数字逻辑与计算机组成实验

LAB 08：VGA 接口控制器实现

郑凯琳 205220025
205220025@smail.nju.edu.cn

（一）实验目的

利用上述控制器，在显示器上显示一张静态图片。课程网站上提供了缺省 图片的 coe 文件，也提供了图片到 coe 的 MATLAB 转换代码。

对于显存实现，可以采用 IP 核生成单口 RAM，每个单元 12bit，大小为 327,680 单元的存储（640 × 512）。用 h_addr 的全部 10 位和 v_addr 的低 9 位 合成 19 位地址来索引显存。为方便寻址，我们给行 v_addr 分配了 512 行的空 间。这样，可以不用对地址进行复杂的转换。此处只需要分配 327680 个连续的 存储单元，不需要考虑 h_addr 大于 640 的情况。注意这里的像素数据其实是按 列连续存放的。

（二）实验原理

VGA 的工作原理

1. 分辨率：VGA 支持多种分辨率，其中最常见的是 640 x 480 像素。分辨率定义了屏幕上水平和垂直方向的像素数量。
2. 刷新率：VGA 显示器以固定的刷新率刷新屏幕内容，常见的刷新率是 60Hz。刷新率定义了在一秒钟内屏幕内容被重新绘制的次数。
3. 图像生成：计算机图形处理器（GPU）生成图像数据，并将其发送给 VGA 控制器。图像数据包含每个像素的颜色信息。
4. 色彩深度：VGA 支持不同的色彩深度，常见的是 8 位色彩深度，也就是每个像素用 8 位表示颜色。8 位色彩深度允许最多 256 种不同的颜色。
5. 水平和垂直同步信号：VGA 使用水平同步和垂直同步信号来同步计算机和显示器的操作。这些信号告诉显示器何时开始新的行和新的帧。
6. 模拟信号输出：VGA 通过 15 个针脚的连接器将模拟视频信号发送到显示器。这些信号包括红色、绿色和蓝色的模拟电压信号，用于控制每个像素的颜色。

当计算机启动时，GPU 生成图像数据，并通过 VGA 控制器将其转换为模拟视频信号。这个模拟信号通过 VGA 连接器发送到显示器。显示器接收信号后，将其转换为可见的图像并显示在屏幕上。

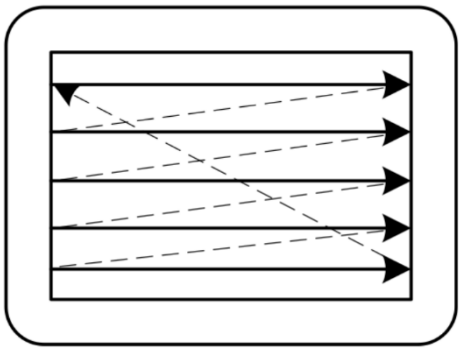


图 8-2: 显示器扫描示意图

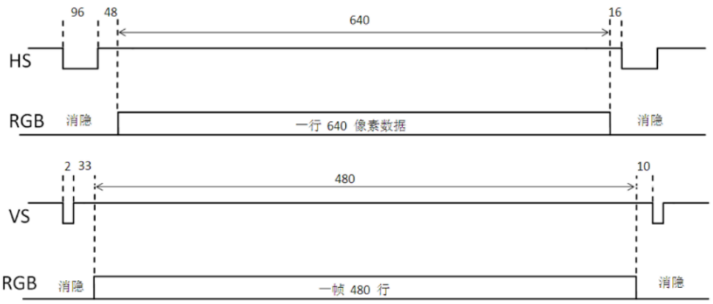


图 8-3: VGA 行扫描、场扫描时序示意图

（三）实验环境/器材等

硬件器材：Nexys A7-100T 开发板
软件平台：Vivado 开发平台

（四）实验过程

模型概述

通过实例化时钟模块、存储器模块和 VGA 控制模块，将各个模块连接在一起，实现对 VGA 显示器的控制和显示。

VGA 控制模块根据输入的开关信号和 VGA 数据，生成 VGA 控制信号和计算当前像素的坐标，同时根据计算结果和输入的颜色数据，设置输出的红、绿、蓝颜色信号，从而控制 VGA 显示器显示相应的图像。

数字抽象

1. 输入：
   - clk —— VGA 时钟信号
   - SW [1:0] —— SW [0] 复位信号，用于复位模块的状态
     —— SW [1] VGA 控制信号，用于控制 VGA 模块的操作
2. 输出：
   - LED [1:0] —— 用于显示状态指示
   - VGA_R [3:0] —— 当前像素点 R 颜色码字的输出
   - VGA_G [3:0] —— 当前像素点 G 颜色码字的输出
   - VGA_B [3:0] —— 当前像素点 B 颜色码字的输出
   - VGA_VS —— 列同步信号
   - VGA_HS —— 行同步信号

设计思路 & 设计代码

vga_ctrl.v

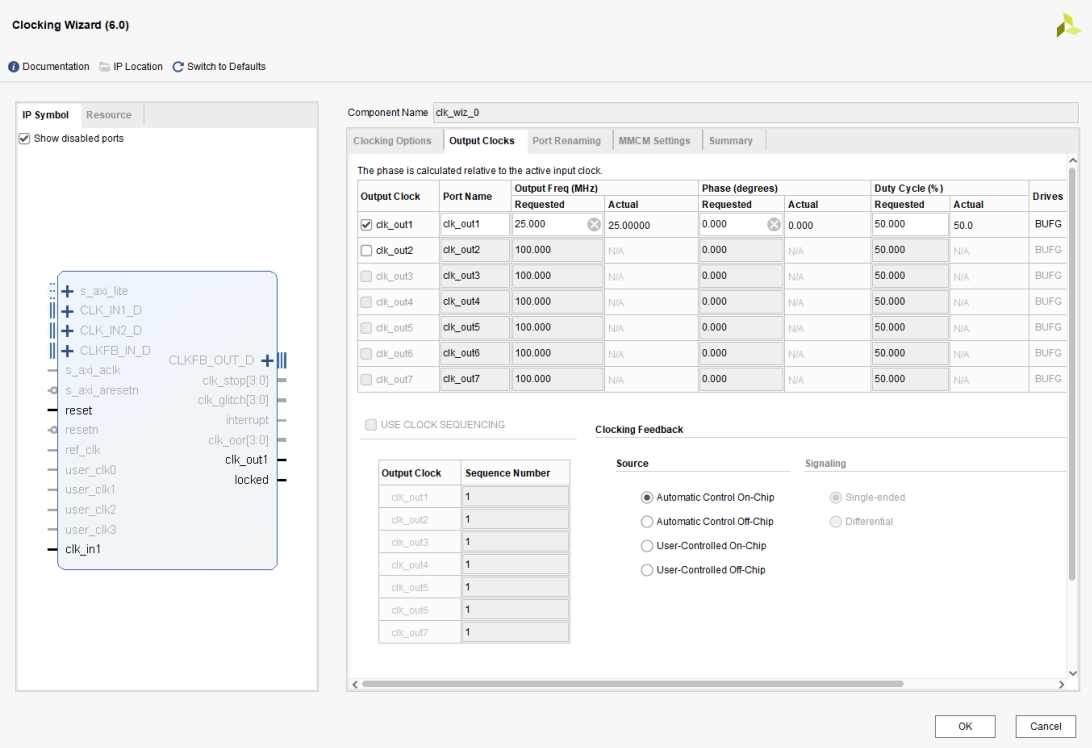
VGA 控制信号：利用行和列的扫描坐标，用传入的 VGA_DATA 改变相应的 vga_r，vga_g 及 vga_b。

```
//设置输出的颜色值
assign vga_r = valid ? vga_data[11:8] : 0;
assign vga_g = valid ? vga_data[7:4] : 0;
assign vga_b = valid ? vga_data[3:0] : 0;
```

稍微对此段代码进行修改：做判断。

IP 核生成

时钟

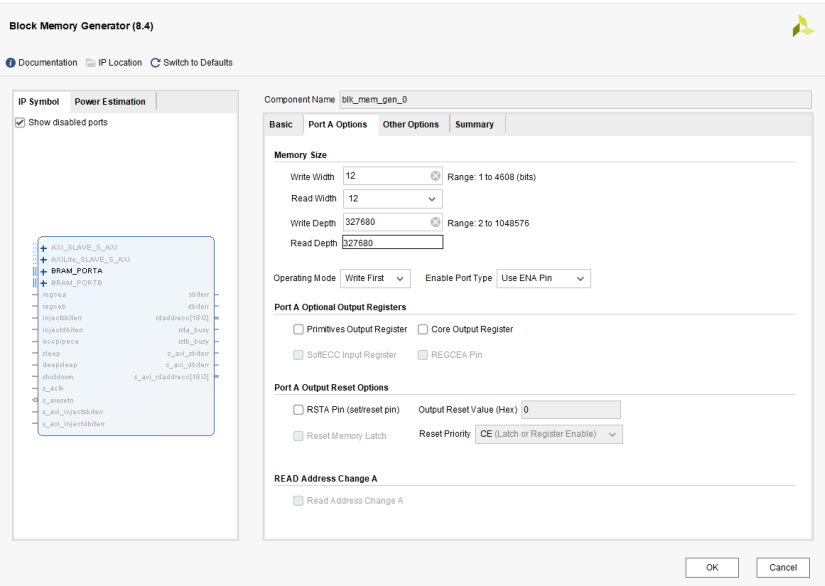


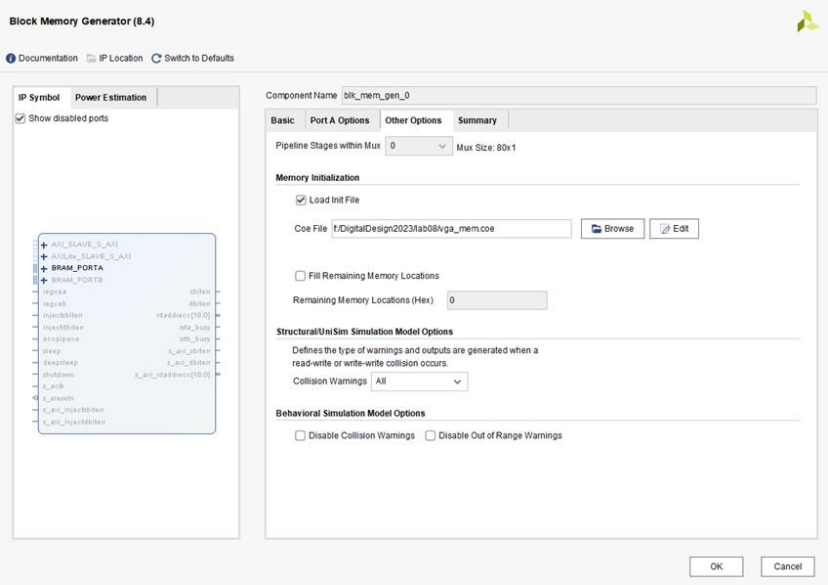
输入时钟为 clk_in1，设置为缺省的系统 100MHz 时钟。

输出时钟为 clk_out1，设置时钟的频率为 25MHz。

* Clocking Wizard 是预先开发好的分频模块，相比简单的计数器分频可以产生更加稳定的高质量时钟信号。

存储器





细节已在**（一）实验目的**说明。

lab08.v

顶层模块:

1. 定义了各种输入输出信号以及内部的中间信号。
2. 实例化了时钟模块 clk_wiz_0，用于产生 VGA 时钟信号 VGA_CLK。
3. 实例化了存储器模块 blk_mem_gen_0，用于存储 VGA 显示的颜色数据。
4. 实例化了 VGA 控制模块 vga_ctrl，用于生成 VGA 控制信号和计算像素坐标。
5. 通过 assign 语句，将像素坐标拼接为地址信号 addr。
6. 通过 assign 语句，将 VGA 控制信号拼接为 VGA 显示信号 VGA_VS 和 VGA_HS。

```
module lab08(
    input clk,
    input [1:0] SW,
    output [1:0] LED,
    output [3:0] VGA_R,
    output [3:0] VGA_G,
    output [3:0] VGA_B,
    output VGA_VS,
    output VGA_HS
    );

    wire VGA_CLK;
    wire [11:0] VGA_DATA;
    wire [9:0] H_ADDR;
    wire [8:0] V_ADDR;
    wire HSYNC;
    wire VSYNC;
    wire VALID;
    wire [18:0] addr;
```

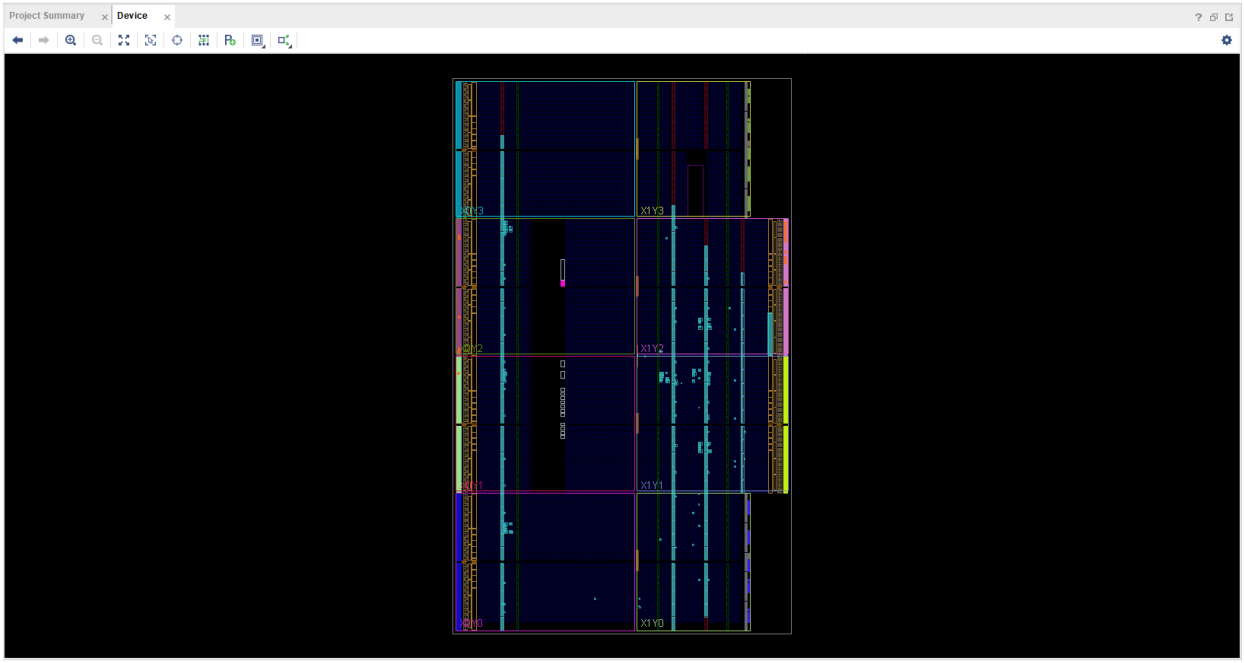
```
clk_wiz_0 myvgaclk(
    .clk_in1(clk),
    .reset(SW[0]),
    .locked(LED[0]),
    .clk_out1(VGA_CLK));

blk_mem_gen_0 myvgaram(
    .clka(VGA_CLK),
    .addra(addr),
    .dina(1'b0),
    .douta(VGA_DATA),
    .ena(1'b1),
    .wea(1'b0));

vga_ctrl myvga(
    .pclk(VGA_CLK),
    .reset(SW[1]),
    .vga_data(VGA_DATA),
    .h_addr(H_ADDR),
    .v_addr(V_ADDR),
    .hsync(HSYNC),
    .vsync(VSYNC),
    .valid(VALID),
    .vga_r(VGA_R),
    .vga_g(VGA_G),
    .vga_b(VGA_B));

assign addr = {H_ADDR, V_ADDR};
assign VGA_VS = VSYNC;
assign VGA_HS = HSYNC;
```

硬件实现（引脚分配）

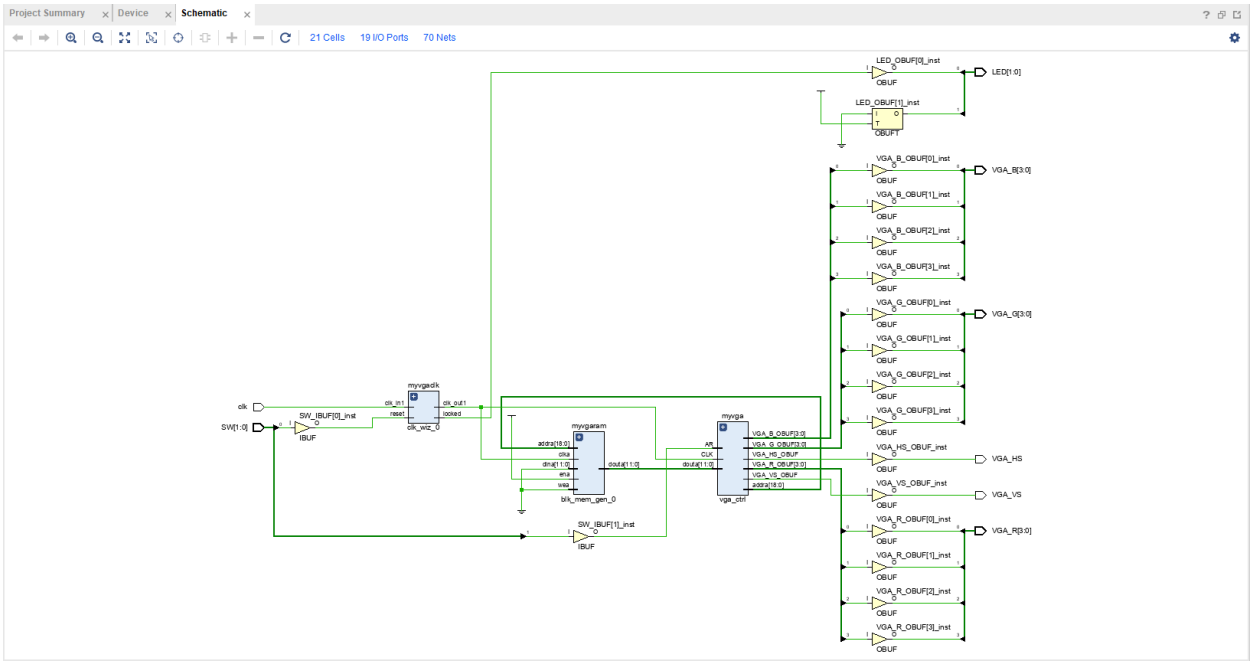
```
## Clock signal
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports { clk }]; #IO_L12P_T1_MRCC_35 Sch=clk100mhz
#create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {CLK100MHZ}];


##Switches
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { SW[0] }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { SW[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
#set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { SW[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
#set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { SW[3] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
#set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { SW[4] }]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
#set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports { SW[5] }]; #IO_L7N_T1_D10_14 Sch=sw[5]
#set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports { SW[6] }]; #IO_L17N_T2_A13_D29_14 Sch=sw[6]
#set_property -dict { PACKAGE_PIN R13   IOSTANDARD LVCMOS33 } [get_ports { SW[7] }]; #IO_L5N_T0_D07_14 Sch=sw[7]
#set_property -dict { PACKAGE_PIN T8    IOSTANDARD LVCMOS18 } [get_ports { SW[8] }]; #IO_L24N_T3_34 Sch=sw[8]
#set_property -dict { PACKAGE_PIN U8    IOSTANDARD LVCMOS18 } [get_ports { SW[9] }]; #IO_25_34 Sch=sw[9]
#set_property -dict { PACKAGE_PIN R16   IOSTANDARD LVCMOS33 } [get_ports { SW[10] }]; #IO_L15P_T2_DQS_RDWR_B_14 Sch=sw[10]
#set_property -dict { PACKAGE_PIN T13   IOSTANDARD LVCMOS33 } [get_ports { SW[11] }]; #IO_L23P_T3_A03_D19_14 Sch=sw[11]
#set_property -dict { PACKAGE_PIN H6    IOSTANDARD LVCMOS33 } [get_ports { SW[12] }]; #IO_L24P_T3_35 Sch=sw[12]
#set_property -dict { PACKAGE_PIN U12   IOSTANDARD LVCMOS33 } [get_ports { SW[13] }]; #IO_L20P_T3_A08_D24_14 Sch=sw[13]
#set_property -dict { PACKAGE_PIN U11   IOSTANDARD LVCMOS33 } [get_ports { SW[14] }]; #IO_L19N_T3_A09_D25_VREF_14 Sch=sw[14]
#set_property -dict { PACKAGE_PIN V10   IOSTANDARD LVCMOS33 } [get_ports { SW[15] }]; #IO_L21P_T3_DQS_14 Sch=sw[15]

## LEDs
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { LED[0] }]; #IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { LED[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]

##VGA Connector
set_property -dict { PACKAGE_PIN A3    IOSTANDARD LVCMOS33 } [get_ports { VGA_R[0] }]; #IO_L8N_T1_AD14N_35 Sch=vga_r[0]
set_property -dict { PACKAGE_PIN B4    IOSTANDARD LVCMOS33 } [get_ports { VGA_R[1] }]; #IO_L7N_T1_AD6N_35 Sch=vga_r[1]
set_property -dict { PACKAGE_PIN C5    IOSTANDARD LVCMOS33 } [get_ports { VGA_R[2] }]; #IO_L1N_T0_AD4N_35 Sch=vga_r[2]
set_property -dict { PACKAGE_PIN A4    IOSTANDARD LVCMOS33 } [get_ports { VGA_R[3] }]; #IO_L8P_T1_AD14P_35 Sch=vga_r[3]
set_property -dict { PACKAGE_PIN C6    IOSTANDARD LVCMOS33 } [get_ports { VGA_G[0] }]; #IO_L1P_T0_AD4P_35 Sch=vga_g[0]
set_property -dict { PACKAGE_PIN A5    IOSTANDARD LVCMOS33 } [get_ports { VGA_G[1] }]; #IO_L3N_T0_DQS_AD5N_35 Sch=vga_g[1]
set_property -dict { PACKAGE_PIN B6    IOSTANDARD LVCMOS33 } [get_ports { VGA_G[2] }]; #IO_L2N_T0_AD12N_35 Sch=vga_g[2]
set_property -dict { PACKAGE_PIN A6    IOSTANDARD LVCMOS33 } [get_ports { VGA_G[3] }]; #IO_L3P_T0_DQS_AD5P_35 Sch=vga_g[3]
set_property -dict { PACKAGE_PIN B7    IOSTANDARD LVCMOS33 } [get_ports { VGA_B[0] }]; #IO_L2P_T0_AD12P_35 Sch=vga_b[0]
set_property -dict { PACKAGE_PIN C7    IOSTANDARD LVCMOS33 } [get_ports { VGA_B[1] }]; #IO_L4N_T0_35 Sch=vga_b[1]
set_property -dict { PACKAGE_PIN D7    IOSTANDARD LVCMOS33 } [get_ports { VGA_B[2] }]; #IO_L6N_T0_VREF_35 Sch=vga_b[2]
set_property -dict { PACKAGE_PIN D8    IOSTANDARD LVCMOS33 } [get_ports { VGA_B[3] }]; #IO_L4P_T0_35 Sch=vga_b[3]
set_property -dict { PACKAGE_PIN B11   IOSTANDARD LVCMOS33 } [get_ports { VGA_HS }]; #IO_L4P_T0_15 Sch=vga_hs
set_property -dict { PACKAGE_PIN B12   IOSTANDARD LVCMOS33 } [get_ports { VGA_VS }]; #IO_L3N_T0_DQS_AD1N_15 Sch=vga_vs
```

（五）实验结果

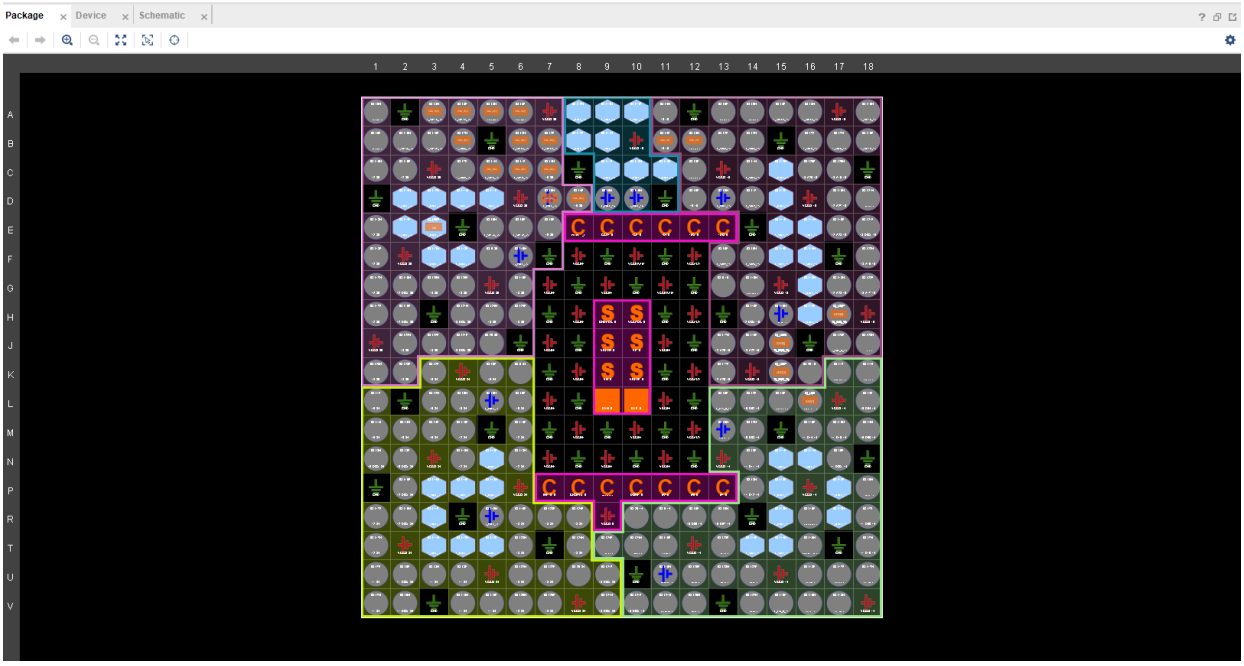
Device：



Schematic:



I/O Planning:



FGPA 结果：

成功显示!

当打开置位按钮时，黑屏!