 不用读操作, FPGA 会主动上传, 格式相同。返回数据为 1 至多个字节, 根据寄存器而定。

7.3.2.寄存器地址分配

表 7.3-2 寄存器地址分配表

名称	地址	读/写	功能描述
fpga_version	7'h0	R	版本信息, 6 字节, 自定义
crc32_cfg	7'h1	W/R	上位机计算位流文件 CRC, 然后配置到寄存器, 4 字节。
test_reg	7'h2	W/R	测试寄存器
crc32_error_ind	7'h3	R	读位流 CRC 校验结果与 crc32_cfg 不相等标志。1'b1:不相等。
hotreset_en	7'h4	W/R	hotreset_en[0]: 热启动使能, 高有效。若有效, 更新位流完成后则热启动。
wr_bs_status	7'h5	R	wr_bs_status[0]: 擦除完成标识, 高有效; wr_bs_status[1]: 单独擦除开关完成标识, 高有效; wr_bs_status[2]: 单独打开完成标识, 高有效;

			wr_bs_status[3]: 擦除超时, 高有效; wr_bs_status[4]: 写位流文件完成标识, 高有效。
crc_check_en	7'h6	W/R	crc_check_en[0]: CRC 校验使能, 高有效。 1'b1:回读位流时进行 CRC 校验; 1'b0:回读位流时不进行 CRC 校验。 注意: 不使能时可以不要回读的步骤。
user_bitstream_cnt	7'h7	R	当前版本支持应用位流个数。由 USER_BITSTREAM_CNT 参数确定, 有效参数值有 2'd1、2'd2、2'd3。
bs_readback_crc	7'h8 7'h9 7'ha 7'hb	R	回读位流 CRC 校验结果, 32 位。低地址对应低字节。
bitstream_up2cpu_en	7'hc	W/R	bitstream_up2cpu_en[0]: 回读位流上传使能。 1'b1, 回读时, 将位流传回上位机。 1'b0, 回读时, 位流不传回上位机。
bitstream_num	7'hd	R	bitstream_num[1:0]:读应用位流号; bitstream_num[3:2]:写应用位流号。 有效值有 2'd1、2'd2、2'd3, 且不超过 user_bitstream_cnt。
open_clear_sw_en	7'he	W/R	open_clear_sw_en[1:0]: 打开开关程序的位流序号, 可选 1/2/3, 且不超过user_bitstream_cnt。 open_clear_sw_en[6]: 打开开关程序使能。 open_clear_sw_en[4]: 单独擦除开关程序使能。 打开开关时, 配置 0x41/0x42/0x43。单独擦除开关时, 配置 0x10。
wr_user_bs_en	7'h11 7'h12 7'h13	W	更新应用位流文件命令。wr_user_bs_en[1:0]对应 bitstream_wr_num, 更新的应用位流序号, 且不超过 user_bitstream_cnt。若 user_bitstream_cnt 为 2'd2, 可更新 1 号或 2 号位流, 写入 7'h11 或7'h12。
rd_user_bs_en	7'h51 7'	W	读应用位流文件命令。rd_user_bs_en[1:0]对应 bitstream_rd_num, 读回的应用位流序号, 且不超过

	h52		user_bitstream_cnt。
	7 ’ h53		

版本信息寄存器 fpga_version 和 CRC 寄存器 crc32_cfg 分别有 6 个字节和 4 个字节,读这两个寄存器时会一次返回全部字节,写 crc32_cfg 时也是一次配置全部字节(e7 e7 e7 e7+01+xx xx xx xx)。其它寄存器都只有一个字节，读/写只有一个字节内容。

返回数据格式为：8’h55+地址+返回数据。例：读版本信息收到数据 55 00 20 20 01 01 12 30，其中，55 为上传命令码，00 为地址，20 20 01 01 12 30 为版本信息寄存器的值

7.3.3.位流文件存储格式

位流文件在 flash 芯片内的存储格式如图 7.3-1 所示。

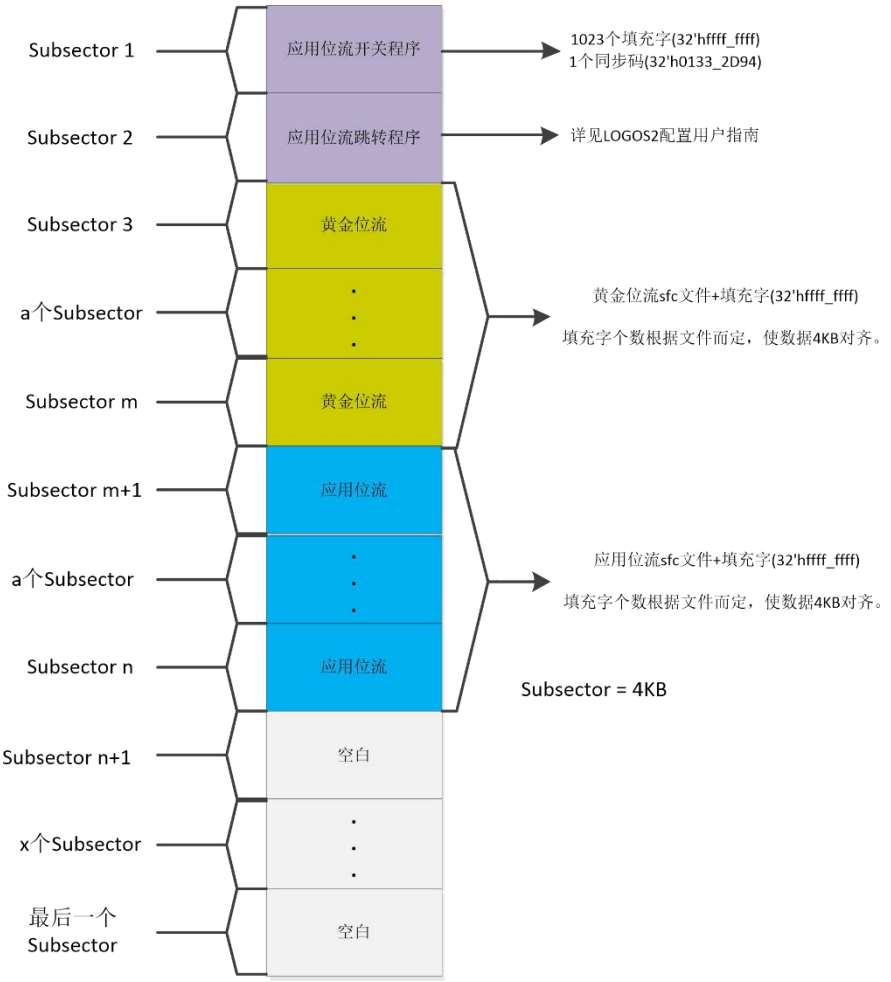


图 7.3-1 位流文件在 flash 中的存储格式

合并位流文件在 flash 中是以 4KB(一个 Subsector)划分的。第一个 4KB 为开关程序，为 1023 个 32’hffff_ffff+1 个同步码(32’h0133_2d94)。第二个 4KB 是应用位流跳转程序。每个应用位流都有一个开关程序和一个跳转程序。黄金位流必须在最后一个跳转程序之后，不满 4KB 的部分填充 32’hffff_ffff。应用位流在黄金位流之后，不满 4KB 的部分填充 32’hffff_ffff。黄金位流和应用位流的起始位置可以在生成合并位流文件时设置，但黄金位流必须在应用位流前，且必须保证各位流之间不

会出现地址重叠。

包含一个应用位流时，开关程序默认打开，即默认启动应用位流。包含多个应用位流时，默认打开开关程序 1，其它开关程序关闭，即默认启动第 1 个应用位流。

7.3.4.位流文件更新流程

远程升级时，更新位流的流程如图 7.3-2 所示。更新的过程，需要上位机与 FPGA 配合完成，其中写位流使能、读位流使能、单独擦除开关使能、打开开关使能、热启动使能由上位机下发命令，其它步骤由 FPGA 完成。升级过程中的命令及返回值如表 5-4-6 所示。

表 7.3-3 操作命令及返回值

名称	命令	返回完成标志
写位流 1 使能	e7 e7 e7 e7 11	55 05 01(擦除完成) 55 05 10(写位流完成) 55 05 08(擦除超时)
写位流 2 使能	e7 e7 e7 e7 12	
写位流 3 使能	e7 e7 e7 e7 13	
读位流 1 使能	e7 e7 e7 e7 51	55 03 01(校验错误) 55 03 00(校验正确)
读位流 2 使能	e7 e7 e7 e7 52	
读位流 3 使能	e7 e7 e7 e7 53	
单独擦除开关使能	e7 e7 e7 e7 0e 10	55 05 02(单独擦除完成) 55 05 08(擦除超时)
打开开关 1 使能	e7 e7 e7 e7 0e 41	55 05 04(打开开关完成)
打开开关 2 使能	e7 e7 e7 e7 0e 42	
打开开关 3 使能	e7 e7 e7 e7 0e 43	
热启动使能	e7 e7 e7 e7 04 01	无返回

升级的具体操作步骤如下：

(1)上位机下发写位流使能命令，等待 FPGA 擦除开关程序和应用位流。擦除完成 FPGA 发送给上位机完成标志(55 05 01)。

(2)上位机收到擦除完成标志后，发送位流文件和位流结束标志(7e 7e 7e 7e)。上位机收到(55 05 10)则表示写应用位流完成，可进行下一步操作。可以读位流进行校验，也可以重复前两步，再次写位流文件。

(3)上位机下发读位流使能，读取位流进行校验。校验完成后 FPGA 上报校验结果(55 03 01/00)。(55 03 01)表示校验结果为错误，(55 03 00)表示校验结果为正确。

(4)上位机发送打开开关使能命令。打开完成后，FPGA 发送给上位机完成标志(55 05 04)。

(5)上位机下发热启动使能，加载新的应用位流。若上步的校验结果为错误，则加载黄金位流。

若不更新位流，只切换应用位流启动，其操作步骤如下：

(1)上位机下发单独擦除开关使能命令，擦除所有开关程序。擦除完成 FPGA 发送给上位机完成标

志(55 05 02)。

(2)上位机发送打开开关使能命令。打开完成后, FPGA 发送给上位机完成标志(55 05 04)。

上位机下发热启动使能, 加载新的应用位流。为了简化切换位流启动的流程, 可关闭校验使能。

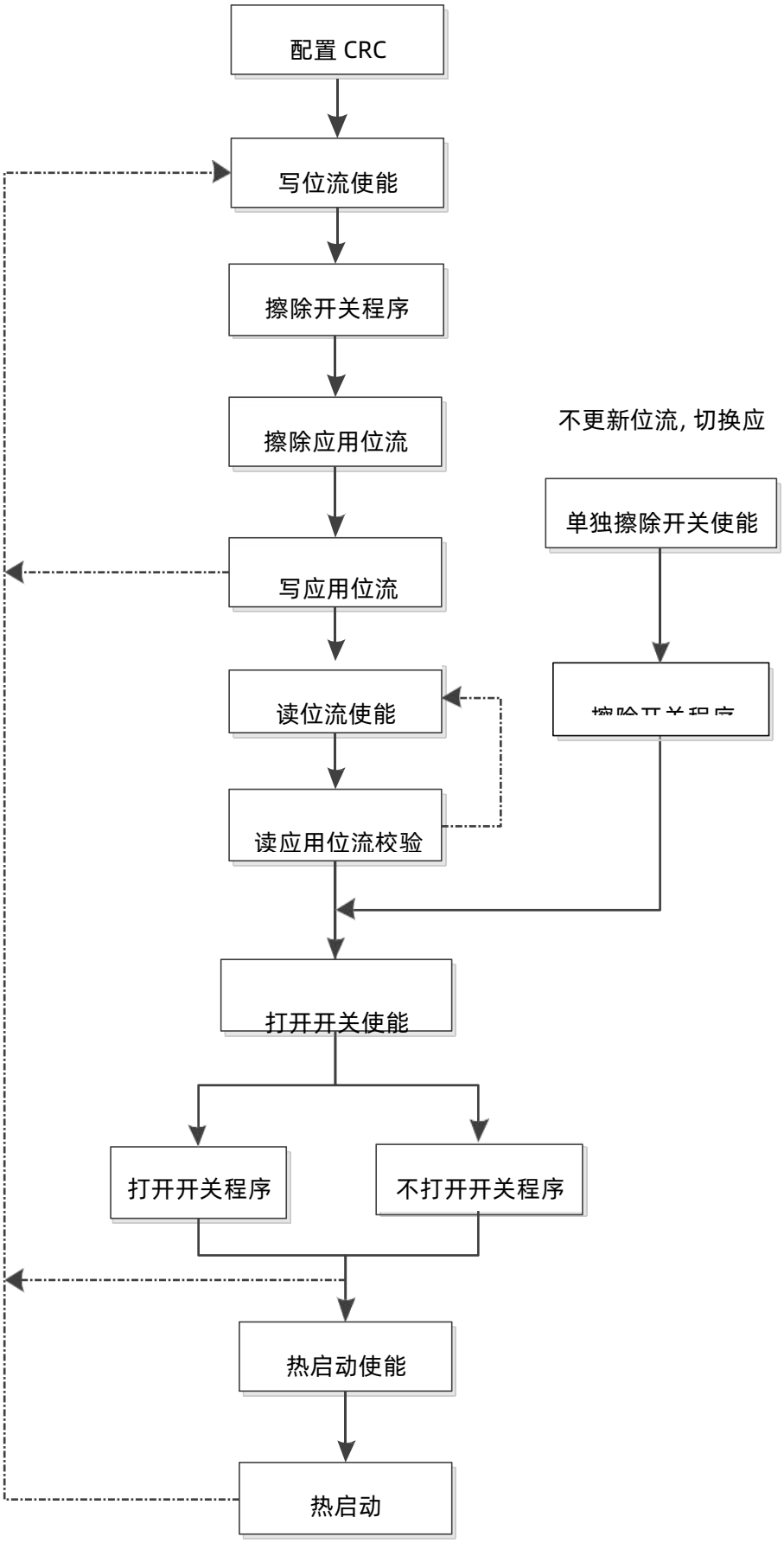

配置 CRC
写位流使能
擦除开关程序
擦除应用位流
不更新位流, 切换应
单独擦除开关使能
写应用位流
读位流使能
读应用位流校验
打开开关使能
打开开关程序
不打开开关程序
热启动使能
热启动

图 7.3-2 应用位流更新流程

8. 设计保护

8.1. 实验原理

Logos2 系列 FPGA 具备位流加密的功能,加密位流文件加载时,必须结合存储在 FPGA 内部的密钥进行解密,只有加密位流文件和密钥匹配时,FPGA 才能正常进入用户模式,否则无法正常工作。

8.2. Encryption 界面介绍

在 PDS 菜单栏【Project】打开【Project Setting】选择【Generate Bitstream】加密设置界面【Encryption】,界面如下图 5-5-1 所示。

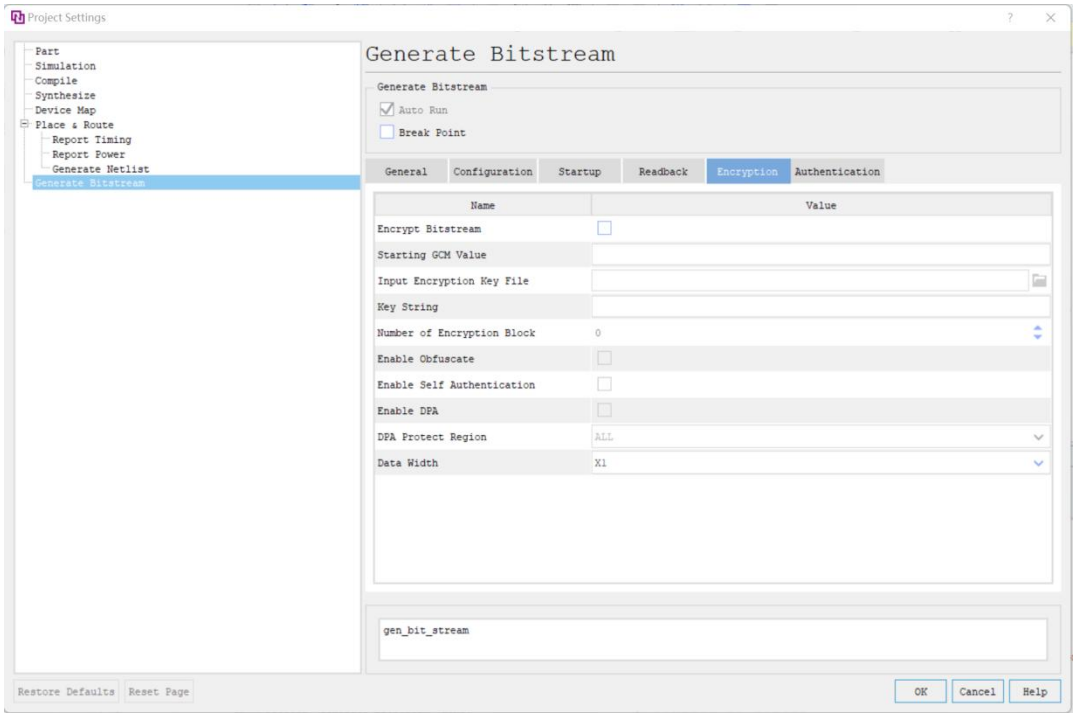


图 5-5-1 Encryption 设置界面

【Encrypt Bitstream】勾选是否加密位流文件,勾选则生成加密的 sbit 文件及 nky 文件(包含加密所需的 96 比特的初始 GCM 值和 256 比特的密钥),不勾选则生成未加密的 sbit

文件, 默认值为不勾选。用户可以编辑 CBC 和密钥字符串或选择 nky 文件, 若用户不指定, 则随机生成 nky 文件;

【Starting GCM Value】用户手动输入初始的 GCM 字符串, 字符为 16 进制, 输入长度为最多 96bit, 默认为空;

【Input Encryption Key File】选择 nky 文件, 如果在此选择了密钥文件, 则软件选择以密钥文件进行加密, 而不管用户在选项中是否输入了密钥字符串, 默认为空;

【Key String】用户手动输入密钥字符串, 字符为 16 进制字符, 输入长度为最多 256bit, 默认为空;

【Number of Encryption Block】分块加密数据的块数, 字符为 10 进制字符, 输入范围为 0~max_number (max_number 在不同的器件中取值不同, PG2L100H:58; PG2L200H:136; PG2L50H:32; PG2L25H:18; PG2T390H:201), 默认值为 0;

【Enable Obfuscate】使能混淆加密, 与加密功能配合使用, 默认不使能

【Enable Self Authentication】使能自认证功能, 对加密数据进行自认证, 默认不使能;

【Enable DPA】使能 DPA 保护功能, 与加密功能配合使用, 默认不使能;

【DPA Protect Region】设置 DPA 保护区域, 默认为 ALL;

【Data Width】加密数据宽度设置, 配置加密数据时使用的数据位宽, 默认值为 X1;

8.3. 位流加密保护演示流程

(注意: 此流程仅烧录临时密钥寄存器, 展示位流加密效果, 请严格按照流程操作, 不当操作将会导致芯片锁死, 无法正常使用, 请自行承担相应责任和后果, 此软件版本仅作测试演示, 正常烧录加密请使用 Pango Design Suite 2022.2-SP6.5 及以上版本)

(1)生成非加密位流文件

正常生成位流, 不设置【Project Setting】中【Generate Bitstream】相关内容, 在此不在赘述。非加密位流可直接下载, 实验现象为流水灯。

(2)生成秘钥 nky 文件及加密 sbit 位流文件

在 PDS 菜单栏【Project】打开【Project Setting】选择【Generate Bitstream】加密设置界面【Encryption】, 勾选【Encrypt Bitstream】后, 用户可以编辑初始 GCM 字符串和 Key String 密钥字符串, 若用户不设置, 则随机生成 nky 文件;

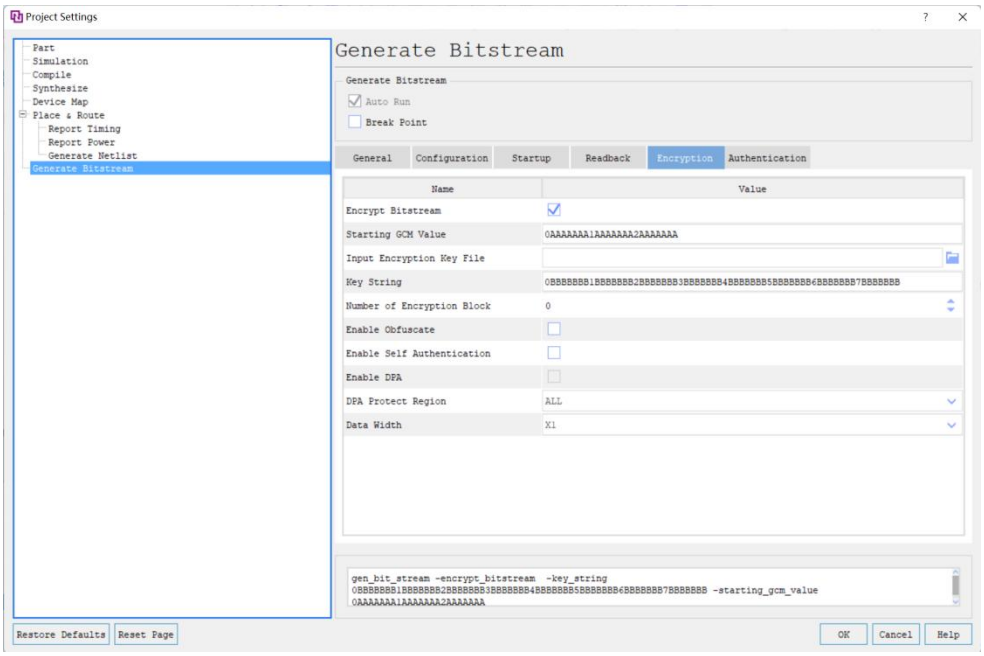


图 5-5-2 Encryption 设置界面

【Encryption】界面设置完成后, 点击 OK, 【Generate Bitstream】右键选择 Rerun 重新

生成 sbit 位流文件, 如图 5-5-3 所示。

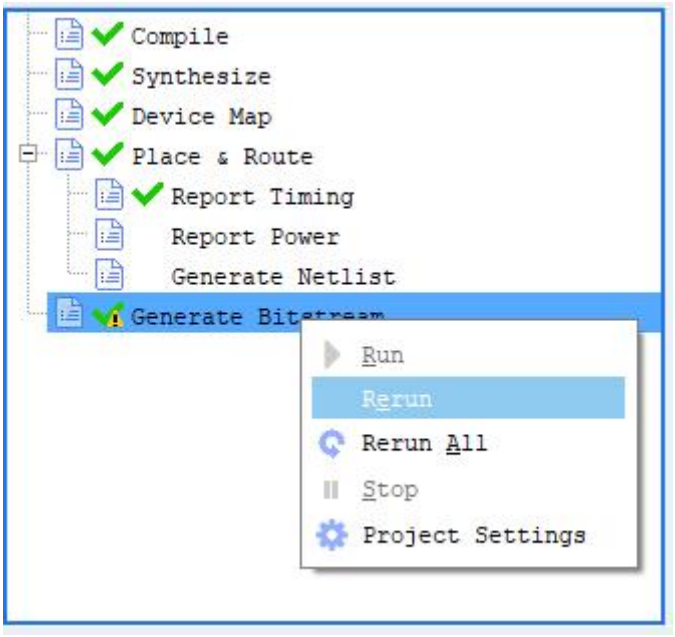


图 5-5-3

重新生成 sbit 位流文件后,在【generate_bitstream】文件夹中生成.nky 秘钥文件及对应加密 sbit 位流文件, 如图 5-5-4 所示;

(注意: 用户需妥善保管秘钥 nky 文件)

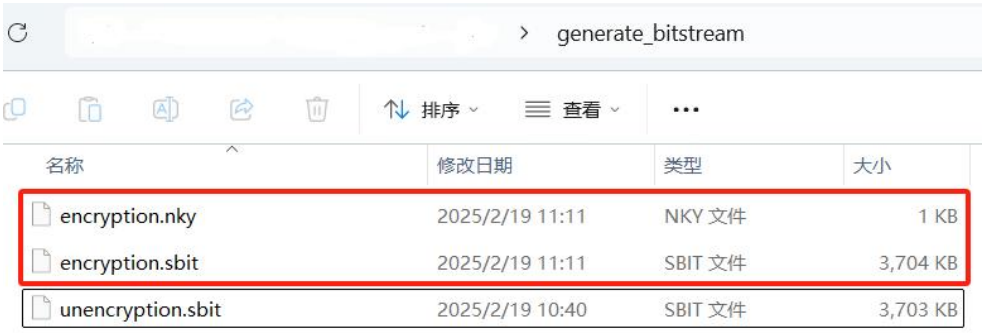


图 5-5-4

(3)直接下载加密位流

在 PDS 的【Fabric Configuration】选择加密 sbit 位流直接下载, 软件告警提示 sbit 位流文件为加密位流文件, 下载报错, 加密位流无法正常工作。表示只有加密位流文件和密钥

匹配时才能正常工作。

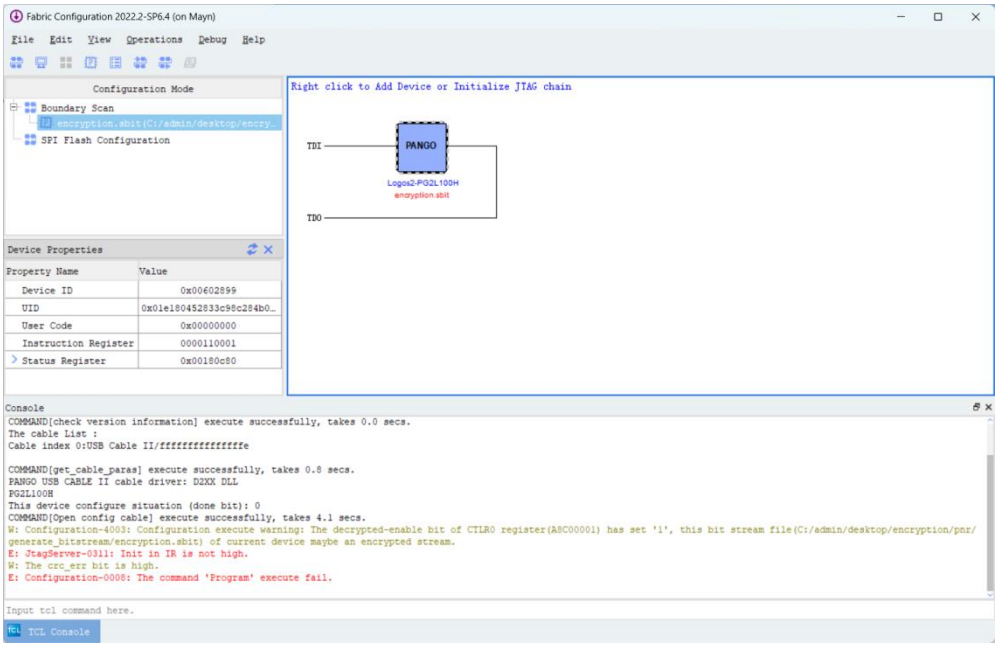


图 5-5-5 加密 sbit 位流直接下载, 无法正常工作

(4)烧写秘钥 nky 文件后, 下载加密位流

在 PDS 的【Fabric Configuration】右键器件选择菜单中【Operate Key and eFUSE】, 如图 5-5-6 所示。

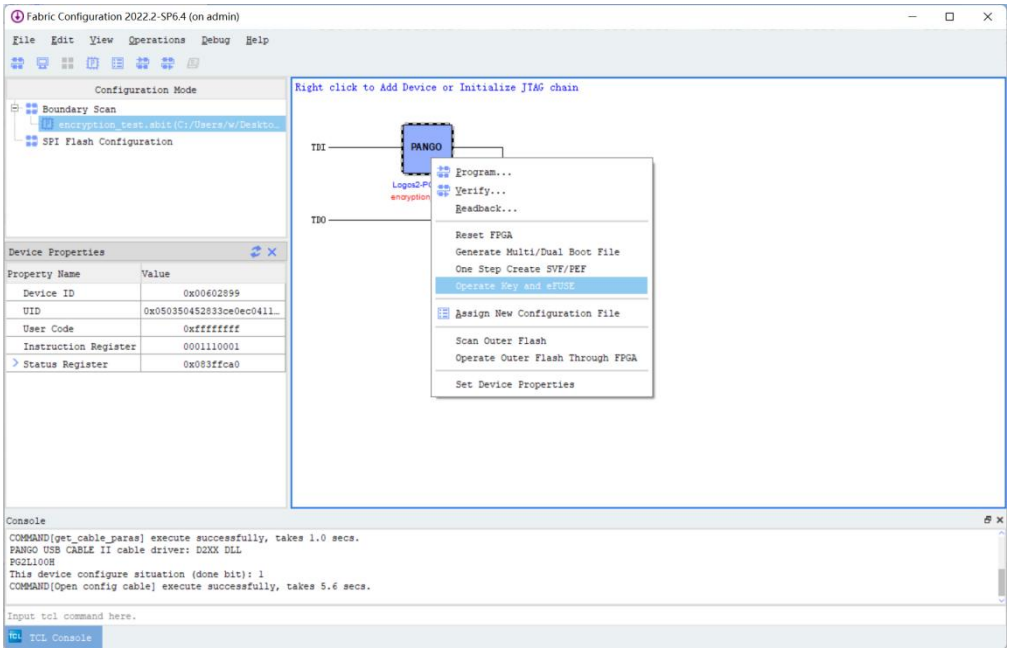


图 5-5-6

【Program eFuse Registers】界面，选择【Program Key File】操作类型，点击【Next】，进入选择秘钥窗口，如图 5-5-7 所示。

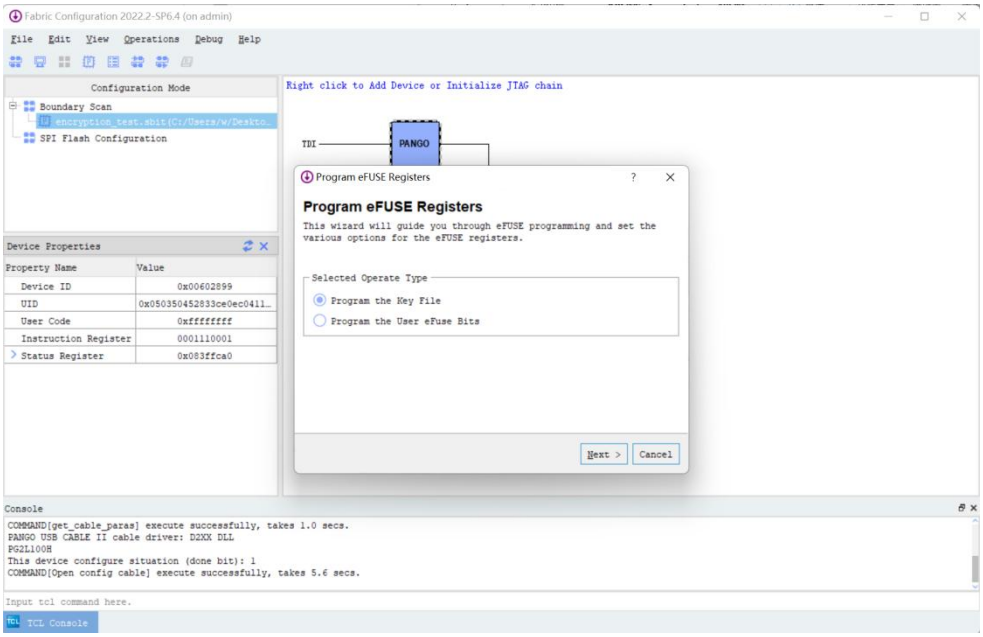


图 5-5-7 选择操作 eFuse 类型

【Configure Key File】界面，在文件输入框选择将要烧录的 nky 文件，软件会解析该秘钥文件与当前所选中的器件是否匹配，如果匹配将会把秘钥信息打印出来，如图 5-5-8 所示，选择好秘钥文件后，点击【Next】。

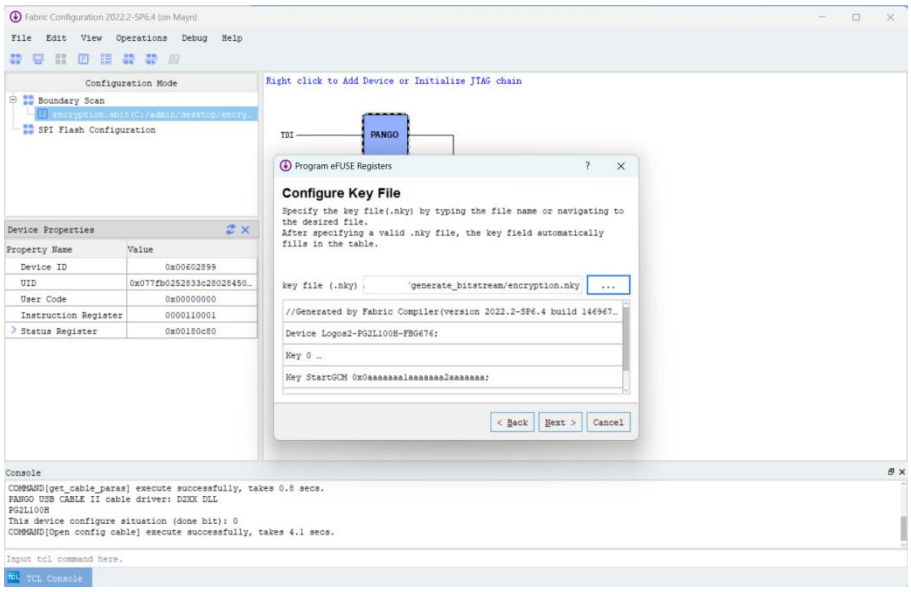


图 5-5-8 选择秘钥 nky 文件

【Set Key Location】界面，选择秘钥 nky 烧写位置【Temporary Key Register】临时密钥寄存器，如图 5-5-9 所示。

（注意：当前软件版本请勿选择【eFUSE Register】和【BBRAM】，不当操作会导致芯片锁死，无法正常使用，正常烧录加密请使用 Pango Design Suite 2022.2-SP6.5 及以上版本）；

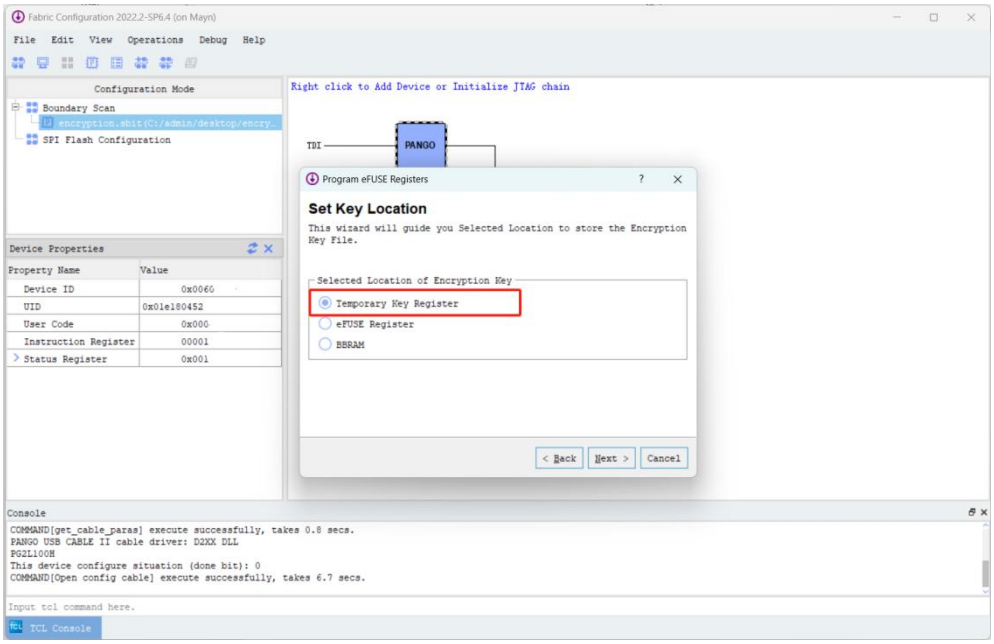


图 5-5-9 选择秘钥 nky 烧写位置【Temporary Key Register】

在 Summary 页将会把所选择的操作类型及秘钥文件信息打印出来，供用户进行确认，如图 5-5-10 所示。确认信息无误后，点击 Finish 即可进行烧写，烧写成功提示如图 5-5-11 所示。

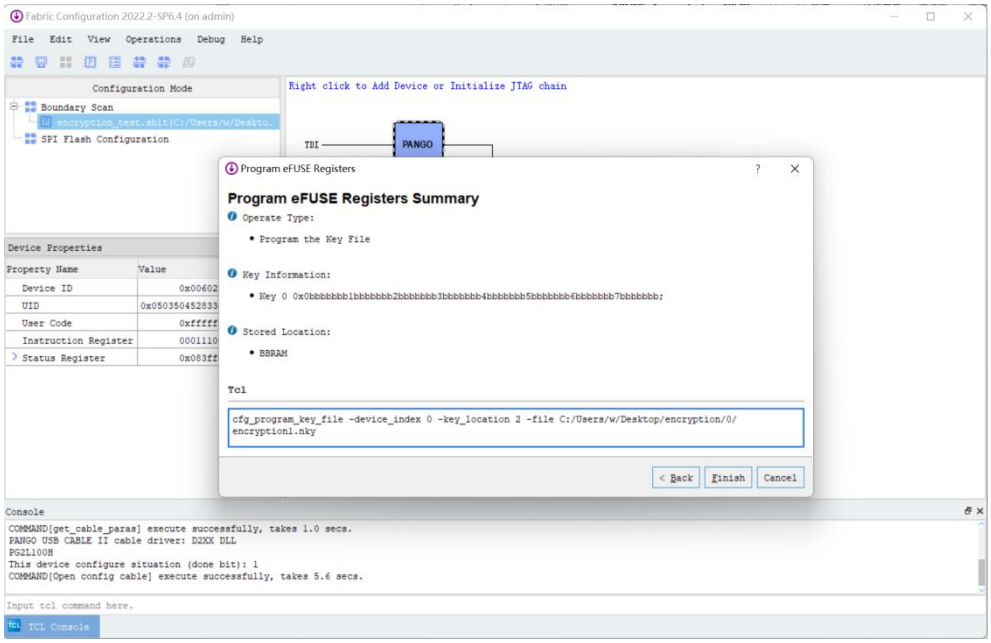


图 5-5-10 配置秘钥信息

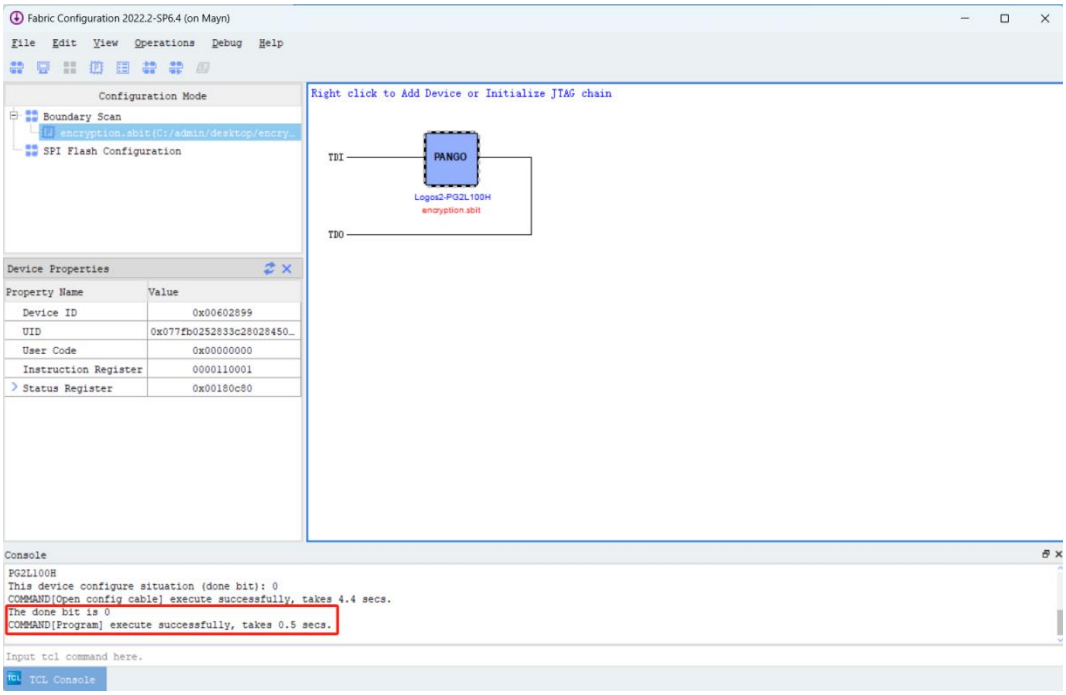


图 5-5-11 秘钥烧写成功

把秘钥成功烧写到 FPGA 后, 下载加密 sbit 位流, 软件告警提示 sbit 位流文件为加密位流文件, 成功下载, 加密位流可正常工作（实验现象为流水灯）。表示只有加密位流文件和密钥匹配时才能正常工作。

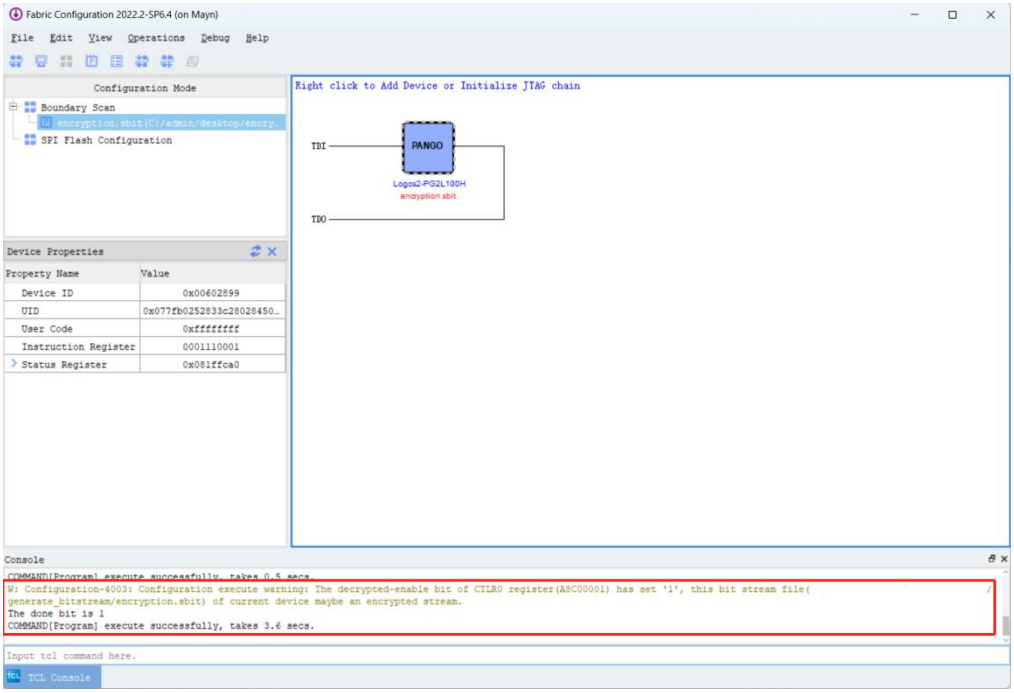


图 5-5-12

9. PDS 时序约束

9.1. 实验原理

本章节以 DDR3 IP 应用实验 demo 为例,在熟悉 PDS 软件基本开发流程的前提下,进行时序约束和在线调试, 确保时序不违例, 并且使用 debugger 工具捕获视频输入信号验证视频输入信号是否正常。

实验 demo 框架如下图所示, FPGA 实现 HDMI_IN 数据接收后经 DDR3 缓存, 最后将图像数据通过 HDMI 输出显示, 在此工程上进行时序约束和在线调试工具的使用。

图 9.1-1 实验 demo 框架

9.2. 接口列表

本实验采用两片 MT41K256M16TW-107:P, 总数据位宽为 32bit, DDR3 相关端口如下表所示:

端口名	I/O	位宽	说明
MEM_ROW_ADDR_WIDTH	\	\	DDR 行地址位宽
MEM_COL_ADDR_WIDTH	\	\	DDR 列地址位宽
MEM_BADDR_ADDR_WIDTH	\	\	DDR BANK 地址位宽
MEM_DQ_WIDTH	\	\	DDR 数据位宽
MEM_DQS_WIDTH	\	\	DDR 数据选通位宽

sys_clk	input	1	系统时钟输入(27MHZ)
clk_p/clkn	input	1	系统差分时钟输入(125MHZ)
mem_rst_n	output	1	DDR3复位信号低电平有效
mem_ck	output	1	DDR3输入系统时钟
mem_ck_n	output	1	DDR3输入系统时钟
mem_cke	output	11	时钟使能信号。高有效
mem_cs_n	output	1	DDR3片选信号
mem_ras_n	output	1	行地址选通信号
mem_cas_n	output	1	列地址选通信号
mem_we_n	output	1	DDR3写操作信号

mem_odt	output	1	DDR 3ODT
mem_a[MEM_ROW_ADDR_WIDTH-1:0]	output	15	DDR3行列地址总线
mem_ba[MEM_BADDR_WIDTH-1:0]	output	3	DDR3 BANK 地址
mem_dqs[MEM_DQ_WIDTH/8-1:0]	inout	4	DDR3 数据选取信号
mem_dqs_n[MEM_DQ_WIDTH/8-1:0]	inout	4	DDR3 数据选取信号
mem_dq[MEM_DQ_WIDTH-1:0]	inout	32	DDR3 数据
mem_dm[MEM_DQ_WIDTH/8-1:0]	output	4	DDR3 数据掩码
heart_beat_led	output	1	DDR3时钟心跳信号
ddr_init_done	output	1	DDR3初始化完成信号
init_over_rx	Output	1	HDMI 配置成功信号

HDMI 输出相关端口

HDMI 相关接口主要包含 IIC 寄存器配置接口以及 HDMI 数据输出接口

端口名	I/O	说明
init_over_rx	output	HDMI 配置成功信号
pixclk_in	input	HDMI 输入的像素时钟
vs_in	input	HDMI 输入的场同步信号
hs_in	input	HDMI 输入的行同步信号
de_in	input	HDMI 输入的像素有效信号
r_in[7:0]	input	HDMI 输入的像素的红色分量
g_in[7:0]	input	HDMI 输入的像素的绿色分量
b_in[7:0]	input	HDMI 输入的像素的蓝色分量
pix_clk	output	HDMI 输出的像素时钟
vs_out	output	HDMI 输出的场同步信号
hs_out	output	HDMI 输出的行同步信号
de_out	output	HDMI 输出的像素有效信号
r_out[7:0]	output	HDMI 输出的像素的红色分量

g_out[7:0]	output	HDMI 输出的像素的绿色分量
b_out[7:0]	output	HDMI 输出的像素的蓝色分量
hd_scl	output	HDMI 输出芯片的 SCL 信号
hd_sda	inout	HDMI 输出芯片的 SDA 信号
rstn_out	output	1ms 复位信号
hdmi_init_led	output	HDMI 输出芯片寄存器初始化完成信号

9.3. 工程说明

9.3.1.时序约束方法

工程中的时钟信号使用如下图所示, demo 中使用外部 27MHz 晶振输入时钟经 PLL 后输出两路时钟 clk_25M 用于 HDMI phy 芯片配置模块,pix_clk 用作 HDMI 输出时钟。pixclk_in 为 HDMI 输入像素时钟。DDR3 IP 采用外部 125MHz 时钟信号作为参考时钟, 输出根据 IP 设置输出 DDR3 差分时钟 men_ck 和 men_ck_n, 并且输出 core_clk。

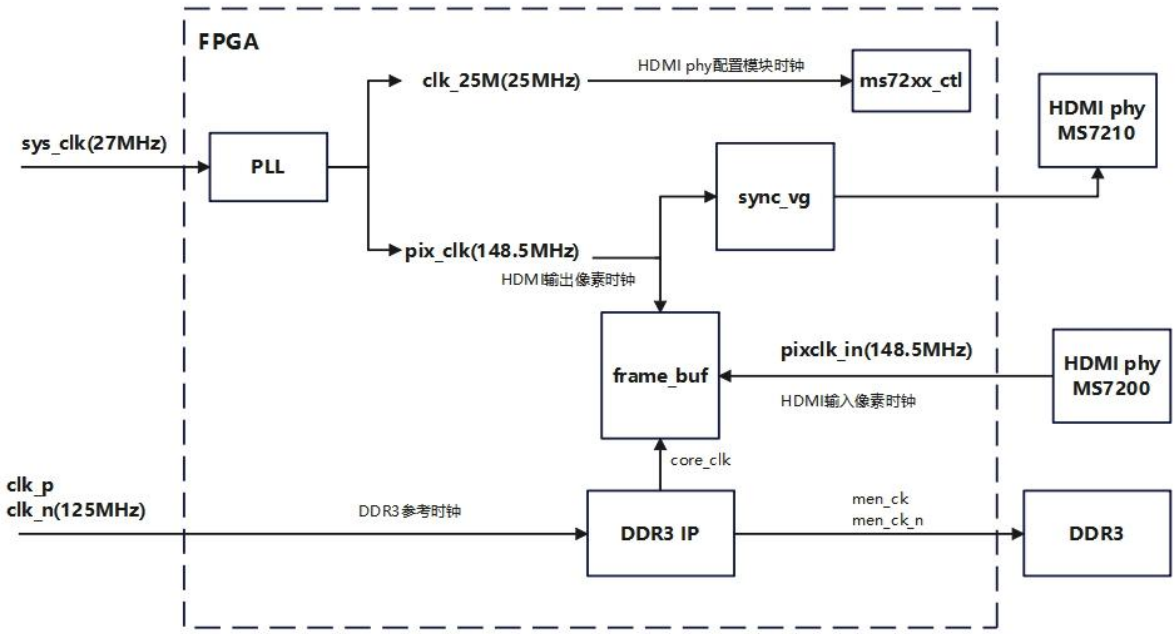


图 9.3-1 工程时钟信号使用详情

在信号处理过程中, 数据跨时钟域传输应做同步处理, 在本实验 demo 中图像数据跨时钟域传输已使用双端口 RAM 做同步处理。

工程中时序约束工具可使用 User Constraint Editor 工具, 对应约束的具体操作流程请参考《User_Constraint_Editor_User_Guide.pdf》

1、Create Clock 命令用于创建基准时钟, 通常指工程所使用外部输入时钟, 其他约束中可能涉及基准时钟需先创建基准时钟。创建基准时钟约束如下图所示

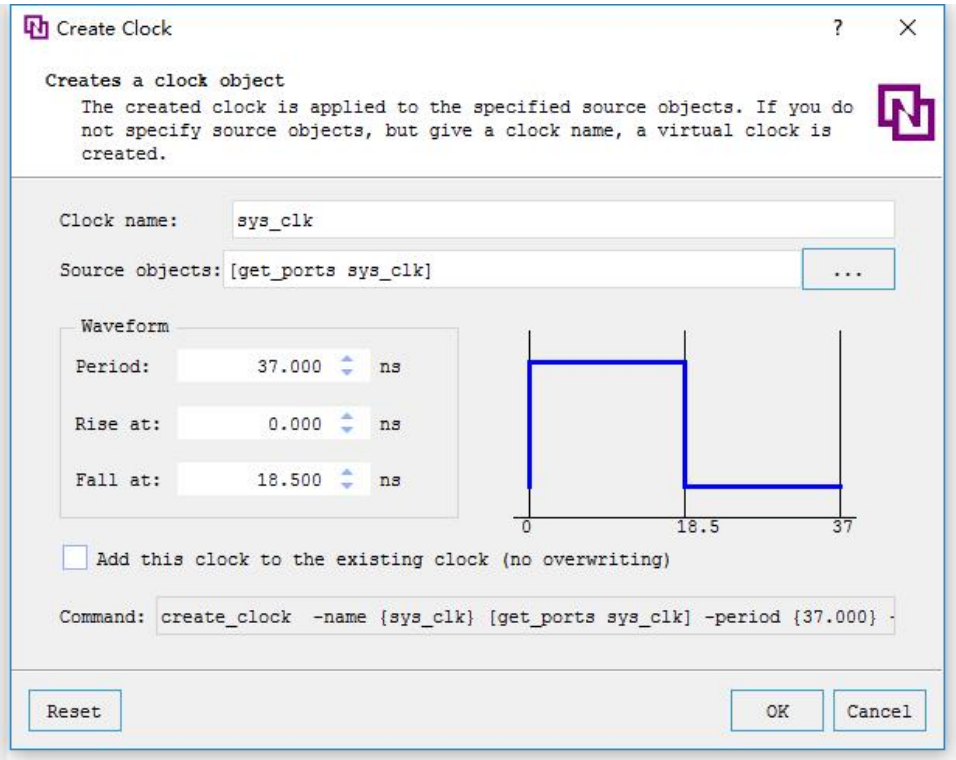


图 9.3-2 创建基准时钟约束

通过界面设置自动生成对应约束语句如下:

create_clock -name {sys_clk} [get_ports sys_clk] -period {37.000} -waveform {0.000 18.500}

在本实验中基准时钟主要包括外部输入 27MHz 时钟 sys_clk, HDMI 输入像素时钟 pixclk_in, 约束每个时钟对应的时钟周期, 时钟上升沿和下降沿在一个时钟周期内的位置。

2、create_generated_clock 命令用于创建生成时钟, 生成时钟在设计内部被 pll 或某些用户逻辑所驱动。生成时钟和主时钟相关联, 主时钟可以是基准时钟或者是其他的生成时钟。

相较于直接指定生成时钟的周期或者波形属性, 生成时钟的属性由主时钟变换而来, 只需要描述相对于主时钟的变换关系。 主时钟和生成时钟之间的关系可以是下列这些情况分

频、倍频、相位移动、波形反转和占空比转换以及他们的任意组合

生成时钟约束如下图所示

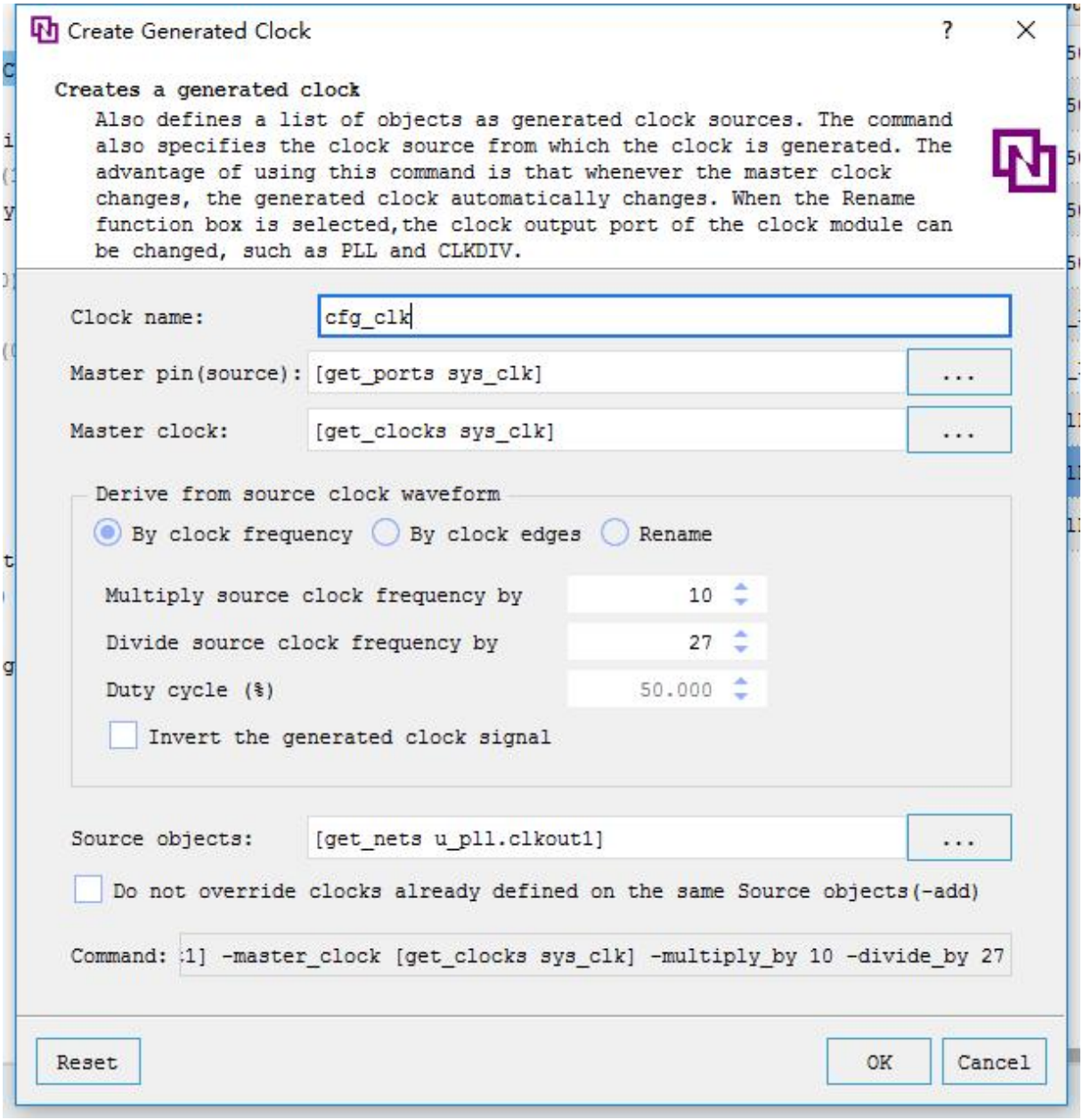


图 9.3-3 生成时钟约束

通过界面设置生成对应约束语句如下:

create_generated_clock -name cfg_clk -source [get_ports sys_clk] [get_nets u_pll.clkout1] -master_clock [get_clocks sys_clk] -multiply_by 10 -divide_by 27

cfg_clk 由 sys_clk 输入 PLL 调频后的输出时钟, 因此 master clock 是 sys_clk, sys_clk 频率为 27MHz, cfg_clk 频率为 10MHz, 所以 Multiply 倍频参数设置为 10, Divide 分频参数设置为 27。

3、set_clock_groups 命令用于设置时钟组, 设置时钟组可以不进行不同时钟组之间的时序分析, 同时对时钟组内的分析没有影响。例如在 demo 工程中数据传输在不同的阶段使用不同时钟, 在跨时钟域的数据传输过程需设计跨时钟域同步电路, 且在设置中将不同时钟域进行时钟组约束, 不进行不同时钟组之间的时序分析, 如下图所示

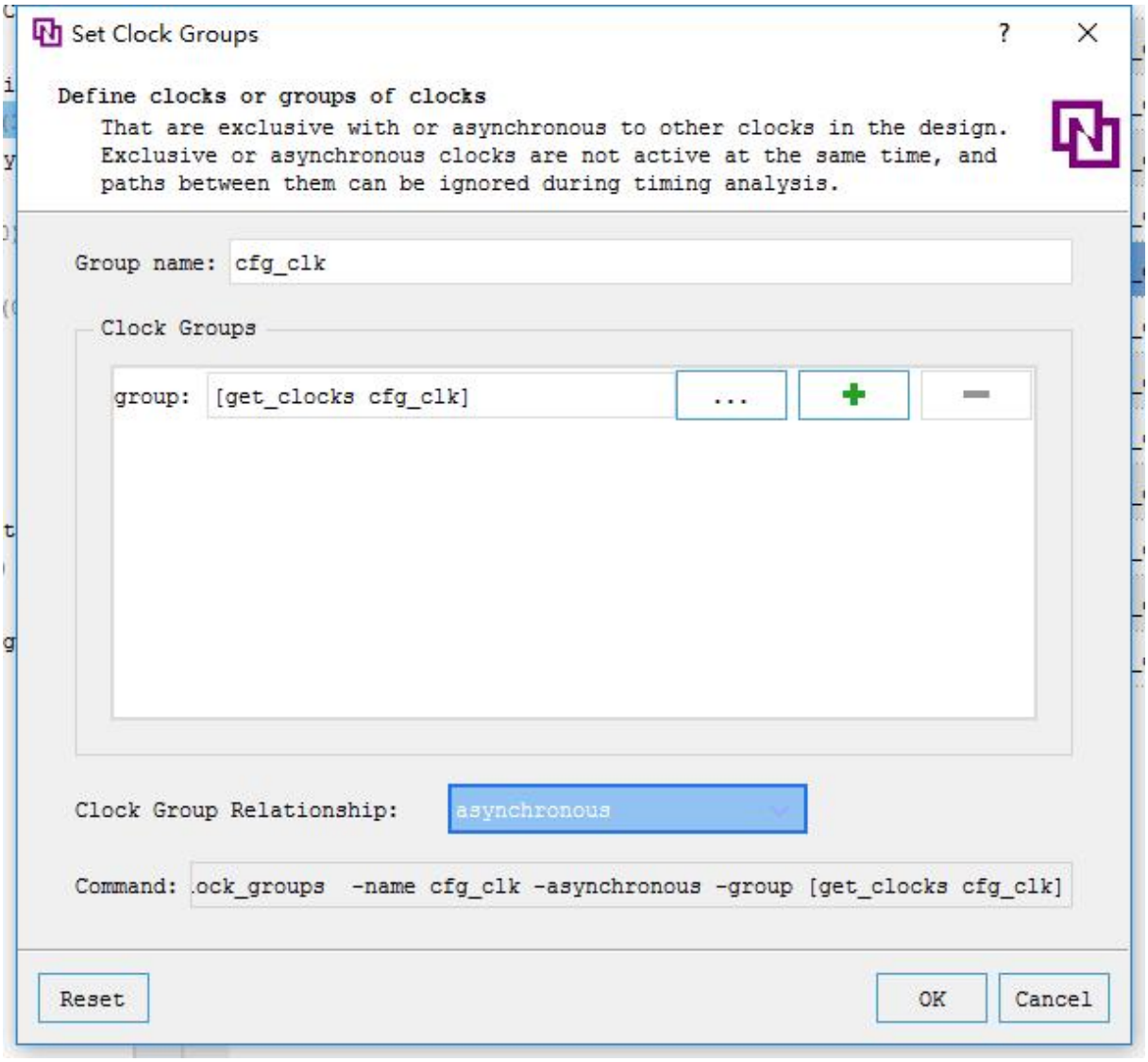


图 9.3-4 时钟组约束

4、DDR3 IP 相关时序约束

ref_clk 是输入参考时钟, ddrphy_sysclk 和 rst_clk 是 GPLL 倍频得到, phy_dq_sysclk 是 PPLL 倍频得到。ddrphy_sysclk 是 IP 软逻辑的系统时钟, rst_clk 是 GPLL 动态移相和复位序列的驱动时钟, phy_dq_sysclk 是 BANK 时钟, 每个 BANK 有一个 PPLL, 该 BANK 的 phy_dq_sysclk 由该 BANK 的 PPLL 分频得到。IP 内部时钟结构如下图所示

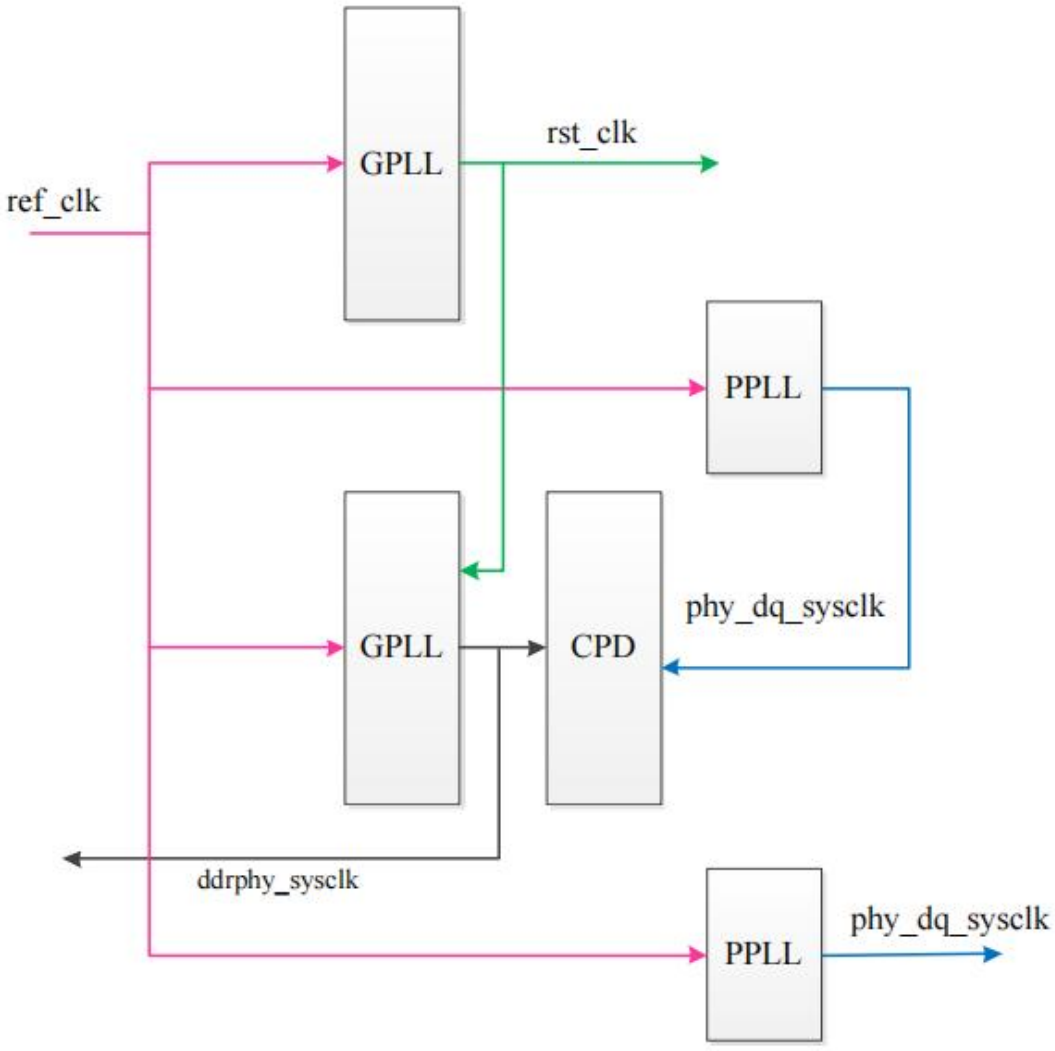


图 9.3-5 DDR3 IP 内部时钟结构

DDR3 IP 相关具体约束方法可参考 IP example 工程 IP /pnr 目录下的.fdc 文件

9.3.2.时序约束报告解读

在生成比特流前, 有一个步骤是 Report Timing, 会生成时序报告, 如下所示:

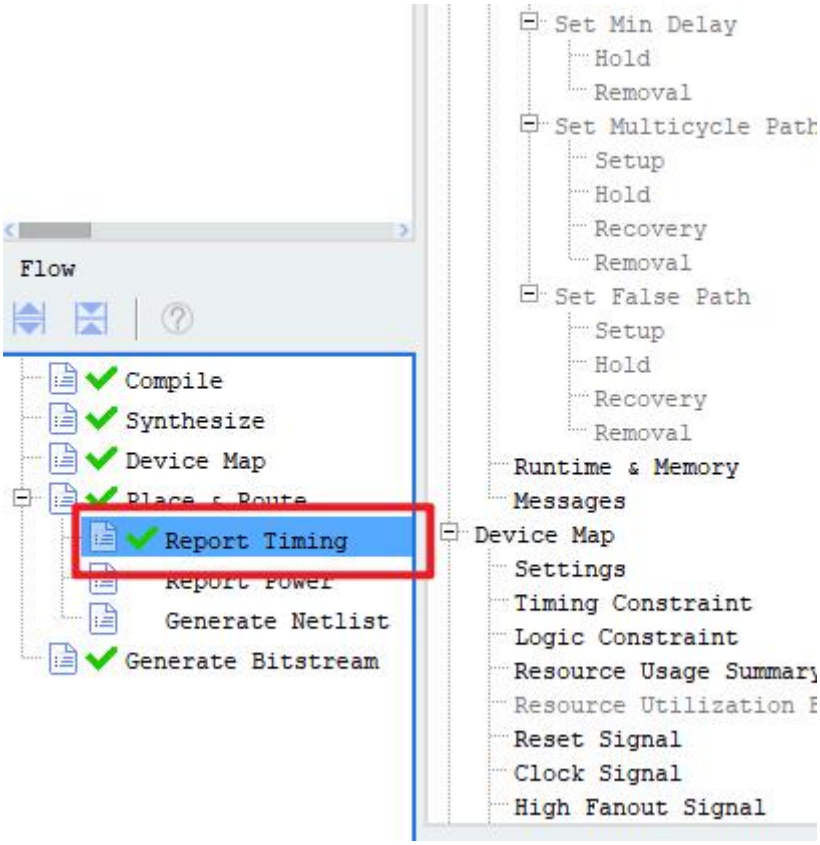


图 9-6

打开 Report Sumary 后可以看到 Report Timing 的报告。如果没出现红色则表示工程的时序没有出现违例。Clock Group 表示是否有设置异步时钟分组，即 asynchronous。设置异步时钟分组的好处是可以标识不同时钟域之间的时序关系，避免工具时序分析误报。比如 sys_clk 和 ref_clk 两者无同步关系，设置异步分组后，就可以避免工具去分析这两者的时序关系，避免工具误报。

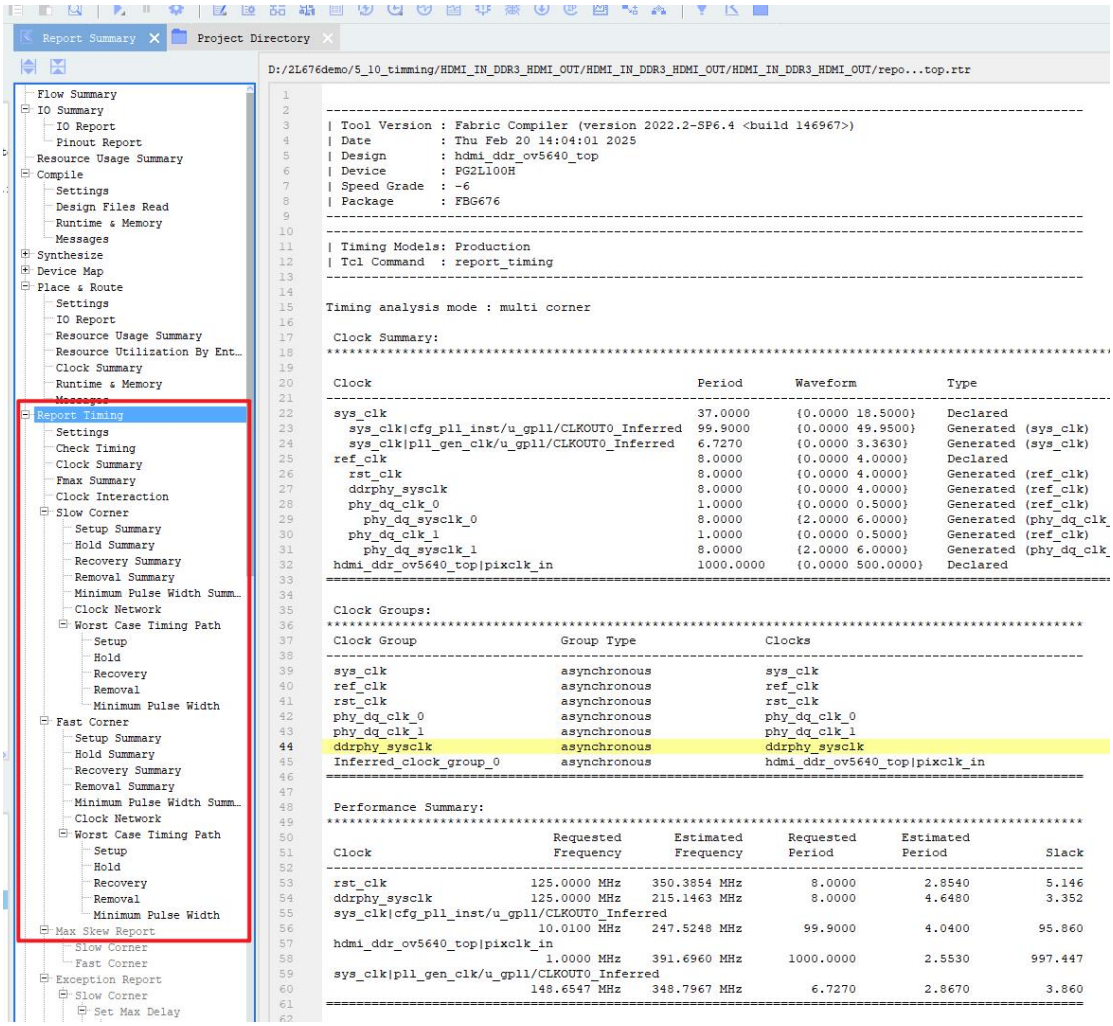


图 9-7

首先可以打开 Fmax Sumary，查看工程最高的时钟频率的报告。

Fmax Summary

Find:

	Clock	Fmax	Requested Frequency	Slack
1	rst_clk	350.3854MHz	125.0000MHz	5.146
2	ddrphy_sysclk	215.1463MHz	125.0000MHz	3.352
3	sys_clk|cfg_pll_inst/u_gpll/CLKOUT0_Inferred	247.5248MHz	10.0100MHz	95.860
4	hdmi_ddr_ov5640_top|pixclk_in	391.6960MHz	1.0000MHz	997.447
5	sys_clk|pll_gen_clk/u_gpll/CLKOUT0_Inferred	348.7967MHz	148.6547MHz	3.860

图 9-8

以 rst_clk 为例子,在工程中我们需要的频率是 125MHZ,而 Fmax 即最大时钟频率可以到 350.3854MHZ,是比我们需要的 125MHZ 还要高,所以满足设计要求。而 Slack 表示的是时间余量,该值为正则表示设计满足时序,该值如果为负的话则表示时序违例,需要优化逻辑,减少延迟等。

接下来打开 Clock Interaction 报告,

图 9-9

还是以 rst_clk 为例子, Launch Clock 表示起始触发数据的时钟边沿,而 Capture Clock 是下一个触发器捕获数据的时钟边沿。这里就要引入一个概念,数据到达 Capture Clock 边沿之前或之后都要保持稳定。也就是建立时间和保持时间,一旦不满足,则时序出现违例。建立时间中, WNS(最差负时序余量),所有时序路径里面最差的 Slack。和 TNS(总负时序余量)所有 Setup Slack 小于 0 的值的总和。当 WNS>=0, TNS=0 时,则表示时序没有违例。WNS<0 则 TNS 也会 < 0,此时表示时序需要优化。WNS 通常决定工程能否在目标频率下工作。在保持时间中,WHS(最差保持时间余量),THS(总保持时间余量)所有 Hold Slack 小于 0 的值的总和。WHS < 0 则表示有保持时间违例,可能会导致亚稳态。所以综上所述, WNS 反映工程能否在目标时钟频率下工作, WHS 违例的话可能会导

致亚稳态。常见的优化手段, 当建立时间违例时, 可以降低组合逻辑延迟, 插入多级流水线, 降低时钟频率, 优化布局布线。当保持时间违例时, 可以增加数据延迟, 插入 buffer, 优化时钟 skew 等。

10. LVDS 应用实验

➢ 目的: 完成 LVDS 自测收发, 验证发送和接收的数据是否一致。

10.1. 实验原理

对于发送端,FPGA 内部产生测试数据,通过 OSERDES 原语将 4 位并行数据转串行数据,再通过 OUTBUFTDS 原语将串行数据转为差分数据,外部的参考时钟由 FPGA 内部 PLL 倍频得到高频时钟, 再通过原语 OUTBUFTDS 原语转为差分时钟信号。

对于接收端, 差分时钟信号通过 BUFGDS 原语转为单端时钟信号。然后通过 IODELAY 原语设置时钟延时参数,使时钟能够正确的实现数据采样。差分数据通过 BUFDS 原语转为单端数据信号, 再通过 ISERDES 原语实现串并转换。

最后检测收到的并行数据是否和发送的并行数据一致, 如果不一致, 亮 err 指示灯。

10.2. 接口列表

工程的顶层接口列表

端口名	输入/输出	说明
ref_clk	input	FPGA 单端时钟输入 50MHZ
dbg_key_oper	input	按键信号,调整 IODELAY 的 STEP 值,每按一次延迟增加约 10ns

rstn_in	input	工程的外部复位
err_flag	output	当接收到的数据错误时, 该信号拉高
tx_data_p/n[CHAN_N]	output	发送端数据差分信号
tx_clk_p/n	output	发送端时钟差分信号
rx_data_p/n[CHAN_N]	input	接收端数据差分信号
rx_clk_p/n	input	接收端时钟差分信号

- pgd_clk_rst_gen 模块

该模块功能是产生发送端时钟及复位信号, 产生接收端复位时序参考时钟。通过锁相环倍频得到两路时钟, 一路经过二分频作为 OSERDES 单元的 IO 时钟, 一路通作为发送的系统时钟。同时也产生了发送和接收的参考复位信号。

端口名	输入/输出	说明
ref_clk	input	输入用来倍频的参考时钟
rstn_in	input	全局复位
sysclk	output	输出给其余模块的系统时钟
rstn_sys	output	输出给其余模块的系统复位
rstn_in	output	工程的外部复位
tx_sysclk	output	发送模块的系统时钟

tx_ioclk	output	发送端高速 IO 时钟
tx_rstn	output	发送模块的复位
tx_gtp_rstn	output	发送端并串转换复位控制

- pgd_ddr_lvds4to1_tx_top 模块

该模块功能是通过使用原语 GTP_OSERDES,将要发送的并行数据转为串行数据,最后使用 GTP_OUTBUFDS 原语实现时钟信号和数据信号的高速差分输出。

端口名	输入/输出	说明
tx_sysclk	input	发送端的系统时钟
tx_rstn	input	发送端系统复位
tx_ioclk	output	发送端高速 IO 时钟
tx_gtp_rstn	output	发送端并串转换复位控制
tx_data[4*CHAN_N-1:0]	input	输入待发送的并行数据
tx_data_p/n[CHAN-1:0]	output	发送端数据差分信号
tx_clk_p/n	output	发送端时钟差分信号

- pgd_ddr_lvds4to1_rx_top 模块

该模块是接收模块顶层, 包括时钟、数据和 trainning 模块。对于接收部分, 首先将接收端的高速差分时钟信号通过 GTP_INBUFGDS 原语转为单端时钟信号,

通过 IODLY 单元设定的延迟参数进行时钟延时, 使时钟能正确稳定地实现数据

采样, 再经过分频得到接收端的系统时钟。通过 GPT_INBUFDS 原语将接收到的差分数据信号转为单端数据信号。最后通过 GPT_ISERDES 原语,将串行的单端数据转换为并行数据。输出的数据会送到 pgd_ddr_lvds4to1_rx_training_ctrl 训练模块中再输出给到用户端。

端口名	输入/输出	说明
sysclk	input	复位时序参考时钟（50MHZ）
dbg_key_oper	input	按键调整 IO delay
rstn_sys	input	输入的复位信号
rx_clk_p/n	input	接收端时钟差分信号
rx_data_p/n[CHAN_N-1:0]	input	接收端数据差分信号
rx_sysclk	output	接收端的系统时钟
rx_rstn	output	接收端系统复位
sync_done	output	通道同步标志
rx_train_done	output	训练结束标志
rx_data[4*CHAN_N-1:0]	output	并行数据输出

- pgd_ddr_lvds4to1_test 模块。

该模块产生了测试数据并对接收数据进行校验。当接收的数据和发送的数据不一致的时候, err 指示灯将亮起。

端口名	输入/输出	说明
tx_sysclk	input	发送端的系统时钟

tx_rstn	input	发送端系统复位
tx_mode[1:0]	input	发送模式
tx_train_done	input	训练完成信号
tx_data[4*CHAN_N-1:0]	output	待发送的并行数据
rx_sysclk	input	接收端的系统时钟
rx_rstn	input	接收端系统复位
rx_data[4*CHAN_N-1:0]	input	接收端的并行数据输入
rx_train_done	input	训练结束标志
test_trig	input	通道同步标志
err_led[CHAN_N-1:0]	output	错误指示灯信号

10.3. 工程说明

- 工程框架

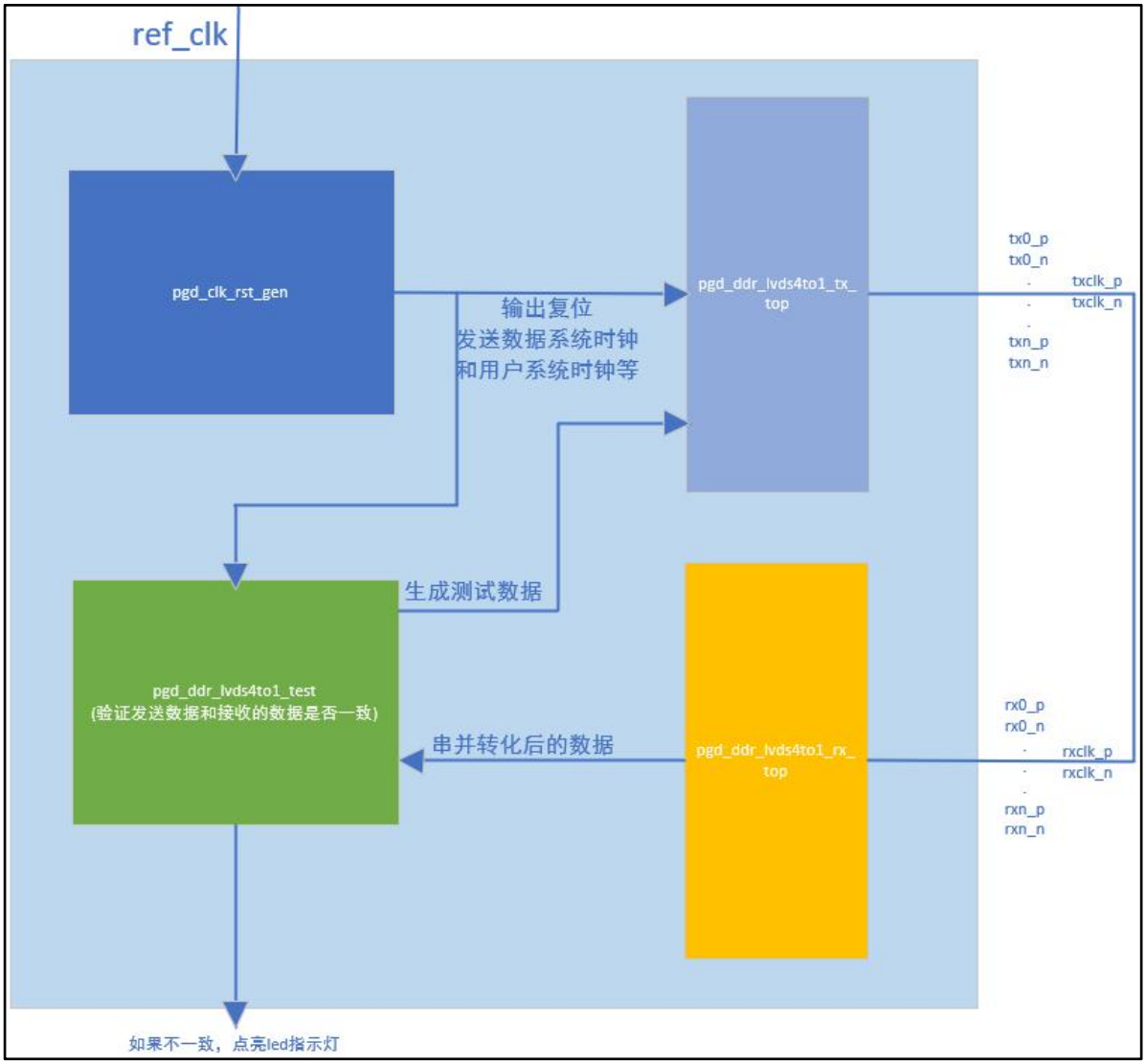


图 10.3-1 LVDS 工程流程图

在线抓取 tx_data 和 rx_data 信号, 观察其数据是否一致, 一致则表示实验成功, 且可以看到开发板上 LED8 并未被点亮, 说明没有错误。

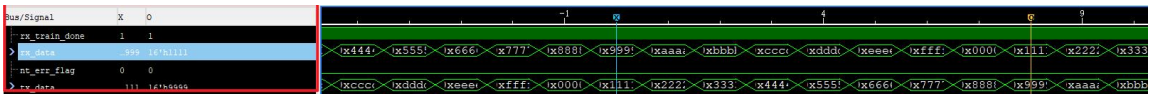


图 10.3-3 结果展示

rx_train_done 高电平则表示训练完成。

nt_err_flag 信号为低电平表示数据没有错误, 实验结果正确。

11. DDR3 IP 应用实验

- 目的: 实现 HDMI_IN 接收, 经 DDR3 缓存输出, 实现 HDMI_OUT 环路显示。

11.1. 实验原理

FPGA 实现对 HDMI 芯片的 IIC 接口配置, 实现 HDMI 输入和 HDMI 输出, 通过 AXI 接口将 HDMI 输入的数据写入到 DDR3 之中, 再通过 AXI 接口把数据从 DDR3 中读出来供 HDMI 屏幕显示。分辨率为 1920x1080@60hz。

- DDR3 IP 的使用

1. DDR3 IP 功能特性

Logos2 系列 FPGA 器件的 DDR3 IP 称为 HMIC_S (High performance Memory Interface Controller Soft core) IP, 该 IP 为结合了 MC 控制器与物理 PHY 层接口的软核, 向用户提供了 AXI4 接口, 可用于 DDR3 SDRAM 的高速系统设计。

HMIC_S IP 的主要特性如下。

- 支持 DDR3, 最高接口速率 1066Mbps;
- 支持最大数据位宽 72bit;
- 支持 PHY Only 模式。

2. DDR3 IP 系统框图

HMIC_S IP 系统框图如图 11.1-1 所示。

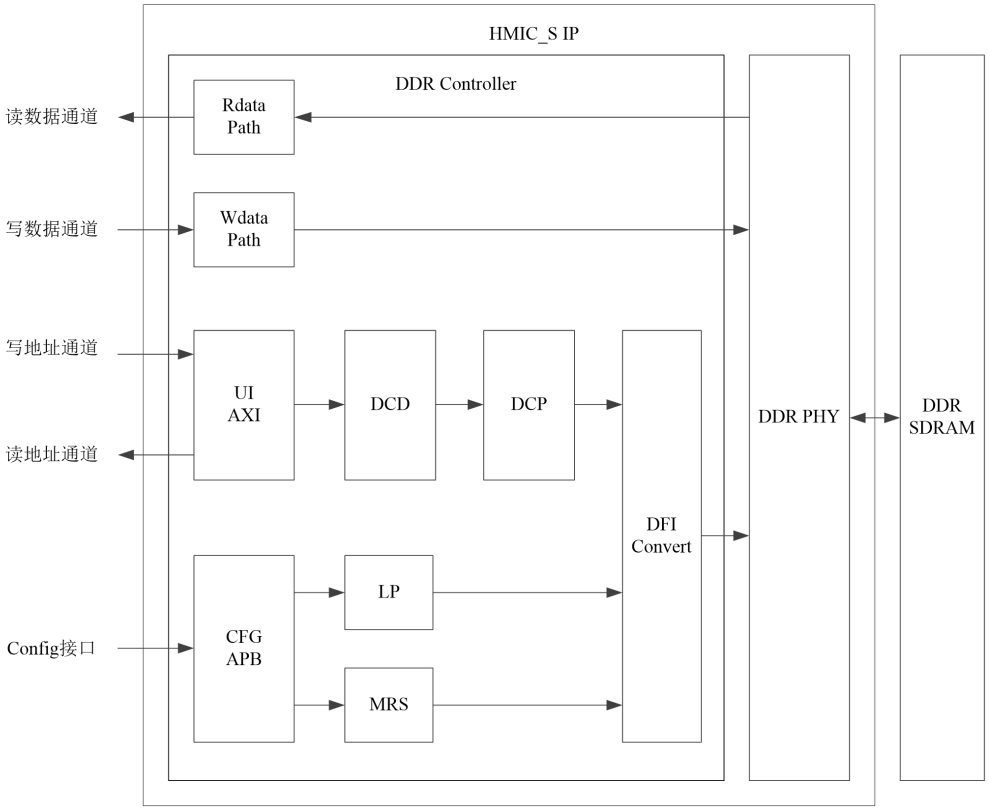


图 11.1-1 HMIC_S IP 系统框图

- DDR3_IP 配置

DDR3 的 IP 参数配置如下所示:

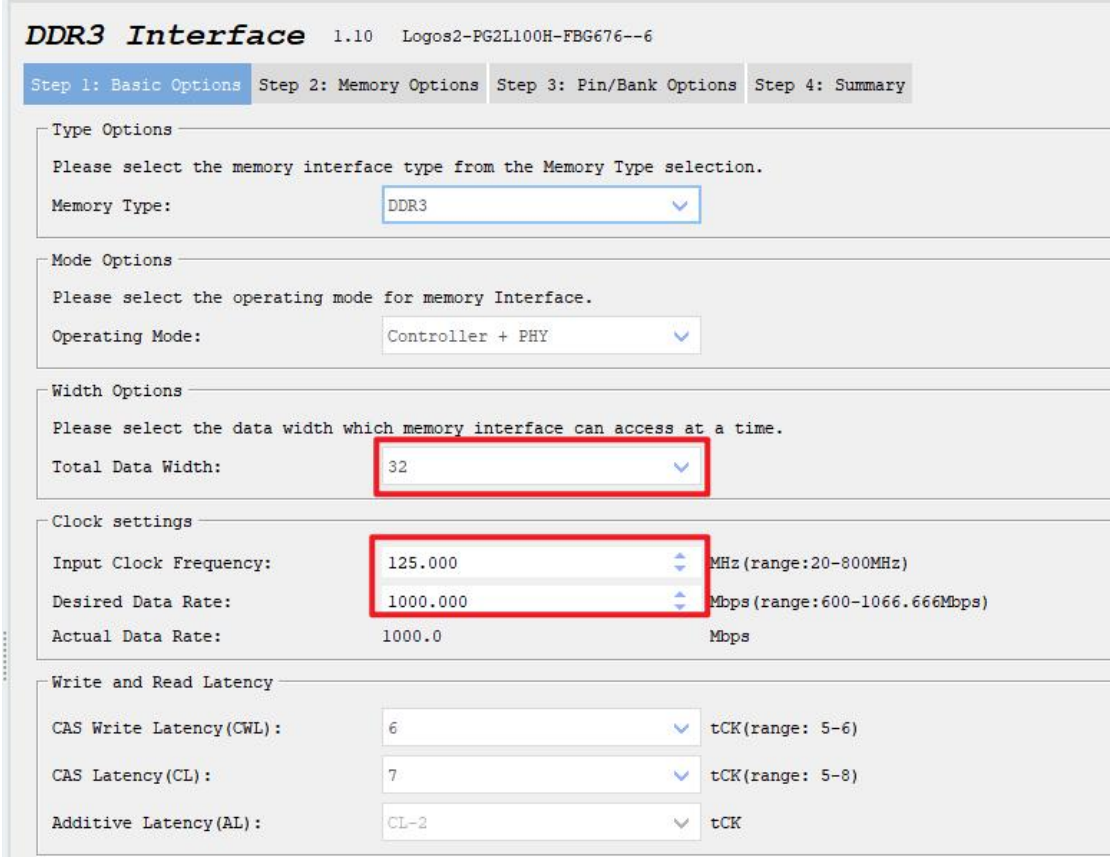


图 11.1-2 HMIC_S IP 基础设置

参数配置如图 11.1-2, 配套的板卡上有两片 DDR3, 一片 DDR3 的数据位宽是 16bit, 因此将 Total Data Width 改为 32bit, 提供给板卡的工作时钟是 125MHZ, 因此 Input Clock Frequency 选择 125MHZ。Desired Data Rate, 即数据速率, 最低可以选择 600Mbps, 最高可以选择 1066.666Mbps, 由于本次实验是传输视频流数据, 数据速率要尽可能大, 本次实验选择 1000Mbps, 即每秒传输 1000M 个 bit。其余保持默认即可。

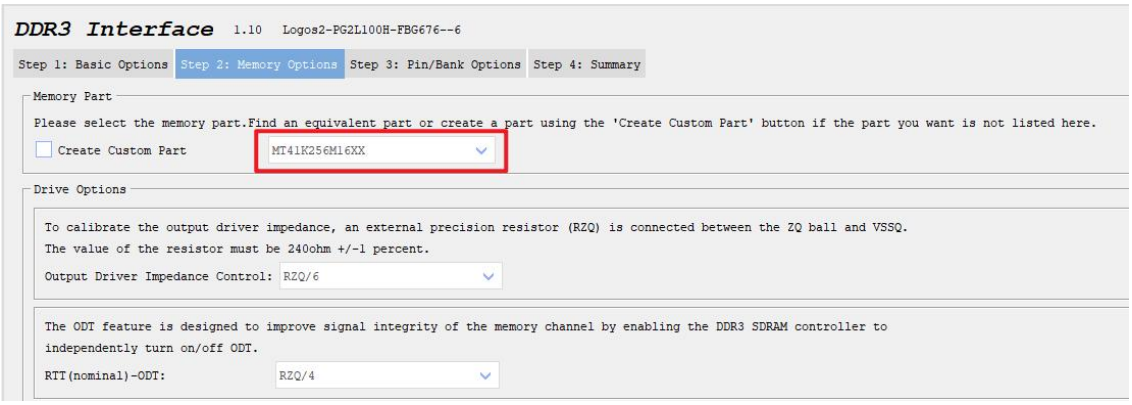


图 11.1-3 HMIC_S IP 内存选项界面

主要修改 Memory Part 部分,板卡上的 DDR3 芯片型号为 MT41K256M16XX,其余保持默认即可。

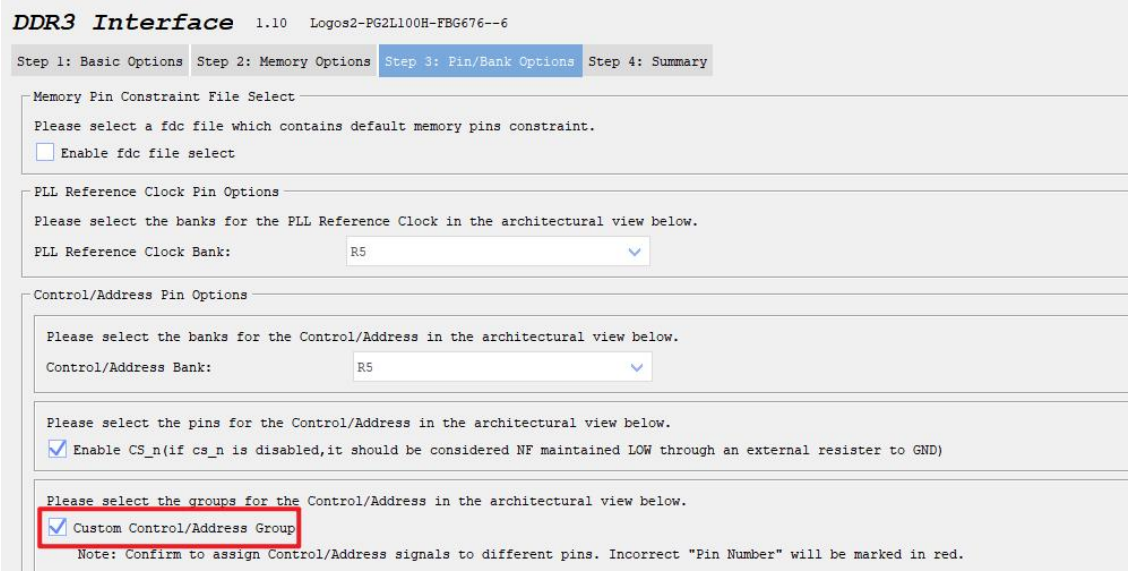


图 11.1-4 HMIC_S IP 引脚绑定界面

这里需要修改为 Custom Control 即用户来绑定管脚,这里需要根据原理图来绑定,此处给出参考, 其余保持默认, 如图 11.1-5,图 11.1-6 所示

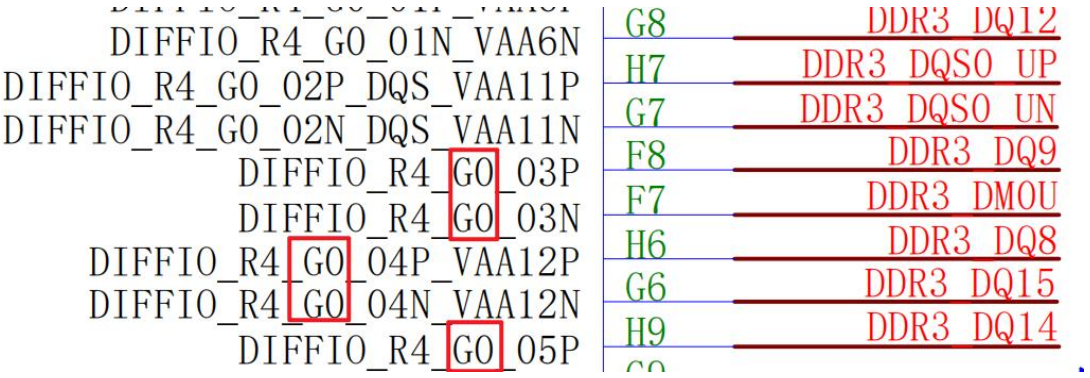


图 11.1-5 HMIC_S IP Group number 示意

DDR3IP 所设置的 group number 在核心板的原理图中参考图 11.1-5 的红色方框部分 G0 即为该引脚的 group number

Signal Name	Group Number	Pin Number
RESET		H1
CKE	G2	R1
CK	G3	U6
CK_N	G3	U5
CS	G2	N4
RAS	G3	P6
CAS	G3	T7
WE	G3	T8
ODT	G1	H2
BA0	G2	P4
BA1	G2	R2
BA2	G3	T5
A0	G1	J1
A1	G1	L3
A2	G1	L2
A3	G2	P3
A4	G2	T2
A5	G1	M1
A6	G2	U1
A7	G1	K1
A8	G2	U2
A9	G1	K2
A10	G2	T3
A11	G2	P1
A12	G2	T4
A13	G1	M2
A14	G1	N1

Data Pin Options

Please select the banks and groups for the data in the architectural view below.

Signal Name	Bank Number	Group Number
DQ[0-7]	R4	G1
DQ[8-15]	R4	G0
DQ[16-23]	R4	G2
DQ[24-31]	R4	G3

图 11.1-6 引脚参考图

11.2. 接口列表

工程接口

表 11.2-1

端口名	输入/输出	说明
MEM_ROW_ADDR_WIDTH	\	DDR 行地址位宽
MEM_COL_ADDR_WIDTH	\	DDR 列地址位宽
MEM_BADDR_ADDR_WIDTH	\	DDR BANK 地址位宽
MEM_DQ_WIDTH	\	DDR 数据位宽
MEM_DQS_WIDTH	\	DDR 数据选通位宽
sys_clk	input	系统时钟输入(27MHZ)
clk_p/clkn	input	系统差分时钟输入 (125MHZ)
mem_rst_n	output	DDR 复位信号 低电平有效
mem_ck	output	DDR 输入系统时钟
mem_ck_n	output	DDR 输入系统时钟
mem_cke	output	时钟使能信号。高有效
mem_cs_n	output	DDR 片选信号
mem_ras_n	output	行地址选通信号

mem_cas_n	output	列地址选通信号
mem_we_n	output	DDR 写操作信号
mem_odt	output	DDR ODT
mem_a[MEM_ROW_ADDR_WIDTH-1:0]	output	DDR 行列地址总线
mem_ba[MEM_BADDR_WIDTH-1:0]	output	DDR BANK 地址
mem_dqs[MEM_DQ_WIDTH/8-1:0]	双向	DDR 数据选取信号
mem_dqs_n[MEM_DQ_WIDTH/8-1:0]	双向	DDR 数据选取信号
mem_dq[MEM_DQ_WIDTH-1:0]	双向	DDR 数据
mem_dm[MEM_DQ_WIDTH/8-1:0]	output	DDR 数据掩码
iic_scl	output	HDMI 输入芯片的 SCL 信号
iic_sda	双向	HDMI 输入芯片的 SDA 信号
pixclk_in	input	HDMI 输入的像素时钟
vs_in	input	HDMI 输入的场同步信号
hs_in	input	HDMI 输入的行同步信号
de_in	input	HDMI 输入的像素有效信号
r_in[7:0]	input	HDMI 输入的像素的红色分量
g_in[7:0]	input	HDMI 输入的像素的绿色分

		量
b_in[7:0]	input	HDMI 输入的像素的蓝色分量
pix_clk	output	HDMI 输出的像素时钟
vs_out	output	HDMI 输出的场同步信号
hs_out	output	HDMI 输出的行同步信号
de_out	output	HDMI 输出的像素有效信号
r_out[7:0]	output	HDMI 输出的像素的红色分量
g_out[7:0]	output	HDMI 输出的像素的绿色分量
b_out[7:0]	output	HDMI 输出的像素的蓝色分量
iic_tx_scl	output	HDMI 输出芯片的 SCL 信号
iic_tx_sda	双向	HDMI 输出芯片的 SDA 信号
rstn_out	output	1ms 复位信号
hdmi_init_led	output	HDMI 输出芯片寄存器初始化完成信号

1） fram_buf 模块

该模块的功能能是将图像数据缓存至 DDR3 之中,再将图像数据从 DDR3 缓存中读出供 HDMI 显示使用。也是整个读写 DDR3 模块的顶层。也是整个工程的核心部分。通过修改 fram_buf 模块的 input 参数 H_NUM、V_NUM 可以适配不同分辨率的视频数据, 修改 PIX_WIDTH 可以适配不同的数据位宽。

表 11.2-2

端口名	输入/输出	说明
MEM_ROW_WIDTH	\	DDR3 的行地址位宽
MEM_COLUMN_WIDTH	\	DDR3 的列地址位宽
MEN_BANK_WIDTH	\	DDR3 的 BANK 位宽
CTRL_ADDR_WIDTH	\	DDR3 的地址位宽
MEN_DQ_WIDTH	\	DDR3 数据位宽。1 片 DDR3--16bit,2 片 32bit
H_NUM	input	图像的宽度
V_NUM	输入	图像的高度
PIX_WIDTH	input	像素数据位宽 16bit、24bit
vin_clk	input	数据输入的时钟
wr_fsync	input	wr_buf 的 ram 复位(HDMI 输入的场同步信号)

wr_en	input	数据输入的有效信号
wr_data[PIX_WIDTH-1:0]	input	输入的像素数据
init_done	output	暂未使用
ddr_clk	input	DDR3 输出的用户时钟
ddr_rstn	input	DDR3 复位完成信号
vout_clk	input	要显示的像素时钟
rd_fsync	input	rd_buf 的 ram 复位(HDMI 显示的场同步信号)
rd_en	input	HDMI 显示的数据有效信号
vout_de	output	RGB 数据输出的有效信号
vout_data[PIX_WIDTH-1:0]	output	输出的 RGB 数据 24bit\16bit
axi_awaddr[CTRL_ADDR-1:0]	output	AXI 写地址
axi_awid[3:0]	output	AXI 写地址 ID
axi_awlen[3:0]	output	AXI 写突发长度 最大 16
axi_awsize[2:0]	output	AXI 突发写的大小
axi_awburst[1:0]	output	AXI 突发写类型
axi_awready	input	AXI 写地址准备信号
axi_awvalid	output	AXI 写地址有效信号

axi_wdata[MEN_DQ_WIDTH*8-1:0]	output	AXI 写数据
axi_wstrb[MEN_DQ_WIDTH-1:0]	output	AXI 写数据选通
axi_wlast	input	AXI 写最后一个数据标志
axi_wvalid	output	AXI 写有效信号
axi_wready	input	AXI 写准备信号
axi_bid[3:0]	input	AXI 写响应信号
axi_araddr[CTRL_ADDR-1:0]	output	AXI 读地址
axi_arid[3:0]	output	AXI 读地址 ID
axi_arlen[3:0]	output	AXI 读突发长度 最大 16
axi_arsize[2:0]	output	AXI 突发读的大小
axi_arburst[1:0]	output	AXI 突发读类型
axi_arvalid	output	AXI 读地址有效信号
axi_arready	output	AXI 读地址准备信号
axi_rdata[MEN_DQ_WIDTH*8-1:0]	input	AXI 读数据
axi_rvalid	input	AXI 读有效信号
axi_rlast	input	AXI 读最后一个数据指示信号
axi_rid[3:0]	input	AXI 读响应信号

通过修改 fram_buf 模块的输入参数 H_NUM、V_NUM 可以适配不同分辨率的视频数据，修改 PIX_WIDTH 可以适配不同的数据位宽。

2） wr_buf 模块

该模块主要完成将 HDMI 输入的数据缓存到 DDR3 中，使用 ram IP 来缓存 HDMI 输入的数据，并完成跨时钟传输到 DDR3 的工作时钟下，因为 AXI 总线数据位宽是 256bit，所以 ram 的读端口的数据位宽为 256bit,通过 ram 将 HDMI 输入的数据缓存并转换到 DDR3 的工作时钟下，以此实现数据传输到 DDR3 中去缓存。

表 11.2-3

端口名	输入/输出	说明
ADDR_WIDTH	\	地址位宽
ADDR_OFFSET	\	偏移地址位宽
H_NUM	\	图像的行的大小
V_NUM	\	图像的列的大小
DQ_WIDTH	\	DDR 数据位宽
LEN_WIDTH	\	长度位宽
PIX_WIDTH	\	像素位宽
LINE_ADDR_WIDTH	\	长度地址位宽
FRAME_CNT_WIDTH	\	帧计数位宽

ddr_clk	input	DDR3 输出的用户时钟
ddr_rstn	input	DDR3 的复位完成信号
wr_clk	input	数据写时钟(HDMI 输入的像素时钟)
wr_fsync	input	数据复位信号(HDMI 输入的场同步信号)
wr_data[PIX_WIDTH-1:0]	input	输入的像素数据
rd_bac	input	未使用
ddr_wreq	output	DDR3 写数据响应信号
ddr_waddr[ADDR_WIDTH-1:0]	output	DDR3 写地址
ddr_wr_len[LEN_WIDTH-1:0]	output	DDR3 突发写长度
ddr_wrdy	input	DDR3 突发写准备信号
ddr_wdone	input	DDR3 突发写完成信号
ddr_wdata[8*DQ_WIDTH-1:0]	output	DDR3 写数据
ddr_wdata_req	input	DDR3 写数据请求信号
frame_cnt	output	帧计数信号

3） rd_buf 模块

该模块主要完成将缓存在 DDR3 中的输出读出,供 HDMI 显示使用。使用 ram IP 将 DDR3 读出 256bit 数据跨时钟传输到 HDMI 显示的时钟下, 此时读出的数据位宽是 32bit, 因此需要通过计数器把 32bit 数据分为 16bit 数据即 RGB565 供 HDMI 显示使用。

表 11.2-4

端口名	输入/输出	说明
ADDR_WIDTH	\	地址位宽
ADDR_OFFSET	\	偏移地址位宽
H_NUM	\	图像的行的大小
V_NUM	\	图像的列的大小
DQ_WIDTH	\	DDR 数据位宽
LEN_WIDTH	\	长度位宽
PIX_WIDTH	\	像素位宽
LINE_ADDR_WIDTH	\	长度地址位宽
FRAME_CNT_WIDTH	\	帧计数位宽
ddr_clk	input	DDR3 输出的用户时钟
ddr_rstn	input	DDR3 的复位完成信号
vout_clk	input	读数据时钟(HDMI 显示的像素时钟)
rd_fsync	input	读地址复位信号(HDMI 的场同步信号)
vout_data[PIX_WIDTH-1:0]	output	像素数据输出
vout_de	output	像素输出有效信号
rd_en	input	HDMI 显示的像素有效信号(读使能信号)

ddr_rreq	output	DDR3 读请求信号
ddr_raddr[ADDR_WIDTH-1:0]	output	DDR3 读地址
ddr_rd_len[LEN_WIDTH-1:0]	output	DDR3 读突发长度
ddr_rrdy	input	DDR3 读准备信号
ddr_rdone	output	DDR3 突发读完成信号
ddr_rdata	input	DDR3 读出的数据
ddr_rdata_en	input	DDR3 数据有效信号
init_done	input	初始化完成信号

4） wr_rd_ctrl_top 模块

该模块负责完成 AXI 总线读写 DDR3。将配置 AXI 总线的突发长度、读写地址、使能以及数据，完成 DDR3 数据的写入和读出。

表 11.2-5

端口名	**输入/输出**	**说明**
CTRL_ADDR_WIDTH	\	DDR3 行列地址总线位

		宽
MEM_DQ_WIDTH	\	DDR3 数据位宽
clk	input	DDR3 输出的用户时钟
rstn	input	DDR3 复位完成信号
wr_cmd_en	input	DDR3 写使能信号
wr_cmd_addr[CTRL_ADDR_WIDTH-1:0]	input	DDR3 写数据地址
wr_cmd_len[31:0]	input	DDR3 写突发长度
wr_cmd_ready	output	DDR3 写准备信号
wr_cmd_done	output	DDR3 写完成信号
wr_bac	output	AXI 突发写握手完成信

		号
wr_ctrl_data[MEM_DQ_WIDTH*8-1:0]	input	DDR3 突发写数据
wr_data_re	output	DDR3 突发写请求信号
rd_cmd_en	input	HDMI 显示的数据有效信号
rd_cmd_addr[CTRL_ADDR_WIDTH-1:0]	input	DDR3 读数据地址
rd_cmd_len[31:0]	input	DDR3 突发读长度
rd_cmd_ready	output	DDR3 突发读准备信号
rd_cmd_done	output	DDR3 突发读完成信号
read_ready	input	读准备信号
read_rdata[MEM_DQ_WIDTH*8-1:0]	output	读出的 DDR3 数据

read_en	output	DDR3 突发读使能
axi_awlen[3:0]	output	AXI 写突发长度 最大 16
axi_awsize[2:0]	output	AXI 突发写的大小
axi_awburst[1:0]	output	AXI 突发写类型
axi_awready	input	AXI 写地址准备信号
axi_awvalid	output	AXI 写地址有效信号
axi_wdata[MEN_DQ_WIDTH*8-1:0]	output	AXI 写数据
axi_wstrb[MEN_DQ_WIDTH-1:0]	output	AXI 写数据选通
axi_wlast	input	AXI 写最后一个数据标志

axi_wvalid	output	AXI 写有效信号
axi_wready	input	AXI 写准备信号
axi_bid[3:0]	input	AXI 写响应信号
axi_araddr[CTRL_ADDR-1:0]	output	AXI 读地址
axi_arid[3:0]	output	AXI 读地址 ID
axi_arlen[3:0]	output	AXI 读突发长度 最大 16
axi_arsize[2:0]	output	AXI 突发读的大小
axi_arburst[1:0]	output	AXI 突发读类型
axi_arvalid	output	AXI 读地址有效信号
axi_arready	output	AXI 读地址

		准备信号
axi_rdata[MEN_DQ_WIDTH*8-1:0]	input	AXI 读数据
axi_rvalid	input	AXI 读有效信号
axi_rlast	input	AXI 读最后一个数据指示信号
axi_rid[3:0]	input	AXI 读响应信号

11.3. 在线调试

在 demo 工程中使用在线调试步骤:

1、 在 Tool 工具栏打开 Inserter 工具

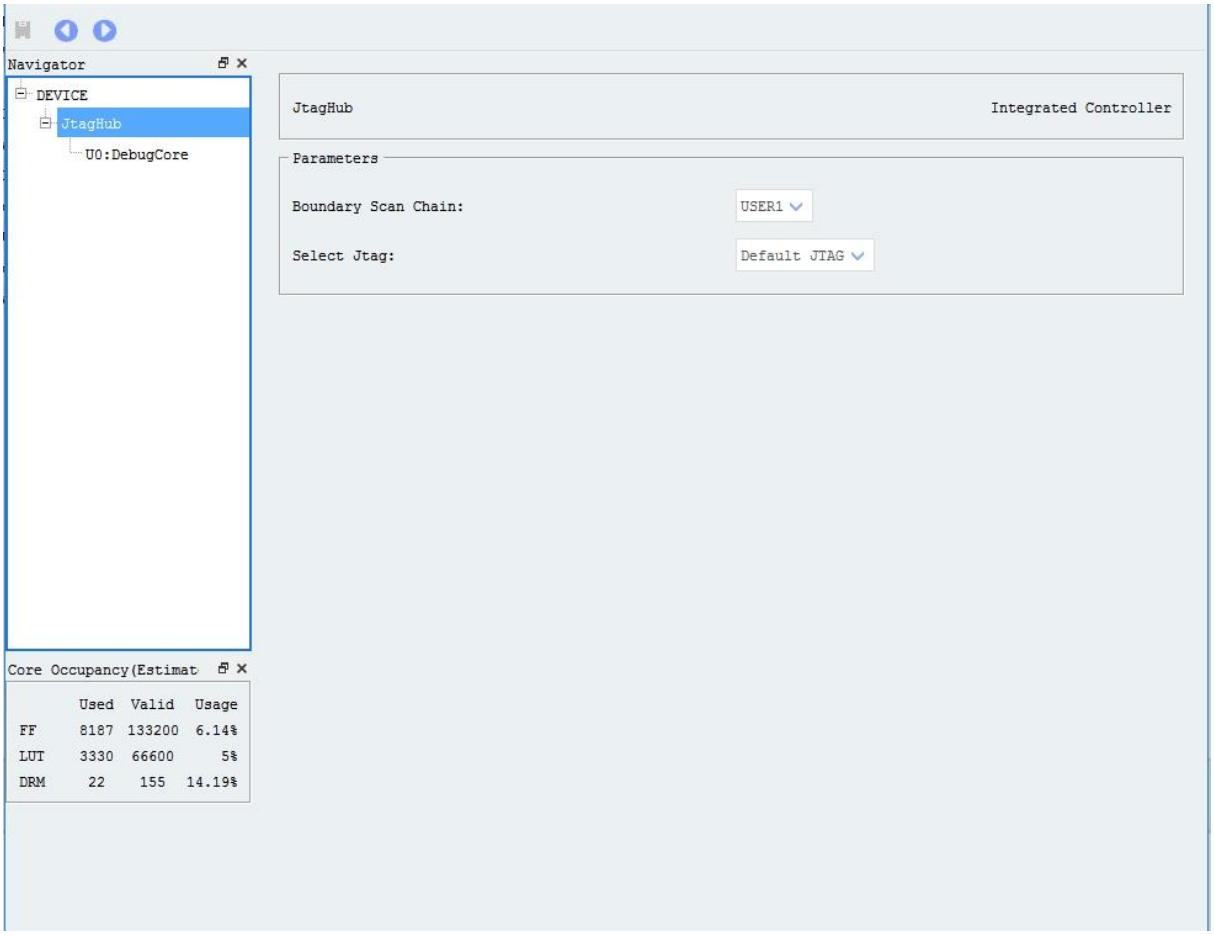


图 11.3-1 Inserter 工具界面

2、 debugcore Trigger Parameter 设置

根据在线调试信号捕获需求选择合适的数据深度,左侧窗口计算 debug core 占用资源,最终保存配置后可查看资源占用情况。

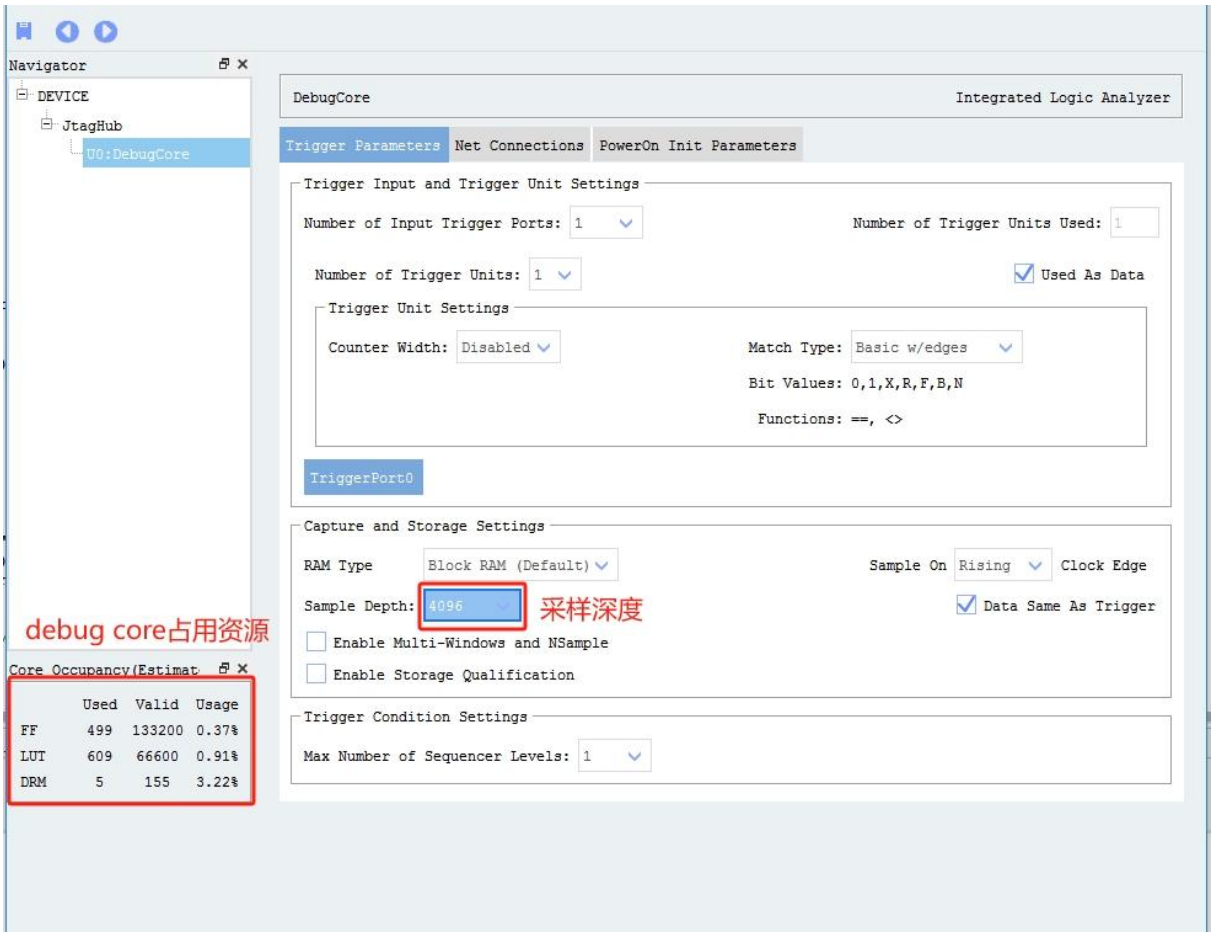


图 11.3-2 采样深度及资源占用情况

3、 点击 Modify Connection 选择采样时钟和捕获信号

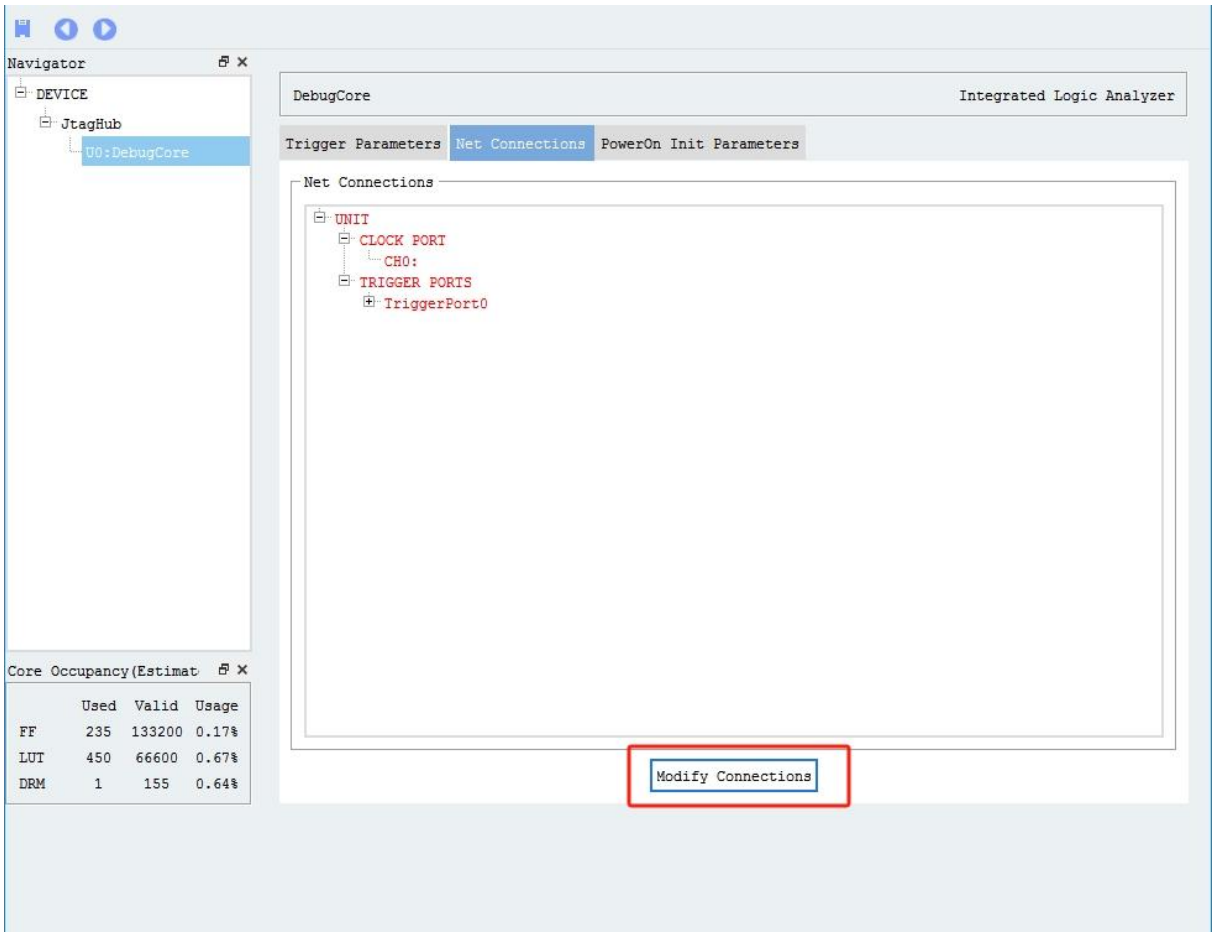


图 11.3-3 Modify Connections

采样时钟应尽量选择捕获信号的同步时钟

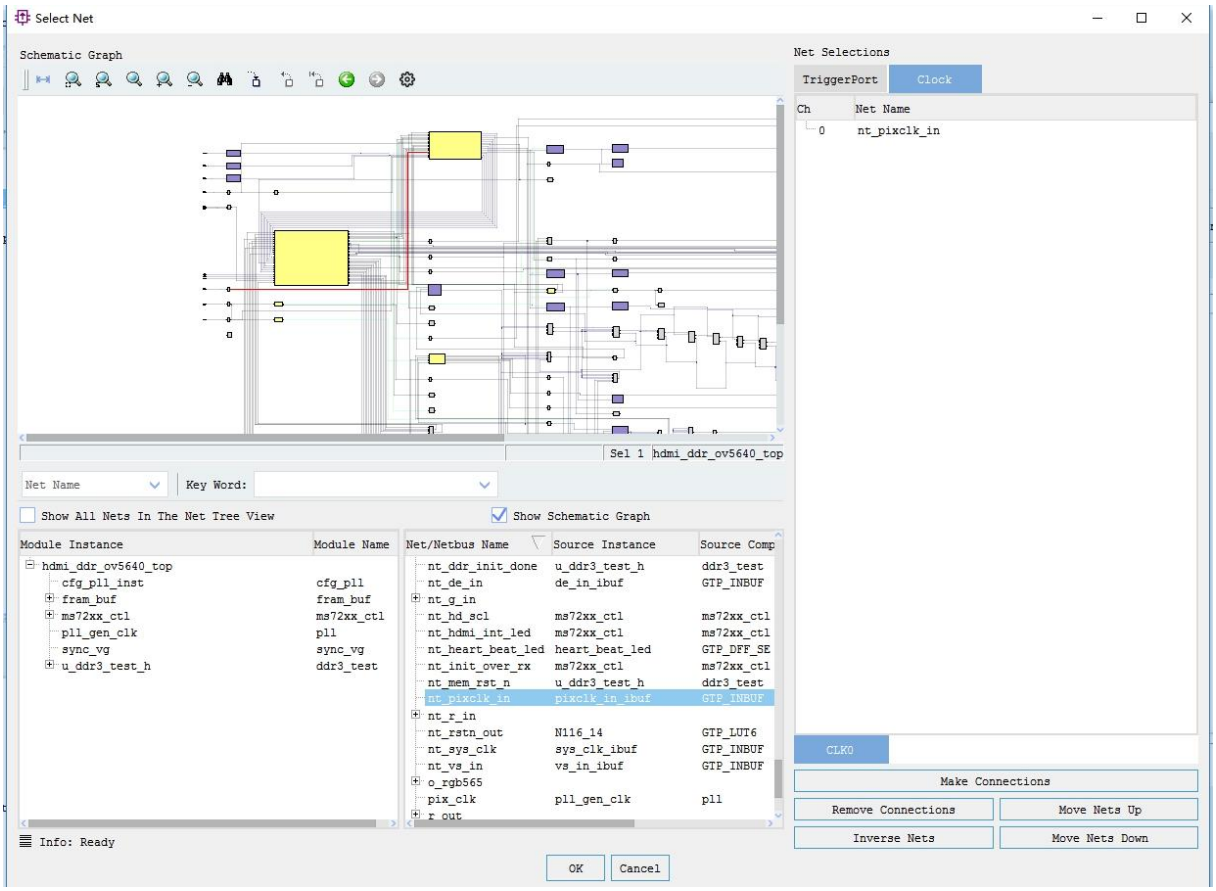


图 11.3-4 选择采样时钟

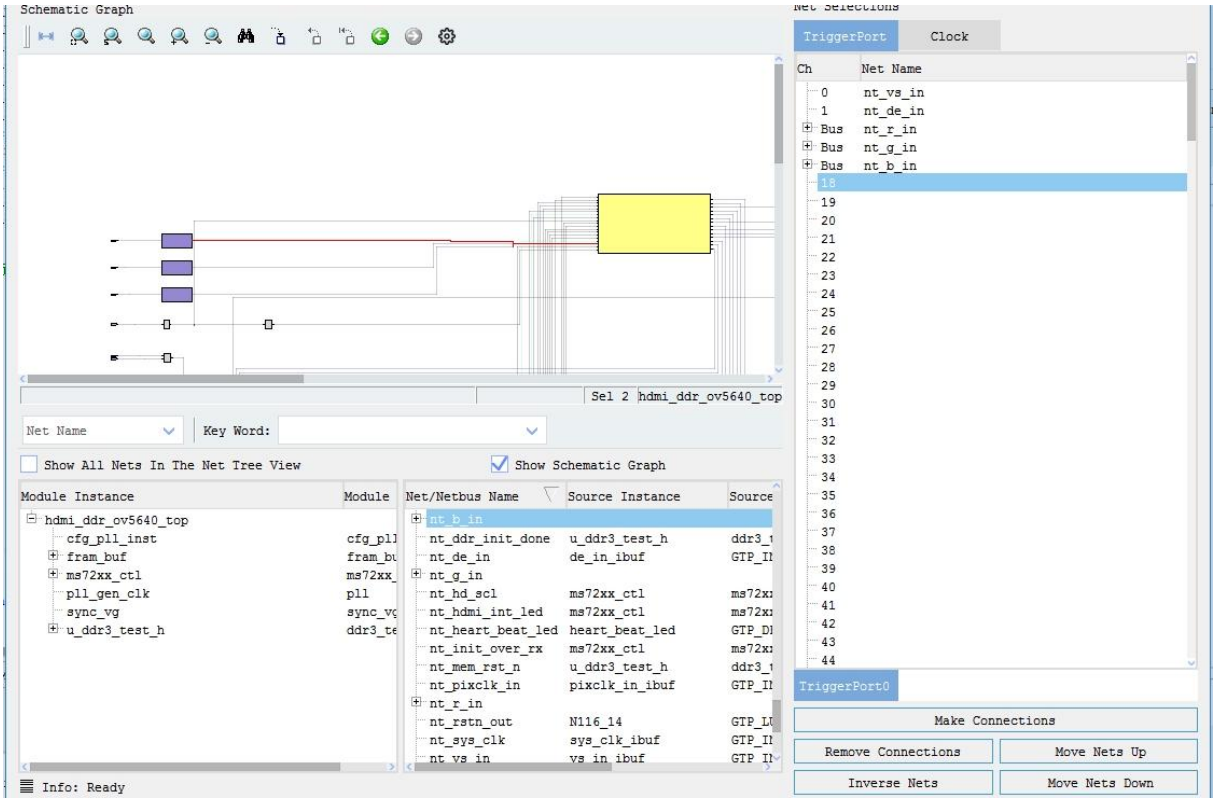


图 11.3-5 选择捕获信号

在经过 compile 和 synthesize 流程之后, 部分信号可能会被优化导致选择捕获信号是无法找到对应命名的信号, 可通过在 RTL 代码中添加特殊的标记, 综合工具会将需要观察的信号保留下来, 同时在 Inserter 中显示时, Inserter 会以其在 RTL 中的名称

进行显示。例如:

wire [7:0] o_rgb565;

wire [7:0] o_rgb565/*synthesis PAP_MARK_DEBUG="1"*/;

需注意部分信号在模块功能中未起实际作用与实际功能不相关的信号在添加特殊标记后仍可能被优化, 若需保留该信号进行测试, 可尝试将该信号改为与实际顶层输出信号相关联的逻辑。

4、 选择采样时钟和捕获信号完成后点击保存, 生成.fic 文件, 再重新经过布局

布线流程生成新的.sbit 位流文件。

5、 打开 Debugger 工具, 烧录.sbit 位流文件并且关联.fic 文件

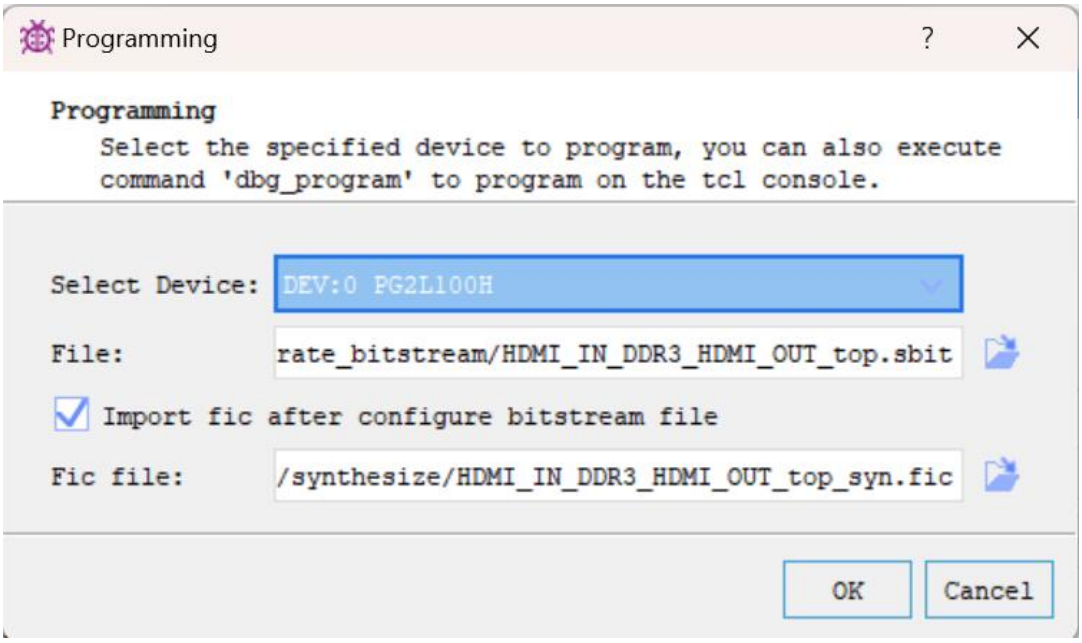


图 11.3-6 烧录.sbit 文件并关联.fic 文件

6、 设置触发条件, 示例设置信号 nt_de_in 出现上升沿时触发捕获信号, 右下方 position 默认设置触发点位置在采样空间中间, 用户可根据实际情况做对应设置。

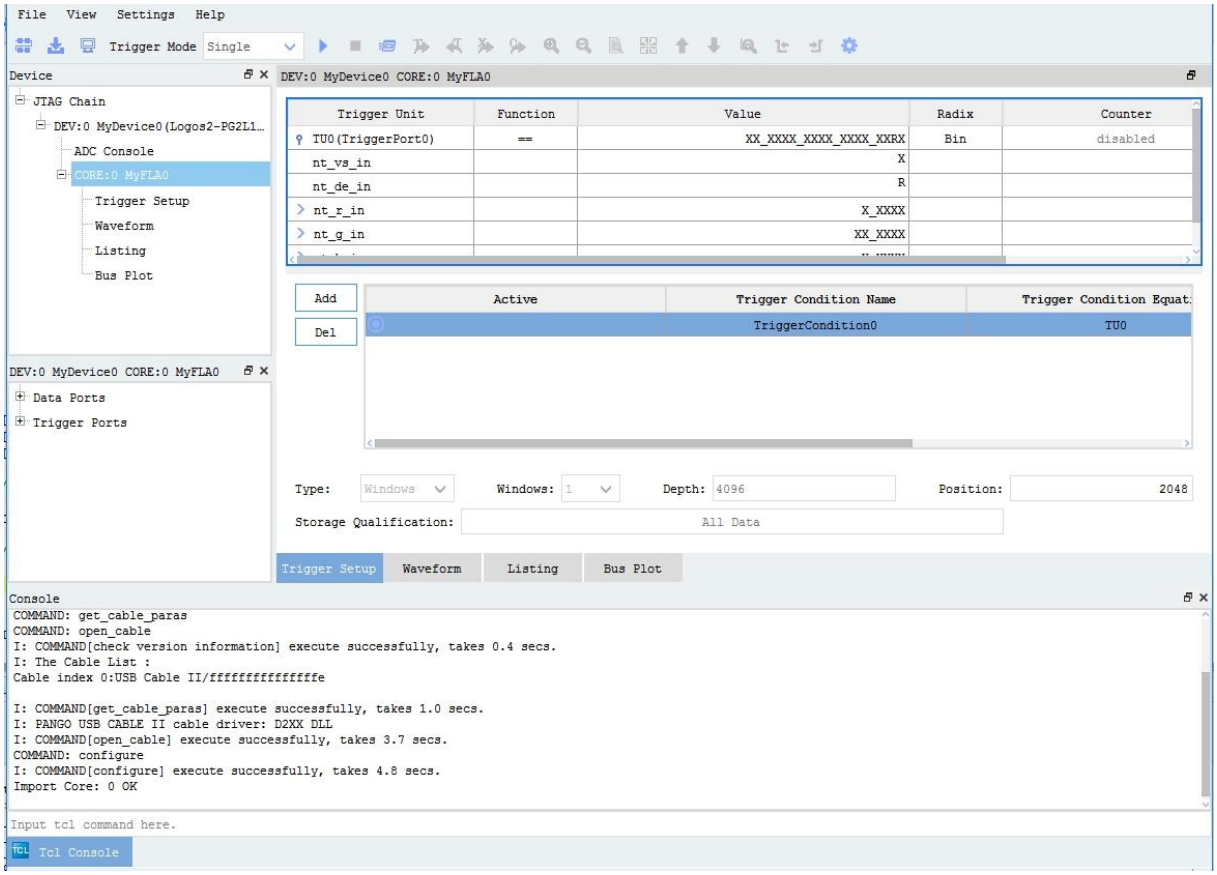


图 11.3-7 设置触发条件

7、 点击 Run 按钮后等待触发条件满足进行信号捕获和显示

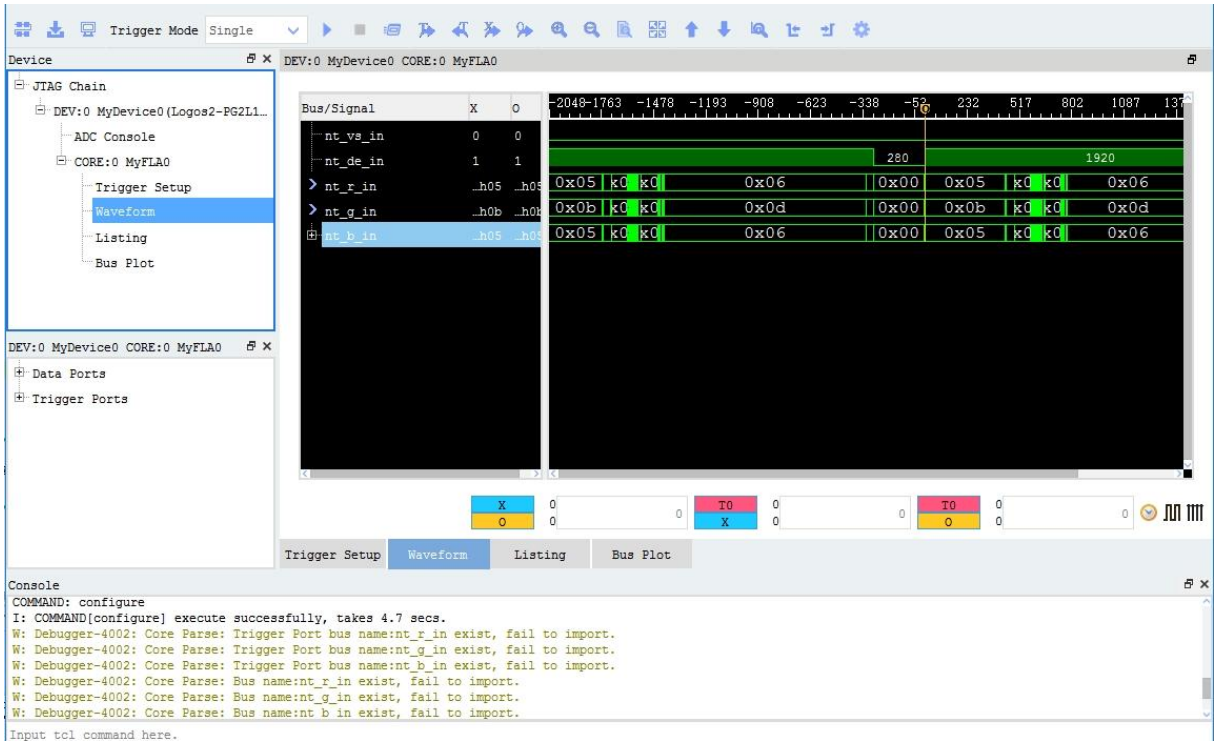


图 11.3-8 点击开始触发捕获信号

8、 若未设置触发条件, 则点击 run 按钮后即开始捕获信号

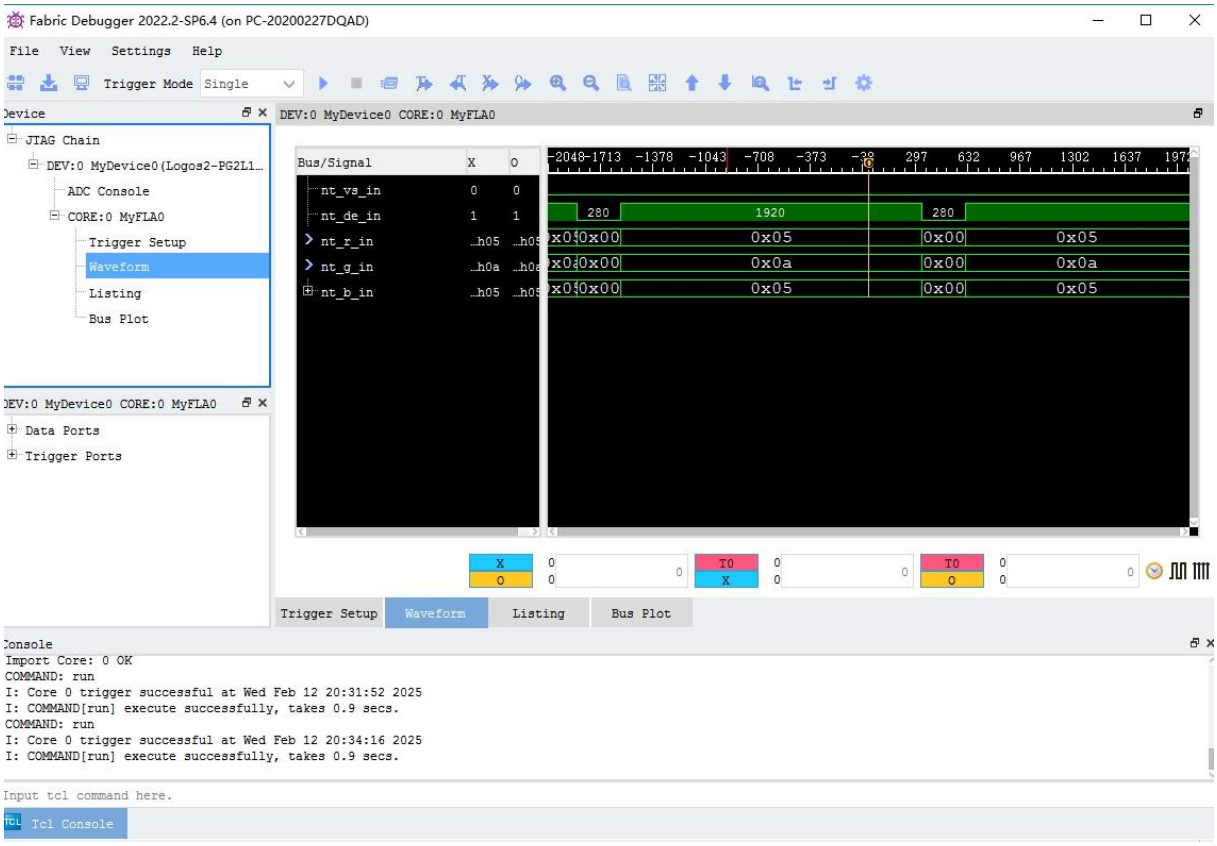


图 11.3-9 开始捕获信号

debugger 工具可实现支持 FPGA 的 ID、USER CODE、状态寄存器及指令寄存器的读取功能、逻辑位流下载、触发条件配置及信号捕捉,、回读 trigger setup 寄存器值、捕获上电初始化数据、波形分析、读取 ADC 数据等。debugger 工具详细功能使用说明请参考《Fabric_Debugger_User_Guide.pdf》

11.4. 工程说明

➢ 工程框架

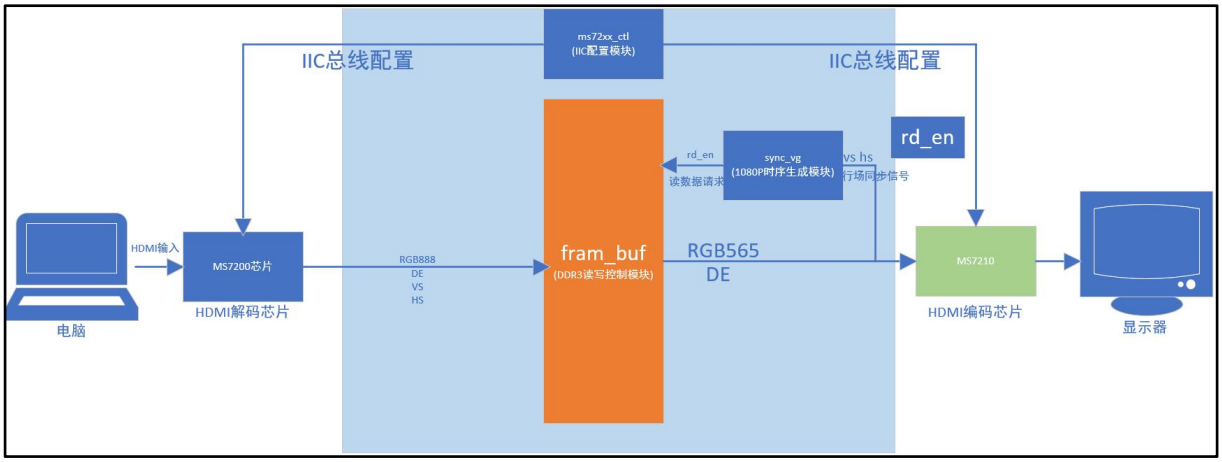


图 11.4-1 系统框架示意图

系统完成 MS7200 芯片和 MS7210 芯片的配置后会亮起对应的指示灯,表示初始化完成。HDMI 输入的数据经过 MS7200 芯片解码后给到 FPGA,通过 fram_buf 模块完成数据的写入和读出。最后将读出的数据显示到 HDMI 屏幕上。

需要两根 HDMI 线, 板卡 HDMI 输入接口需要与 PC 的 HDMI 输出接口相连, 板卡 HDMI 输出接口与显示器的 HDMI 输入接口相连。接入 12V1A 或 2A 的电源, 连接下载器后, 板卡上电。打开工程, 烧录程序。

图 11.4-3 实际效果图

下载后, 可以看到显示器会显示出笔记本电脑的画面(上图为板卡输出的 HDMI 画面)。

12. 三速以太网适配实验

12.1. 实验原理

➢ 以太网速据率识别

以太网传输在物理层是通过几组无同源时钟的 LVDS 传输数据的, PHY 芯片会对 LVDS 进行数据及时钟的恢复, 此过程可以明确以太网的传输数据率, 故而我们可以通过对 PHY 芯片的状态进行读取后得到当前网口的工作数据率, 后端数据接口按照实际的数据率进行处理; 本次实验通过 MDIO 读写 RTL8211F-CG, 实时监测当前网口的工作速率。

PHY 芯片的配置接口是 MDIO, 通过 MDIO 可以对 PHY 芯片进行配置以及状态获取, MDIO 的协议格式,功能定义,通信时序如图 12.1-1、图 12.1-2、图 12.1-3、图 12.1-4 所示;

	Management Frame Fields							
	Preamble	**ST**	**OP**	**PHYAD**	**REGAD**	**TA**	**DATA**	**IDLE**
Read	1…1	01	10	AAAAA	RRRRR	Z0	DDDDDDDDDDDDDDDD	Z
Write	1…1	01	01	AAAAA	RRRRR	10	DDDDDDDDDDDDDDDD	Z

图 12.1-1 MIDO 帧格式

Name	**Description**
Preamble	32 Contiguous Logical 1's Sent by the MAC on MDIO, along with 32 Corresponding Cycles on MDC. This provides synchronization for the PHY.
ST	Start of Frame. Indicated by a 01 pattern.
OP	Operation Code. Read: 10 Write: 01
PHYAD	PHY Address. Up to eight PHYs can be connected to one MAC. This 3-bit field selects which PHY the frame is directed to.
REGAD	Register Address. This is a 5-bit field that sets which of the 32 registers of the PHY this operation refers to.
TA	Turnaround. This is a 2-bit-time spacing between the register address and the data field of a frame to avoid contention during a read transaction. For a read transaction, both the STA and the PHY remain in a high-impedance state for the first bit time of the turnaround. The PHY drives a zero bit during the second bit time of the turnaround of a read transaction.
DATA	Data. These are the 16 bits of data.
IDLE	Idle Condition. Not truly part of the management frame. This is a high impedance state. Electrically, the PHY's pull-up resistor will pull the MDIO line to a logical '1'.

图 12.1-2 MDIO 帧功能描述

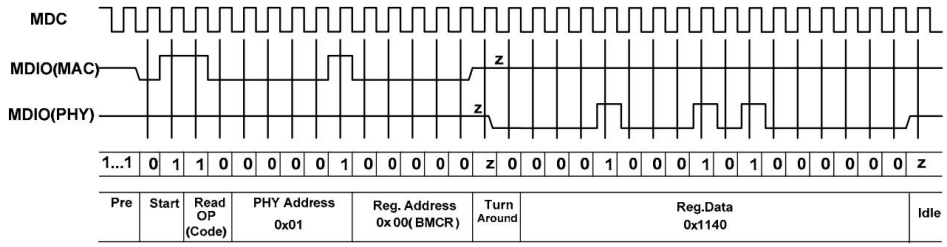


图 12.1-3 MDC/MDIO 读时序

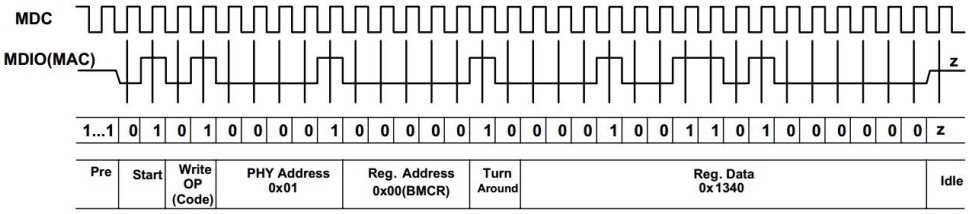


图 12.1-4 MDC/MDIO 写时序

对于 PHY 的寄存器参考具体器件的数据手册, 盘古 100 开发板上使用的 PHY 是 RTL8211F-CG, 以 RTL8211F-CG 为例, 要读取其对应的状态需要先往寄存器 31(十进制)写入 0xa43 切换 page。因为根据手册即图 12.1-6 所示, 状态寄存器在 page0xa43, 地址偏移为 0X1A。

7.11.3. Change Page

Set MDIO commands as shown below in order to switch to the desired Page 0xXYZ (in Hex).

1. Write Register 31 Data = 0x0XYZ (Page 0xXYZ)

2. Read/Write the target Register Data

3. Write Register 31 Data = 0x0000 or 0xa42 (switch back to IEEE Standard Registers)

图 12.1-5 切换 page

8.3.18. PHYSR (PHY Specific Status Register, Page 0xa43, Address 0x1A)

Table 39. PHYSR (PHY Specific Status Register, Page 0xa43, Address 0x1A)

Bit	Name	Type	Default	Description
26.15	RSVD	RO	0	Reserved.
26.14	ALDPS State	RO	0	Link Down Power Saving Mode. 1: Reflects local device entered Link Down Power Saving Mode, i.e., cable not plugged in (reflected after 3 sec). 0: With cable plugged in
26.13	MDI Plug	RO	0	MDI Status. 1: Plugged 0: Unplugged
26.12	NWay Enable	RO	1	Auto-Negotiation (NWay) Status. 1: Enable 0: Disable
26.11	Master Mode	RO	0	Device is in Master/Slave Mode. 1: Master mode 0: Slave mode

Bit	Name	Type	Default	Description
26.10:9	RSVD	RO	00	Reserved.
26.8	EEE capability	RO	0	1: Both local and link-partner have EEE capability of current speed
26.7	Rxflow Enable	RO	0	Rx Flow Control. 1: Enable 0: Disable
26.6	Txflow Enable	RO	0	Tx Flow Control. 1: Enable 0: Disable
26.5:4	Speed	RO	00	Link Speed. 11: Reserved 10: 1000Mbps 01: 100Mbps 00: 10Mbps
26.3	Duplex	RO	0	Full/Half Duplex Mode.

图 12.1-6 RTL8211F-CG 寄存器

12.2. 接口列表

1) mdio_ctrl.v mdio 控制模块

该模块主要通过状态机控制 mdio 发起对应的读写操作，在初始化结束后，会先发起写操作，往地址 0x1F(10 进制 31)写入 0xa43 切换 page，之后每隔一定时间读取寄存器 0x1a 来实时检测网口速率。配套实验代码源码及仿真详见《紫光同创 FPGA 权威设计方法学实验指导手册》

端口名	输入/输出	说明
i_clk	input	系统时钟, 50mhz
i_rst_n	input	异步复位, 低电平复位
mdc	output	总线时钟信号
mdio	input/output	数据线
eth_status	output	网口速率状态 2'b10:1000mbps 2'b01:100mbps 2'b00:10mbps

2) mdio_master_driver.v

该模块主要驱动 mdio, 完成对 phy 的读写操作。配套实验代码源码及仿真详见《紫光同创 FPGA 权威设计方法学实验指导手册》

端口名	输入/输出	说明
clk	input	系统时钟, 50mhz
rstn	input	异步复位, 低电平复位
reset_n_done	output	总线时钟信号
mdc	output	总线时钟线
mdio	output	数据线
start	input	开始传输信号

opcode	input	2bit 操作码 10 表示读 01 表示写
phy_addr	input	5bit 的 phy 地址
reg_addr	input	5bit 的寄存器地址
write_data	input	要写入的数据
write_done	output	写完成信号
read_data	output	读出的数据
read_data_en	output	读完成信号
ready	output	发送数据期间为低电平

3) ethernet_test.v 顶层模块

该模块为顶层模块, 主要是例化一些子模块以及 pll, 完成各个子模块之间的链接。配套实验代码源码及仿真详见《紫光同创 FPGA 权威设计方法学实验指导手册》

端口名	输入/输出	说明
sys_clk	input	系统时钟, 50mhz
sys_rst_n	input	异步复位, 低电平复位
status_led	output	网口速率状态。2'b10:1000mbps 2'b01:100mbps 2'b00:10bmps
led_test	output	预留测试灯

phy_rstn	output	phy 的复位信号
mdc	output	总线时钟线
mdio	input/output	数据线

12.3. 工程说明

1) 准备工作

第一步: 首先确认一下自己 PC 的网卡是否是千兆网卡, 用户可以点击本地连接查看, 再用五类+或者六类网线连接开发板的网口和 PC 的网口。板卡上电后下载程序, 然后观察板卡上的 led 灯的状态。如果电脑网卡速率配置为千兆, 则此时 LED0 熄灭, LED1 亮。如果电脑网卡速率配置为百兆,则此时 LED1 亮,LED1 熄灭。如果电脑网卡速率配置为十兆,则此时 LED0 熄灭, LED1 熄灭。