基于 FPGA 的 VGA 图像显示与控制

课程要求：采用 verilog 语言，基于 FPGA 的 VGA 图像显示，即能够在显示器上实现动态彩色图像的显示。

VGA 显示接口的理论分析：硬件采用 CycloneII 系列的 EP2C20Q240C8N,它含有 240 个引脚。对于 VGA 的显示器遵循 800*600@75 模式，其中 800 是指每行中显示的像素的个数，而 600 是指屏幕每一列所包含的像素的个数。

VGA 工业标准规定了具体地，在扫描过程中的时序图如下:

行扫描时序图

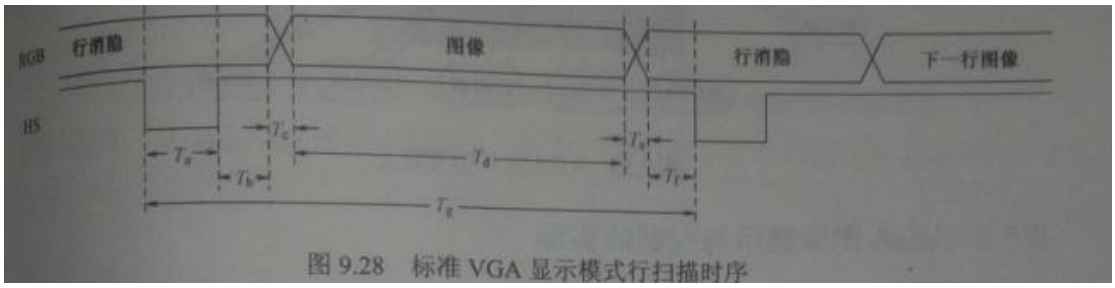


图 9.28　标准 VGA 显示模式行扫描时序

场扫描时序图

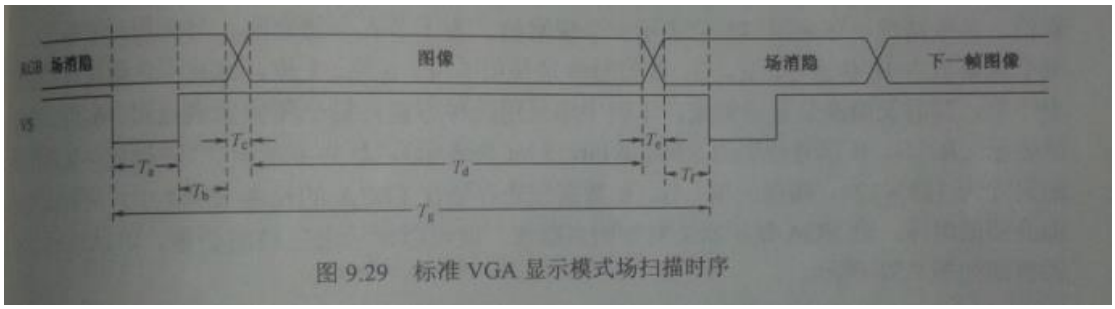


图 9.29　标准 VGA 显示模式场扫描时序

每场信号对应 625 个行周期，其中 600 行为图像显示行，每场有场同步信号，该脉冲宽度为 3 个行周期的负脉冲；每行显示行包括 1056 个点时钟，其中 800 点为有效显示区，每行有一个行同步信号的负脉冲，该脉冲宽度为 80 个点时钟。这样我们可以知道，行频为 625*75=46857Hz。需要的点时钟的频率为：625*1056*75=49.5MHz 约为 50MHz。由上图可知，实际上在真正的实现过程中，每一行扫描所花的时间实际上比显示一行的像素所花的时间多了 1056-800=256 个像素点。同理，每一场的扫描时间多了 625-600=25 个行时间。

设计思路：

我们采用 BmpToMif 工具把 BMP 格式的图像转换为.mif 文件。利用 QuartusII7.2 自带的 MegaWizard Plug-In Manager 产生一个 ROM 存储器，并用其来初始时.mif 文件。即将图像文件写入到存储器里面。然后利用编程来控制图像的显示。

设计步骤：

1、工程创建：创建一个 Project 取名字为 vga,在创建工程的向导中选着 CycloneII 系列的 EP2C20Q240C8。

2、代码书写：新建一个 Verilog HDL File 编写程序代码如下:

```
`timescale 1ns/1ps
module    tupian (clk,rst_n,hsync,vsync,vga_r,vga_g,vga_b,addr);
input clk,rst_n;
```

```
output      hsync,vsync,vga_r,vga_g,vga_b;
reg hsync,vsync;
output [14:0]      addr;
reg        [14:0]      addr;
reg        [10:0]      x_cnt;          //行坐标
reg        [9:0]       y_cnt;          //列坐标
parameter h_Ta=80,h_Tb=128,h_Tc=32,h_Td=800,h_Te=8,h_Tf=8,h_Tg=1056;
parameter v_Ta=3, v_Tb=14,v_Tc=7,v_Td=600,v_Te=0.8,v_Tf=0.2,v_Tg=625;

//-------------------  行场的计数-------------------
always @(posedge clk)
begin
      if(x_cnt == h_Tg-1) x_cnt<=0;
      else x_cnt<=x_cnt+1;
end

always @(posedge clk)
begin
      if(y_cnt==v_Tg-1) y_cnt<=0;
      else if ( x_cnt == h_Tg-1 )
         y_cnt<=y_cnt+1;
end

//-------------------同步信号产生-------------------
always @(posedge clk)
begin
      if(x_cnt<=h_Ta-1) hsync<=0;
      else hsync<=1;
end
always @(posedge clk)
begin
      if(y_cnt <= v_Ta -1) vsync <= 0;
      else vsync <=1;
end

//---------------有效显示区坐标--------------------
wire                  valid;
assign                valid = (x_cnt >= 11'd187) && (x_cnt <= 11'd987) && (y_cnt >=
10'd31) && (y_cnt <= 10'd631);

wire      [9:0]       xpos;
wire      [9:0]       ypos;

assign                xpos      =    x_cnt-11'd187;
```

```
assign                ypos     =    y_cnt-10'd31;

//-----------------显示图像----------------------

reg[27:0] k;
always @ (posedge clk )
begin
      if(k<=67108864)
     begin
           if((ypos >= 9'd100 && ypos <= 9'd229)&&(xpos >= 10'd65 && xpos <=
10'd192))
                       addr     <= (ypos-100)*128 + (xpos-65);

           else addr<=0;
     end

     else
      begin
           if((ypos >= 9'd100 && ypos <= 9'd229)&&(xpos >= 10'd573 && xpos <=
10'd700))
                       addr     <= (ypos-100)*128 + (xpos-65);
           else addr<=0;
     end

 if(k>134217728) k=0;
 else       k=k+1;


end

endmodule
```

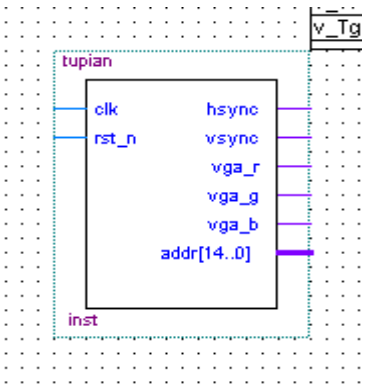
编写好程序后，点击保存，将文件起名为 tupian，并将其设置为顶层文件。点击 start compilation 按钮，进行汇编。

3、为 tupian.v 产生 Symbol file(符号文件)：

编译完成后，点击 File 菜单——Create/Update——Create symbol files for current file。

4、新建原理图文件，并导入产生的符号文件：

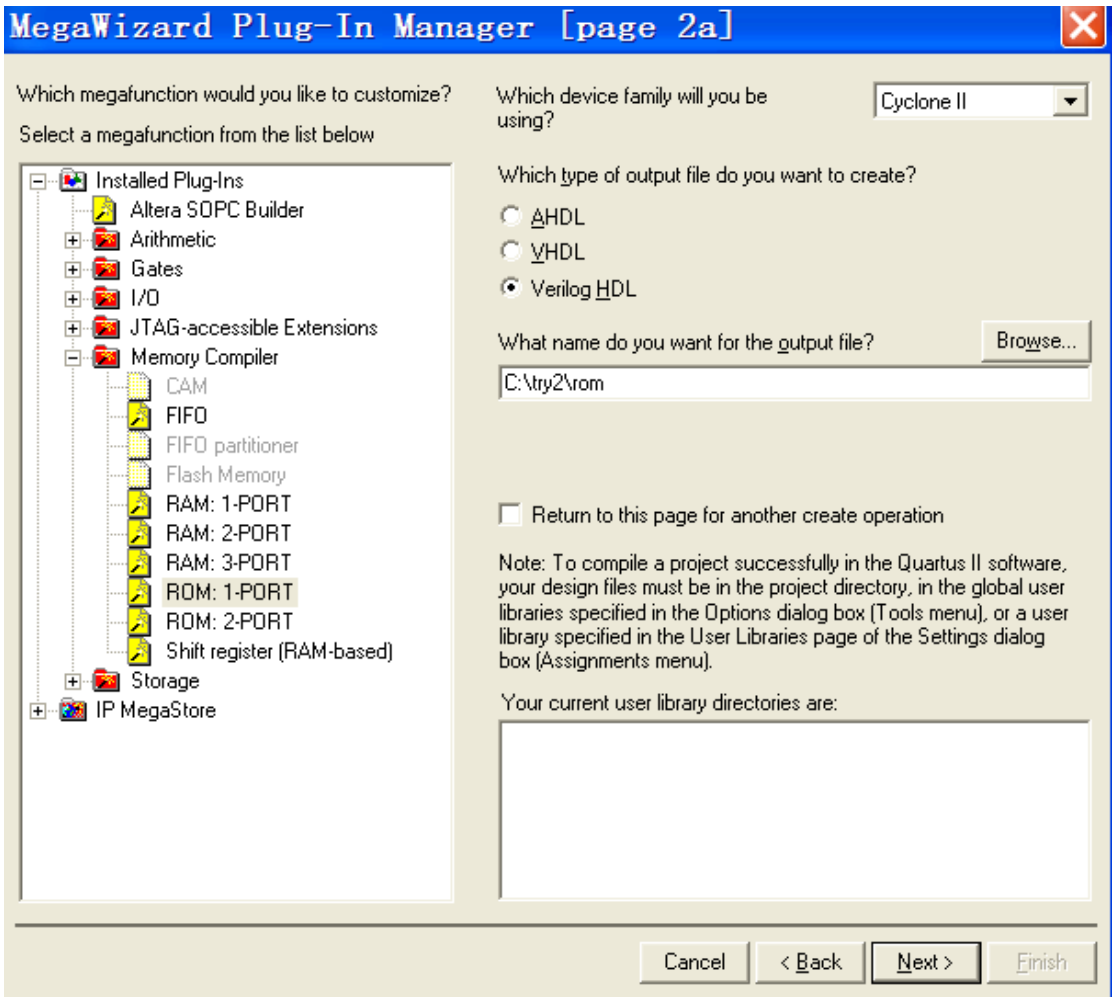
点击菜单 File——new——Block Diagram/Schematic File，双击原理图的空白处，将 Project 目录下的刚产生的符号文件引入到原理图。如下图：



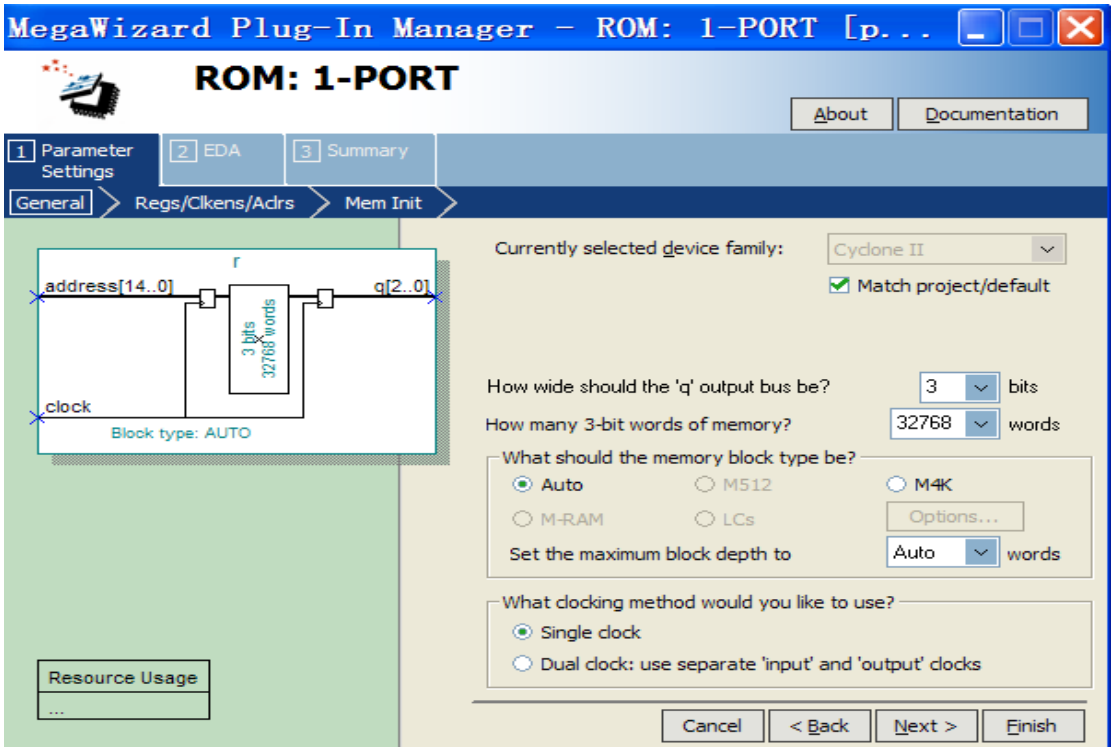
5、创建 ROM 并用它存储要显示的图片文件：

1）采用 BmpToMif 工具将 BMP 格式的图片转换为.mif 文件如下图，打开软件，打开要转换的图片，颜色类型选为彩色(8)色，点击生成 Mif 文件。给生产的文件起一个名字。

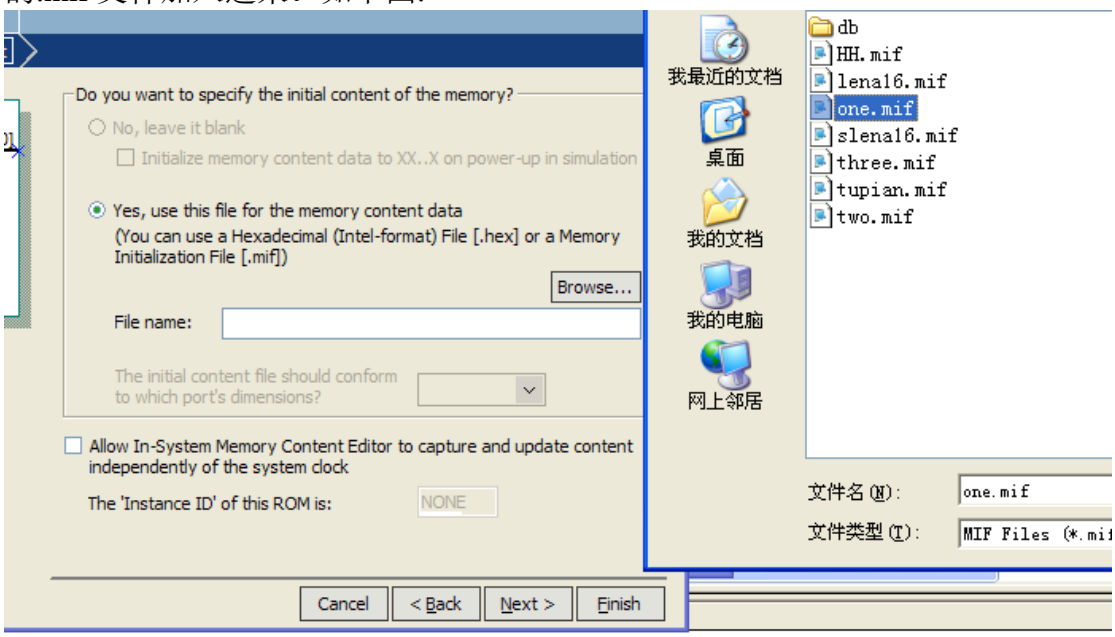
2）利用 Quartus II7.2 自带的 MegaWizard Plug-In Manager 产生一个 ROM 并将图片引入进去。双击原理图空白处，点击 MegaWizard Plug-In Manager——选择 Creat a new custom megafunction variation,在安装的插件中找到 Memory Compiler，选择 ROM：1-PORT，选择 Verilog 语言，并为输出的文件 起一个名字。如设置如下:



点击 Next，在弹出的窗口中，设置输出总线为 3 位，并为其分配的存储空间能够存储图片总的像素点的大小。其他默认。设置如下：

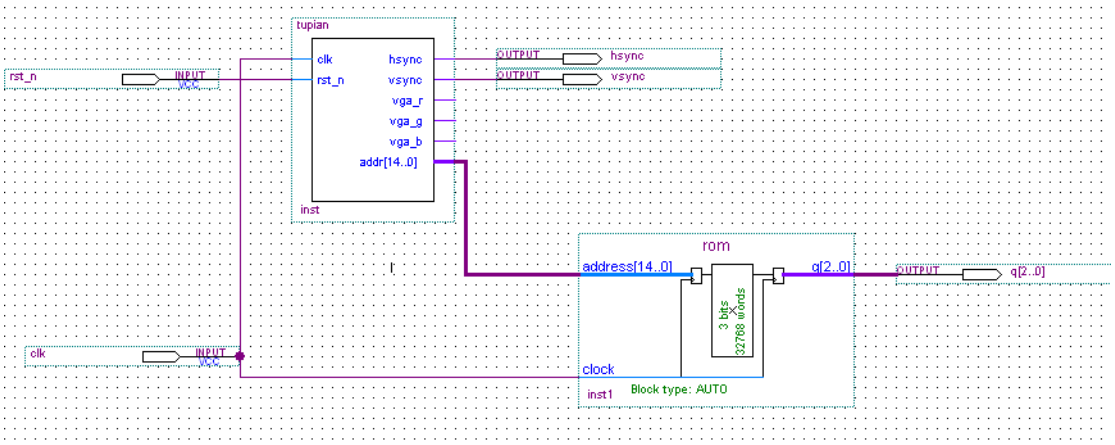


点击 Next，前面一直默认，走到 page 5 of 7,即第五步，点击 Browse,将之前产生的.mif 文件加入进来。如下图:



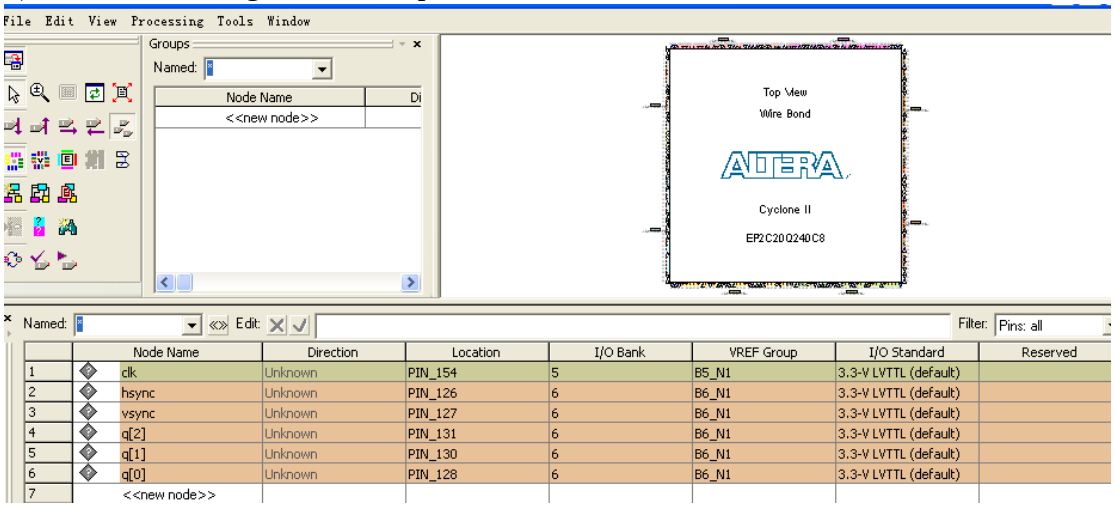
点击 Next，直至完成。

6、将 tupian 符号文件和刚生成的 Rom 进行连线。如下图:



7、引脚的绑定和设置

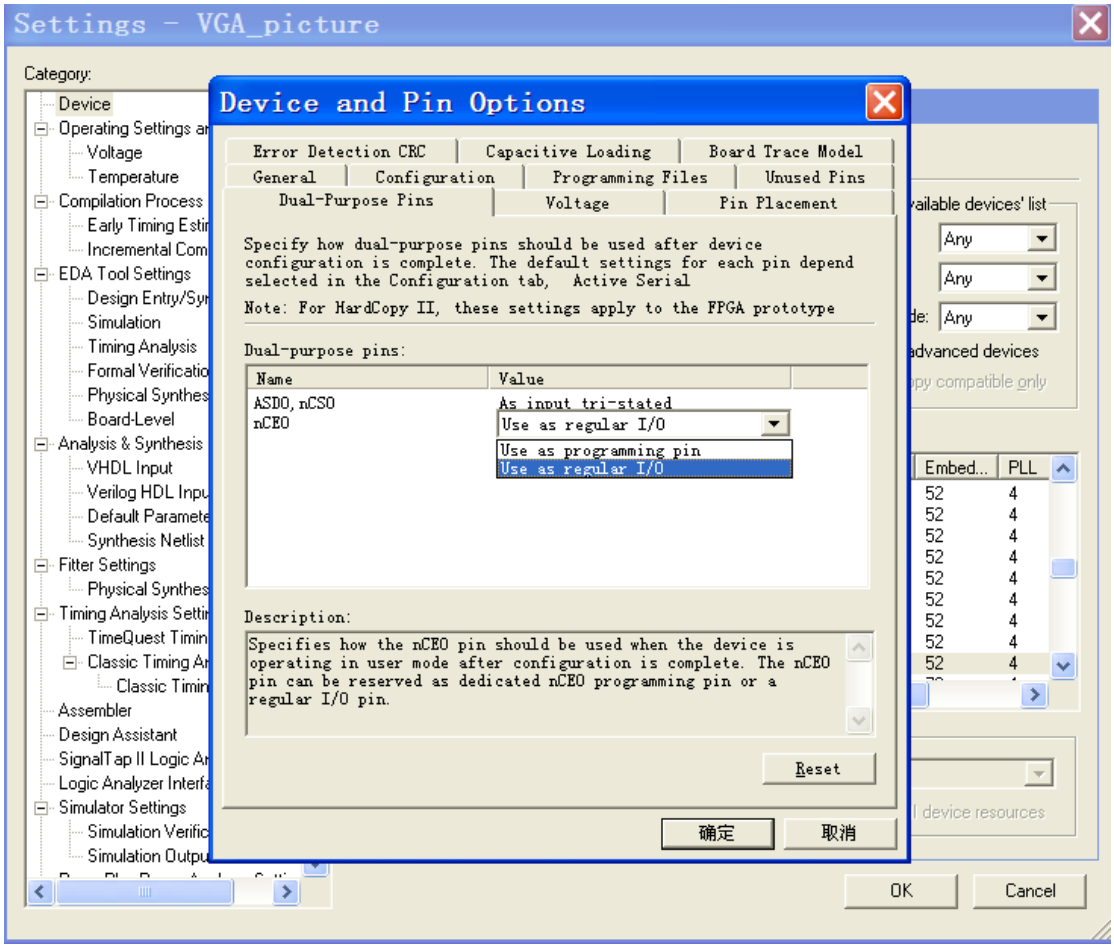
1) 点击菜单 Assignments——pin,为各引脚绑定如下:



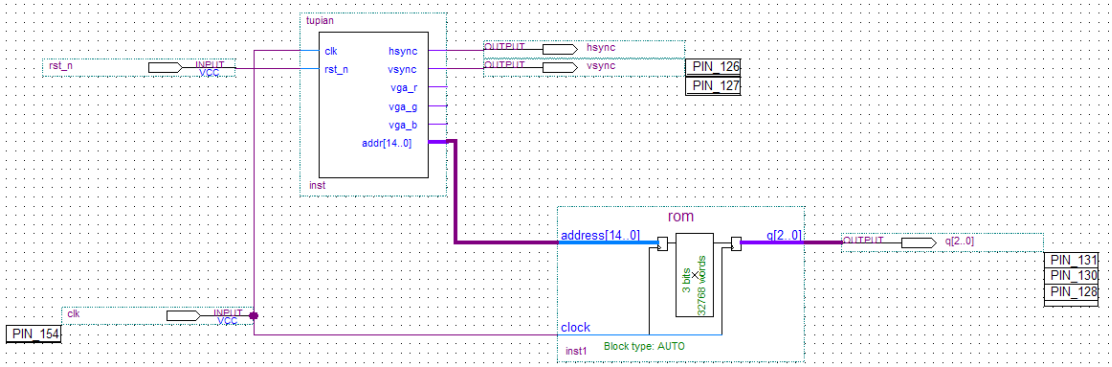
	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved
1	clk	Unknown	PIN_154	5	B5_N1	3.3-V LVTTL (default)	
2	hsync	Unknown	PIN_126	6	B6_N1	3.3-V LVTTL (default)	
3	vsync	Unknown	PIN_127	6	B6_N1	3.3-V LVTTL (default)	
4	q[2]	Unknown	PIN_131	6	B6_N1	3.3-V LVTTL (default)	
5	q[1]	Unknown	PIN_130	6	B6_N1	3.3-V LVTTL (default)	
6	q[0]	Unknown	PIN_128	6	B6_N1	3.3-V LVTTL (default)	
7	<<new node>>						

2) 将绑定的管脚设置为 I/O 复用：

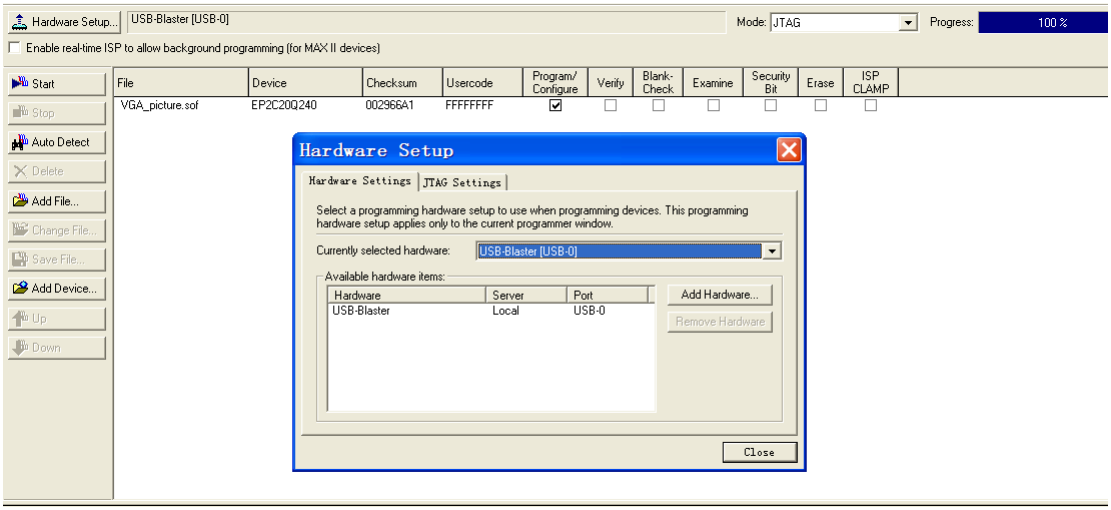
点击菜单 Assignments——Device——点击 Device and Pin Options，切换到标签 Dual-Purpose Pins，将 nCEO 的值设置为常规的 I/O 口。如下图:



8、将原理图文件设置成为顶层文件，然后点击编译运行，运行后，引脚绑定如下：



9、将编译生成的.sof 文件下载到硬件里:



实验结果：

实验效果如下图，彩色图片能够在两个位置进行动态显示，可以在代码中控制。