Lab6 VGA 接口实验

VGA（Video Graphics Array）视频图形阵列，现在通常指计算机的模拟显示标准、VGA 连接器或 640x480 分辨率本身，用于计算机输出显示器的接口标准。本次实验目的是掌握驱动 VGA 显示器的接口设计方法，掌握 VGA 接口显示图片、键盘输入字符和简易字符终端的设计方法。

一、实验目的

1) 掌握VGA显示器的接口设计方法。

2) 掌握键盘显示器接口ASCII码字符显示方法。

3) 掌握简易字符终端设计方法。

二、实验环境

1. Vivado 开发环境

2. Nexys A7-100T 实验板

三、实验原理

VGA 连接器是一种三排 15 针/孔的梯形接口，有 5 个主要信号，分别是：红 R、绿 G、蓝 B、水平同步/行同步 HS（Horizontal Synchronization）和垂直同步/帧同步 VS（Vertical Synchronization），如图 6.1 所示。VGA 信号是模拟信号，以电压值来表示颜色信号的强弱，一个红色输入信号 R 就能表示红色信号的强度信息。而 HS 和 VS 提供了该点应显示在屏幕上的参考位置。只有根据 VGA 不同分辨率下规范时序进行驱动这五个信号，才能在 VGA 监视器上显示所希望的图像。

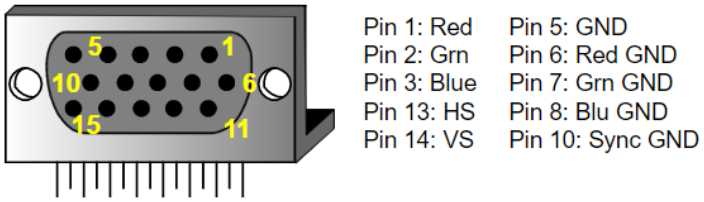

图 6.1 VGA 接口形状及信号示意图

显示器的分辨率是指在显示区域上的水平扫描次数所对应的“行”数，以及在每行上分配给像素的区域所对应的“列”数。分辨率有不同的标准，如 640×480、1024×768、1280×1024、1920×1080、3840×2160 等。现代的显示器上可以适应不同的分辨率，VGA 控制器电路通过定时信号来决定分辨率，不同分辨率的光束在显示器上的跟踪频率以及电子束的调制频率都不相同。

VGA 控制器电路必须生成 HS 和 VS 定时信号，并基于像素时钟协调显示数据的传送。像素时钟定义了显示一个像素信息的可用时间。VS 信号定义显示器的“刷新”频率，或重新绘制显示器上所有信息的频率。实际刷新频率在 50Hz 至 120Hz 范围内。

每一帧图像的显示都是从屏幕的左上角开始，然后自左向右、自上而下一行行依次显示。像素显示前，首先发送行同步信号 HS，行同步信号 HS 有效后，由 RGB 端送出当前行显示各像素点的 RGB 电压值；当一帧显示结束后，再发送帧同步信号 VS，重新开始从屏幕的左上端开始显示下一帧图像，如图 6.2 所示。

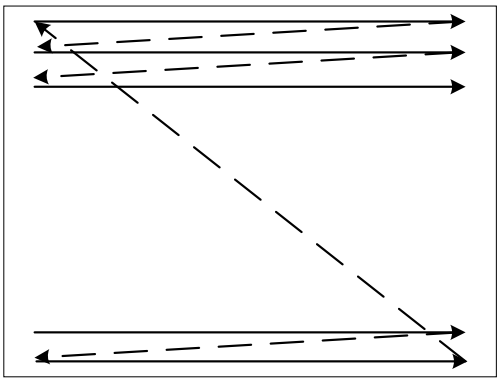

图 6.2　显示器扫描示意图

需要显示数据通常存放在显示存储器中，每个像素位置分配一个或多个字节，在 Nexys A7-100T 实验板中，每像素使用 12 位。当电子束在显示器上移动时，控制器必须读取到显示存储器中在给定像素位置上的显示数据，并将其输出到显示器 RGB 颜色信号端。

RGB 端并不是所有时间都在传送像素信息，由于电子束从上一行的行尾到下一行的行头需要时间，从屏幕的右下角返回到左上角开始下一帧也需要时间，这时 RGB 送的电压值为 0（黑色），这些时间称为电子束的行消隐时间和帧消隐时间，行消隐时间以像素为单位，帧消隐时间以行为单位。

不同分辨率的行、帧扫描时序各不相同，640×480 分辨率的行扫描、帧扫描时序示意图如图 6.3 所示，行和帧同步信号采用负脉冲信号。

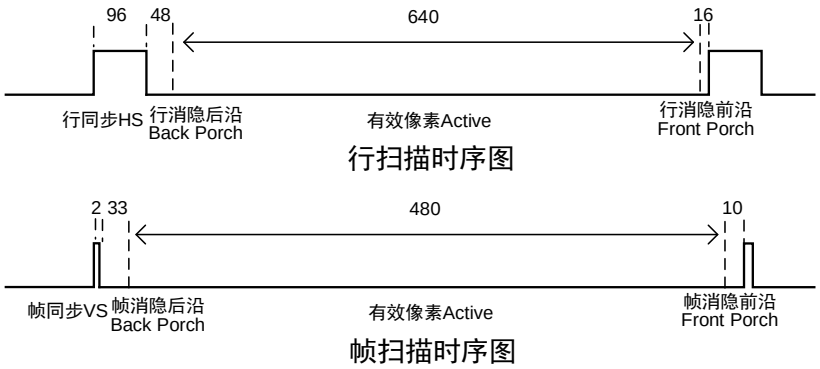

图 6.3 VGA 行扫描、帧扫描时序示意图

行同步信号 HS 宽度为 96 个像素，行消隐后沿（Back Porch）宽度是 48 个像素（包含了左边框），然后每行显示 640 个像素点，最后行消隐前沿（Back Porch）宽度是 16 个像素点（包含了右边框）。所以显示一行 640 个有效像素，共需要 96+48+640+16=800 个像素点时间，其中行消隐时间为 160 个像素点时间。

帧同步 VS 脉冲宽度为 2 个行显示时间，帧消隐后沿（Back Porch）需要 33 个行显示时间，然后每帧显示 480 行，帧消隐前沿（Back Porch）需要 10 个行显示时间。所以显示一帧 480 行有效像素，共需要 2+33+480+10=525 行时间，其中帧消隐时间为 45 行显示时间。

其他分辨率的像素时序如表 6.1 所示，不同像素时钟频率对应了不同的显示分辨率。

表 6.1 不同分辨率的像素时序表

显示模式	行同步信号时序（像素）					帧同步信号时序（像素）					像素时钟频率
	行同步	消隐后沿	有效像素	消隐前沿	行周期	帧同步	消隐后沿	有效像素	消隐前沿	帧周期	
640×480@60Hz	96	48	640	16	800	2	33	480	10	525	25.2Mhz
800×600@60Hz	128	88	800	40	1056	4	23	600	1	623	40MHz
1024×768@60Hz	136	160	1024	23	1344	6	29	768	3	806	65MHz
1280×720@60Hz	40	220	1280	110	1650	5	20	720	5	750	74.25Mhz
1280×1024@60Hz	112	248	1280	48	1688	3	38	1024	1	1066	108MHz
1920×1080@60Hz	44	148	1920	88	2200	5	36	1080	4	1125	148.5MHz

1、VGA 接口

Nexys A7-100T 开发板上使用不同阻值的电阻来将 FPGA 输出的数字信号转换到 VGA 模拟信号，对于 RGB 每种颜色提供 4 位二进制数的亮度信息，每个像素用 12 位二进制数来表示，连接方式如图 6.4 所示。例如开发板提供 RED[3:0]，当所有比特均为 0 时，合成的模拟红色信号电压为 0，亮度最低。如果所有比特都为 1 时，合成电压为 0.7V（VGA 信号线自身内阻为 75 欧姆），红色亮度最高。不同比特位的权重通过连接的电阻大小来确定，因此，通过 4 位的 RED[3:0]可以产生 16 种不同的红色亮度。这样红、绿、蓝三种颜色可以组合 4096 种不同颜色，实现较好的显示效果。

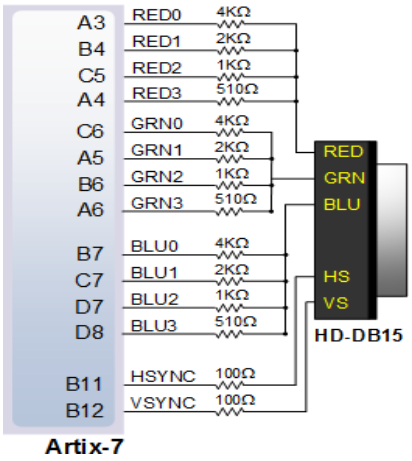

图 6.4　Nexys A7-100T 的 VGA 连接示意图

在 FPGA 开发板下显示接口的流程一般分为下列几步：

首先，根据分辨率的大小设置像素时钟频率。其次，计算像素在行和列中位置，生成行同步和帧同步信号，确定该像素位置是否属于有效像素区域。第三，如果属于有效像素区域，则读取显示存储器中该像素的数据或者直接生成该像素的数据。最后，把同步信号和该像素三种颜色分量的数据送到显示器接口中，如图 6.5 所示。

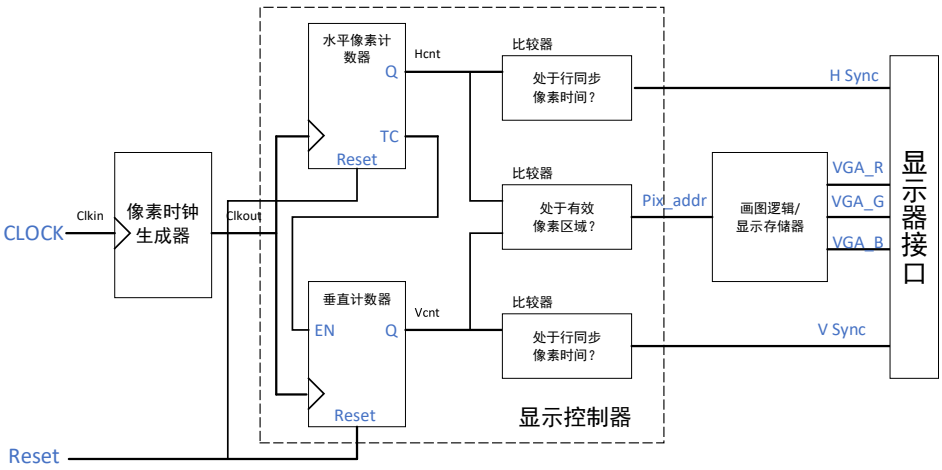

图 6.5　显示器中图像显示的流程

其中计算像素位置，生产同步信号的过程通常称为显示控制器，读取和生成像素数据的过程称之为画图逻辑或直接访问显示存储器。

下面将通过在显示器屏幕上显示不同色彩的图案来演示 VGA 接口的设计方法。

像素时钟频率

假设设置显示器分辨率为 1280×1024，刷新频率为 60Hz，实际行周期是 1688 个像素，列周期是 1066 行，则每秒中需要处理 1688×1066×60 = 107964480 个像素，因此像素点的时钟频率是 108MHz，显示一帧图像的时钟周期约为 16.8 毫秒。可以直接使用实验板上提供的 100MHz 的时钟频率作为像素频率。

显示控制器

显示控制器使用两个计数器，利用时钟信号对当前像素的水平和垂直位置进行计数，判断当前像素所处的位置，根据 1280×1024 分辨率像素时序规范的要求，判断该像素是否处于消隐时间还是在有效显示时间，并生成水平和垂直同步信号。

VGACtrl 模块实现显示控制器的功能，其参考代码如下：

```
module VGACtrl(
      input     wire                pix_clk,       //像素时钟信号
      input     wire                pix_rst,       //复位信号
      output    wire    [11:0]      pix_x,         //像素在可显示区域中的水平位置
      output    wire    [11:0]      pix_y,         //像素在可显示区域中的垂直位置
      output    wire                hsync,         //水平同步信号
      output    wire                vsync,         //垂直同步信号
      output wire                   pix_valid      //像素在可显示区域标志
);
//hsync1688         1280×1024@60Hz
    parameter       H_Sync_Width = 112;
    parameter       H_Back_Porche = 248;
    parameter       H_Active_Pixels =1280;
    parameter       H_Front_Porch = 48;
    parameter       H_Totals = 1688;
 //vsync1066
    parameter       V_Sync_Width = 3;
    parameter       V_Back_Porche = 38;
    parameter       V_Active_Pixels =1024;
    parameter       V_Front_Porch = 1;
    parameter       V_Totals = 1066;
reg         [11:0]    cnt_h           ;
reg         [11:0]    cnt_v           ;
wire                  rgb_valid     ;
```

```
  //同步信号，低电平有效
assign hsync = ((cnt_h >= H_Sync_Width)) ? 1'b0 : 1'b1;       //大于水平同步像素，hsync=0
assign vsync = ((cnt_v >= V_Sync_Width)) ? 1'b0 : 1'b1;       //大于帧同步像素，vsync=0
//cnt_h，cnt_v 像素位置计数
always@(posedge pix_clk) begin
            if (pix_rst) begin
                cnt_h <= 0;
                cnt_v <= 0;
            end
          if (cnt_h == (H_Totals-1)) begin                          // 行像素结束
                cnt_h <= 0;
                cnt_v <= (cnt_v == (V_Totals-1)) ? 0 : cnt_v + 1;   // 帧像素结束
              end
            else begin
                    cnt_h <= cnt_h + 1;
                end
  end
    //pix_valid=1，表示像素处于有效显示区域
  assign    pix_valid = (((cnt_h >= H_Sync_Width +H_Back_Porche)
                          && (cnt_h <= H_Totals- H_Front_Porch))
                          &&((cnt_v >= V_Sync_Width +V_Back_Porche)
                          && (cnt_v <= V_Totals - V_Front_Porch)))
                          ? 1'b1 : 1'b0;
  //Hsync,Vsync active，计算像素在可显示区域的位置
    assign pix_x = (pix_valid==1) ? (cnt_h - (H_Sync_Width + H_Back_Porche)):12'h0;
    assign pix_y = (pix_valid==1) ? (cnt_v - (V_Sync_Width + V_Back_Porche)):12'h0;
endmodule
```

该控制器代码对上层模块提供了显示接口需要的同步信号 hsync 和 vsync，并提供了处于可显示区域时的像素坐标 pix_x 和 pix_y，上层模块可以根据当前像素的坐标，选择合适的数值赋予该像素数据，也可以在上层模块分配一块显示存储器，利用 pix_x 和 pix_y 来索引该显存，每次扫描到特定像素点时，按照读取出显存的值来设置像素数据。这样，其他应用就可以直接对显存进行操作，显存改变则自动反应到显示器的屏幕上，而不用关心具体的扫描过程。

画图逻辑

根据显示控制器输出的像素坐标，读取或赋值每个像素的颜色数据，并按照红 R、绿 G、蓝 B 三种颜色的数据格式进行输出，则可以在显示器的像素坐标上显示该像素的颜色。

VGADraw 画图模块实现通过像素点的坐标位置来确定不同的显示颜色，其参考代码如下：

```
module VGADraw(
    input     wire                pix_clk    ,
//     input     wire                pix_rst ,
```

```
        input     wire       [11:0]     pix_x    ,
        input     wire       [11:0]     pix_y    ,
        input     wire                  pix_valid,
        output    wire        [11:0]    pix_data
    );
    reg [27:0] cntdyn;
    reg [7:0] temp_r,temp_g,temp_b,temp_d;
    always@(posedge pix_clk )begin
        cntdyn<=cntdyn+1;
        temp_d <=cntdyn>>20;
        temp_r<=-pix_x-pix_y-temp_d;
        temp_g<=pix_x-temp_d;
        temp_b<=pix_y-temp_d;
    end
    assign    pix_data[11:8]=temp_r[7:4];
    assign    pix_data[7:4]=temp_g[7:4];
    assign    pix_data[3:0]=temp_b[7:4];
endmodule
```

为了显得色彩丰富，模块中通过像素频率产生动态计数 cntdyn，读取 cntdyn 高 8 位，并和像素坐标位置 pix_x 和 pix_y 进行运算，分别产生红色、绿色和蓝色的信号值。

VGASim 模块在 VGA 显示器上显示图像。

```
    module VGASim(
        input CLK100MHZ,              //系统时钟信号
        input    BTNC,                //  复位信号
        output [3:0] VGA_R,        //红色信号值
        output [3:0] VGA_G,        //绿色信号值
        output [3:0] VGA_B,         //蓝色信号值
        output    VGA_HS,              //行同步信号
        output    VGA_VS               //帧同步信号
     );
    wire [11:0] vga_data;
    wire valid;
    wire [11:0] h_addr;
    wire [11:0] v_addr;
    VGACtrl vgactrl(.pix_x(h_addr),.pix_y(v_addr),.hsync(VGA_HS),.vsync(VGA_VS),
                    .pix_valid(valid),.pix_clk(CLK100MHZ),.pix_rst(BTNC));
    VGADraw vgadraw(.pix_data(vga_data),.pix_x(h_addr),.pix_y(v_addr),.pix_valid(valid),
                      .pix_clk(CLK100MHZ));
    assign VGA_R=vga_data[11:8];
    assign VGA_G=vga_data[7:4];
    assign VGA_B=vga_data[3:0];
```

endmodule

添加 VGASim.xdc 约束文件，可在显示器上展示如图 6. 所示的图案，可以修改画图逻辑模块中像素点数据生成算法，则可显示不同的图案。

2、显示静态图像

在计算机系统中，像素点的颜色数据一般都是从显示存储器（显存）中直接读取的。因此需要事先将数据保存到显示存储器中。Nexys A7-100T 实验板中每种颜色接口宽度是 4 位，每个像素需要 12 位，存储一幅 640×480 分辨率的图像需要 3.686Mb 存储空间；如果图像的分辨率为 800×600，则需要 5.76Mb 的存储空间。

实验板中的 FPGA 芯片带有块存储器 Block RAM，存储空间为 4.86Mb，可以用作分辨率为 640×480 的显示存储器。

利用块存储器实现显示存储器的功能，并加载分辨率为 640×480 的图像，可以通过 Vivado 提供的 IP 核来实现。具体方法如下：

1、在 IP Catalog 的搜索框中输入 ram，在显示结果中双击 Block Memory Generator 所在行，弹出分布式块存储的属性框，如图 6.6 所示。

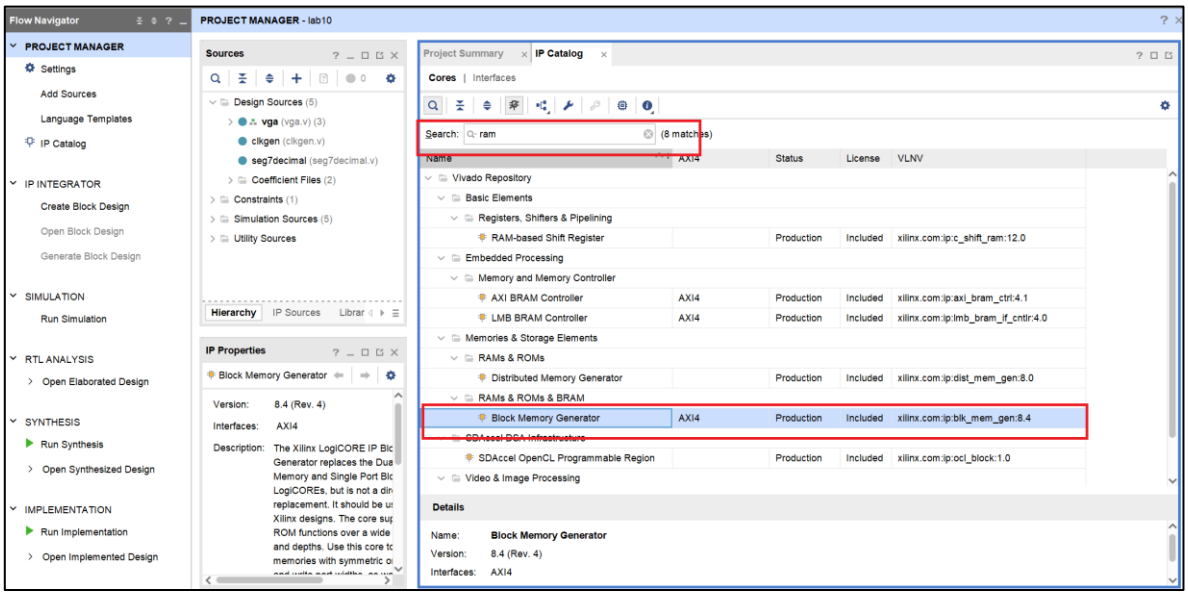

图 6.6 块存储器的 IP 核

2、在弹出的 RAM 存储器的 IP 核属性框中，设置组件名称为 vga_mem，设置存储器类型为单端口 RAM，如图 6.7 所示。

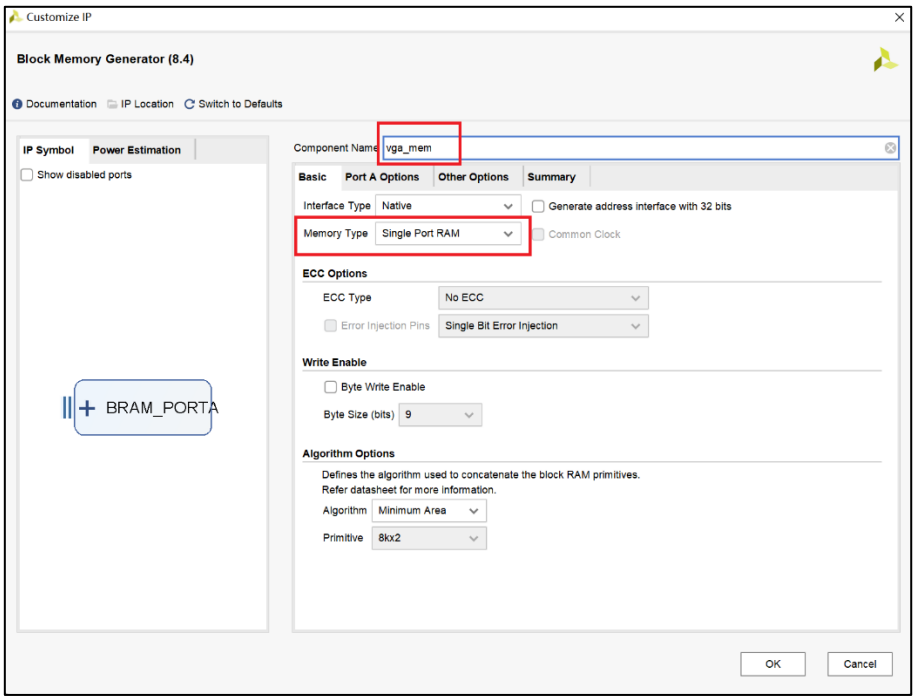

图 6.7　设置块存储器的名称和类型

3、点击 Port A Options 对话框，每个像素存储 12 位的数据，则设置 Write Width 的参数为 12；显示图像的分辨率为 640×480，则需要存储的像素共计为 307200 个单元，参数设置如图 6.8 所示。

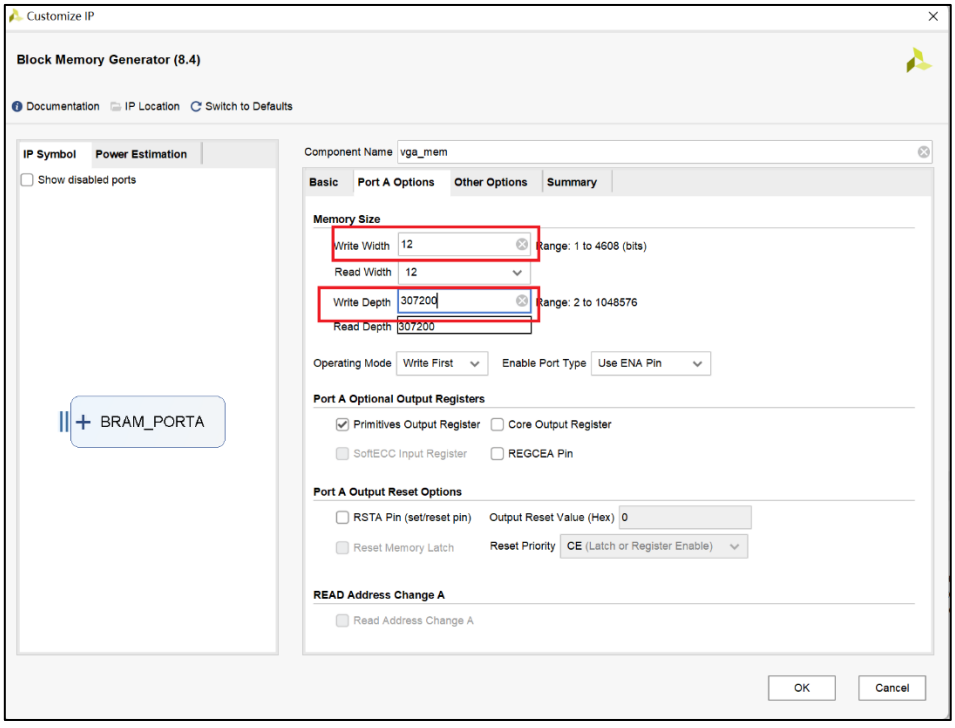

图 6.8　设置块存储器存储空间

4、点击 Other Options 对话框，在内存初始化下，选择装载内存初始化文件，选择实验提供的图像 COE 文件，如 img640x480-h.coe 文件，并点击 Edit 按钮，在弹出的对话框中，点击 Validate 按钮，显示校验成功信息框，如图 6.9 所示，点击 OK 按钮。

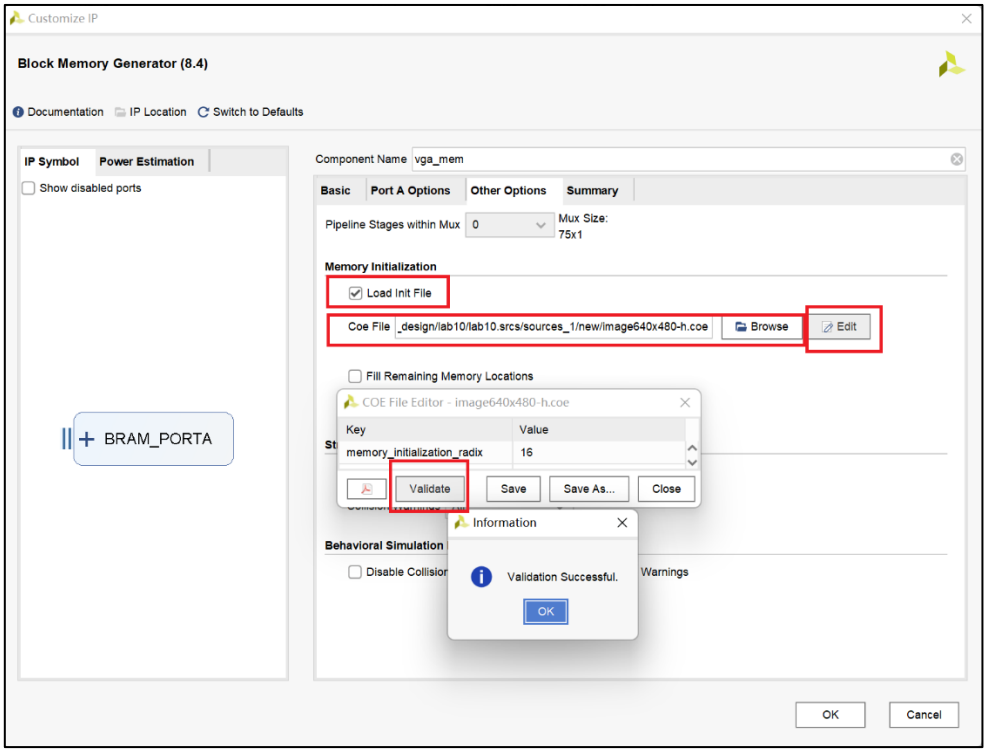

图 6.9　装入提供的图像文件

COE 文件是一种 ASCII 文本文件，文件头部定义数据基数（Radix），可以是 2、10 或 16。数据以向量的形式给出，向量之间用逗号或空格隔开，向量以分号或者回车换行符结束。

COE 文件格式要求如下所示：

memory_initialization_radix=16;

memory_initialization_vector=

47b,47b,47b,47b,47b,47b,46b,46b,36b,36b,36b,36b,,…，223;

需要提醒的是，在使用 COE（Coefficient）文件来传递图像数据时，需考虑 COE 文件中像素数据的排列次寻是行优先还列优先。实验中提供的 img640x480-h.coe 文件是行优先排列，则在生成显存读取地址时要注意计算格式；image640x480-v.coe 则是按照列优先排列。

可以使用 MATLAB 或其他编程语言实现图像文件到 COE 文件的转换，Vivado 会解析 COE 文件格式，并在生成 IP 核时导出相关的 MIF 格式。

5、点击 Generate 按钮，生成块存储器 IP 核的接口文件 vga_mem.xci，如图 6.10 所示。

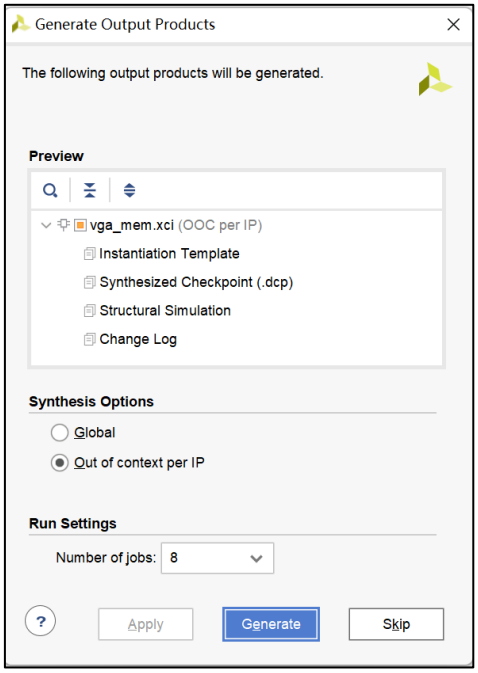

图 6.10　生成块存储器 IP 核的接口信息文件

6、创建 IP 核引用文件后，在项目源文件中增加一项 IP 核文件 vga_mem，如图 6.11 所示。

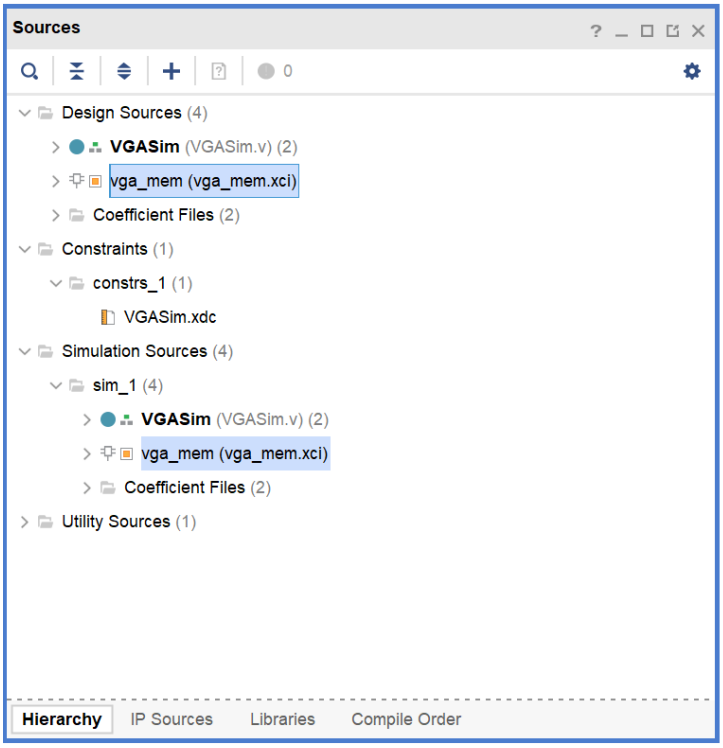

图 6.11　项目源文件中增加了 IP 核的接口文件

7、点击“IP Sources”按钮，IP 核接口文件 vga_mem 下，点击实例模板 Instantiation Template，选择 vga_mem.veo 文件，显示在 Verilog 语言下 vga_mem 文件实例引用接口模板，如图 6.12 所示。

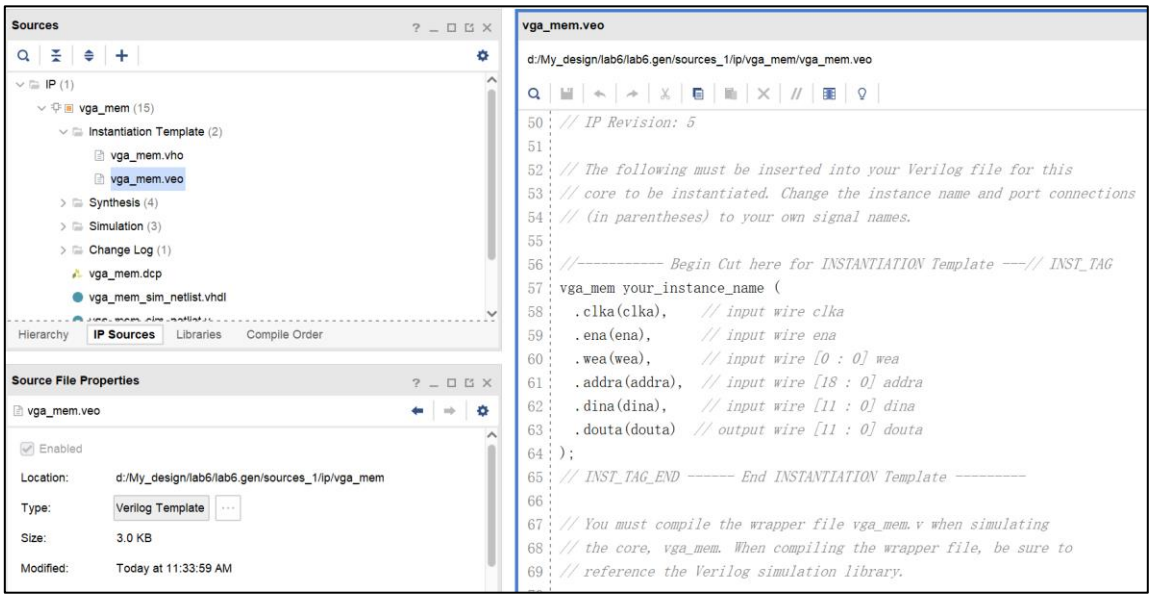

图 6.12　IP 核文件 vga_mem 的实例引用模板

通过引用 vga_mem，来读取显存中的数据，在画图逻辑模块 VGADraw 中，首先根据 COE 文件中像素数据的排列方法计算显存地址 ram_addr，然后添加引用 vga_mem 的实例语句，通过显存地址读取指定像素位置的数据，如下所示：

```
vga_mem my_pic(.clka(pix_clk),.ena(1'b1),.wea(1'b0),.addra({ram_addr}),.dina(12'd0),.douta(pix_data));
```

显示器上显示该图像，需要还需要修改显示控制器模块 VGACtrl 中像素时钟频率、分辨率参数等。

VGA 标准 640×480，刷新频率设置为 60Hz，每秒中需要处理 800×525×60=25200000 个像素，因此像素点的时钟信号频率是 25.2MHz，考虑到误差和处理方便，一般设置为 25MHz，可以通过实验板中提供的 100MHz 的时钟信号进行分频来产生。如果需要更为精准的时钟信号，则通过引用 Vivado 中提供时钟 IP 核来实现。

3、显示 ASCII 字符

在 VGA 屏幕上显示 ASCII 字符，其所需的资源比较少。标准 ASCII 字符用 7 位二进制数表示共 128 个字符，某些情况下，使用 8 位二进制数表来示扩展的 ASCII 码共 256 个字符。在屏幕上显示 ASCII 字符，需要在显示系统中预先存储这 256 个字符的字模点阵，以像素数组（或者位图）的形式保存每个字形。典型的 ASCII 字符字模有 7*5、12*8、16*8、16*12 等。点阵字库具有读取方便的优点，但在缩放或其他转换时，会出现失真的情形。目前在内存和处理能力限制的数字系统中仍在使用点阵字体，以便提升显示速度和处理的简便性。

本次实验提供了 16×8 的 ASCII 码字库文件，每个字符字模 16 行，每行 8 个像素点，对应的像素点为“1”时显示白色，为“0”时显示黑色。这样保存一个字符的字模需要 16 个字节，保存 256 个字符共需 4096 个字节，如图 6.13 所示。

图 16.13　ASCII 字符字模

实验资源中的 ASC16-96.txt 文件提供了一个 96 个可见 ASCII 码字符的点阵文件。每个字符用 16 个字节的数据表示。如：字符“b”的对应的字模点阵是：{0x00, 0x00, 0x70, 0x30, 0x30, 0x3C, 0x36, 0x33, 0x33, 0x33,0x33,0x6E,0x00,0x00,0x00,0x00}。用 0 和 1 表示的字形如下：

0x00 点阵表示为 00000000；

0x00 点阵表示为 00000000；

0x70 点阵表示为 01110000；

0x30 点阵表示为 00110000；

0x30 点阵表示为 00110000；

0x3C 点阵表示为 00111100；

0x36 点阵表示为 00110110；

0x33 点阵表示为 00110011；

0x33 点阵表示为 00110011；

0x33 点阵表示为 00110011；

0x33 点阵表示为 00110011；

0x6E 点阵表示为 01101110；

0x00 点阵表示为 00000000；

0x00 点阵表示为 00000000；

0x00 点阵表示为 00000000；

0x00 点阵表示为 00000000；

数字“1”表示了字符“b”的字形。

提示：在使用字库文件时，要注意字形数据的高低与显示次序是否一致；如果不一致，则需要在显示之前进行位置转换。

实验资源中的 ASC16 文件提供了 256 个 ASCII 字符 16×8 点阵字库文件（4KB），不可识别的字符使用图形来表示。

有了字符字模点阵后，显示控制器就可以不再记录屏幕上每个像素的颜色信息，只需要记录屏幕上显示的 ASCII 字符即可。在显示时，根据当前光标位置，确定需要显示的字符，根据字符查找对应的字模点阵，再把点阵数据显示在屏幕上。对于分辨率为 640×480 的显示器，可以显示 30 行（480÷16=30），80 列（640÷8=80）的 ASCII 字符。数字系统的显存只需要 30×80=2400 个存储单元，每个存储单元存放 1 个 ASCII 码。保存一帧显示字符只需要 2.4KB，加上 ASCII 字符字模点阵 4KB，比保存一帧图像数据小多了。

在 VGA 控制模块中输出当前有效像素点（640×480）的行和列位置信息，需要修改为输出当前位置到对应字符阵列（80×30）的坐标位置。通过坐标位置查询字符显存得到当前坐标位置的字符 ASCII 码；根据 ASCII 码，查询对应的字符字模点阵库文件；再根据点阵数据，转换成像素点的行和列信息，可以知道当前扫描到的是字符内的哪个点；再根据该点对应的位是 1 还是 0，选择颜色输出白色还是黑色，如图 6.14 所示。

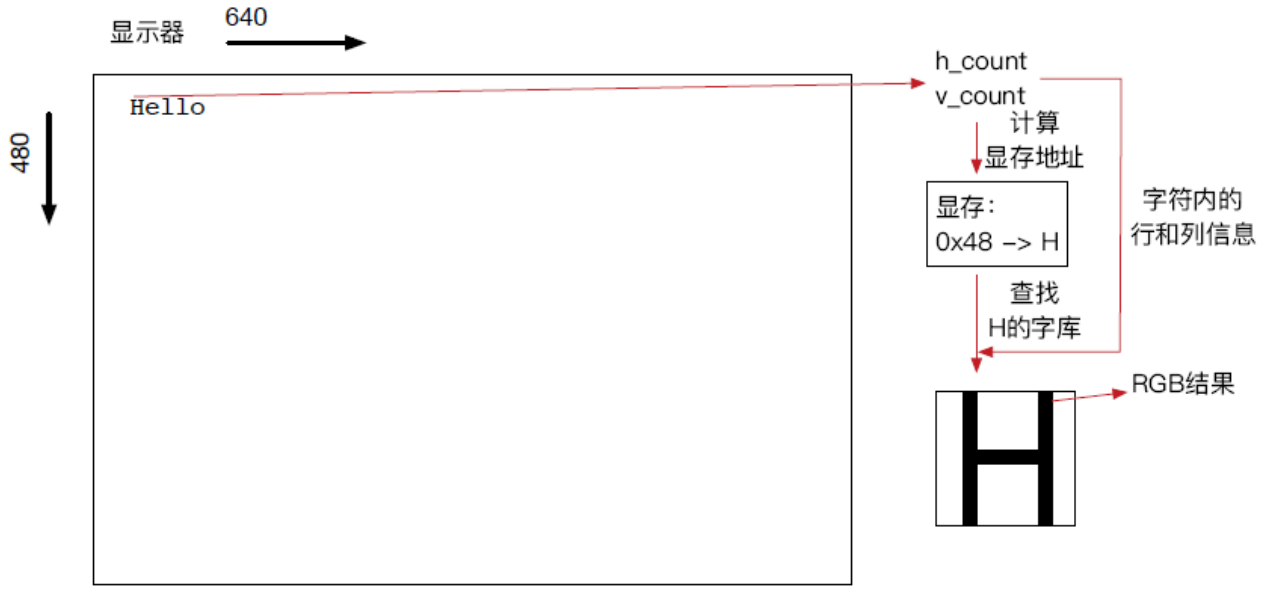

图 6.14 字符显示流程示意图

字符显示的过程如下：

1） 根据当前像素点位置，获取对应的字符的 x,y 坐标，以及像素点到单个字符点阵内的行列信息；

2） 根据字符的 x,y 坐标，查询字符显存，获取对应 ASCII 码；

3） 根据 ASCII 码和字符内的行信息，查询字模点阵文件，获取对应行的 8 位二进制数据；

4） 根据字符内的列信息，取出对应的位，并根据该位的数值设置颜色。此处可以显示黑底白字或其他彩色字符，只需要按需求分别设置背景颜色和字符颜色即可。

4、创建精确时钟频率

实验开发板上提供了 100MHz 的时钟信号，如果需要更为精准的分频时钟信号，需要通过引用 Vivado 中提供时钟 IP 核来实现。使用 IP 核生成像素时钟信号的方法如下：

1、在导航窗口中，点击 IP Catalog 菜单，在 Search 输入框中输入“clock”进行检索，在结果列表中找到 Clocking Wizard 行，如图 6.16 所示。Clocking Wizard 是 Vivado 提供的分频 IP 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。

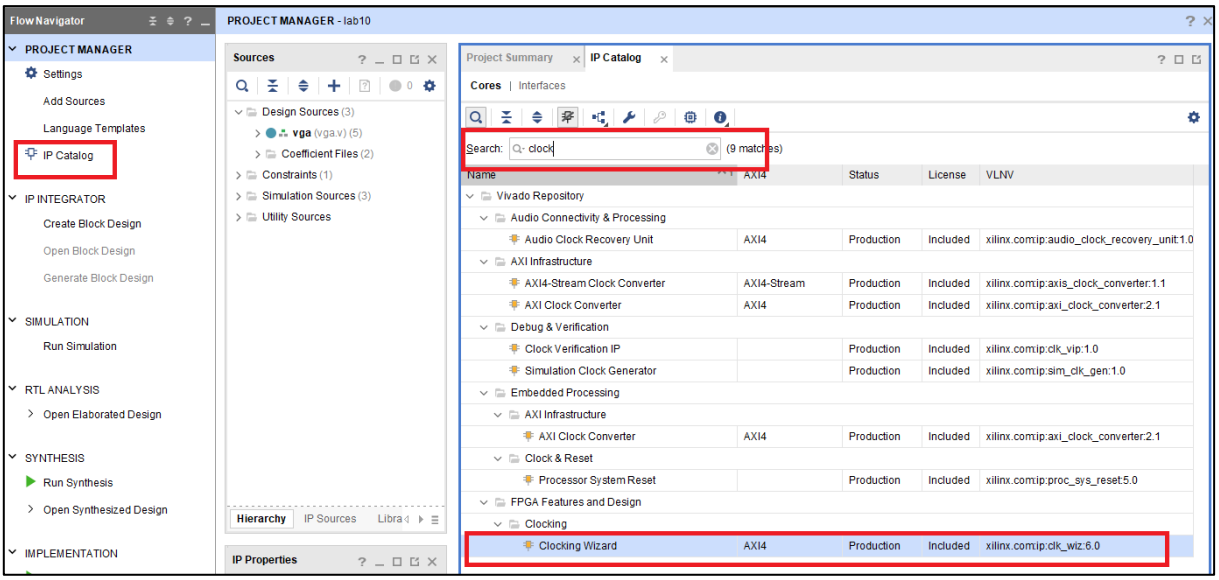

图 6.16　IP 目录中的 Clocking Wizard

双击 Clocking Wizard 行，弹出设置界面，模块名为 clk_wiz_0，Primitive 设置为锁相环 PLL。输入时钟 clk_in1 的设置为缺省的系统时钟 100MHz，其他保持缺省设置即可，如图 6.17 所示。

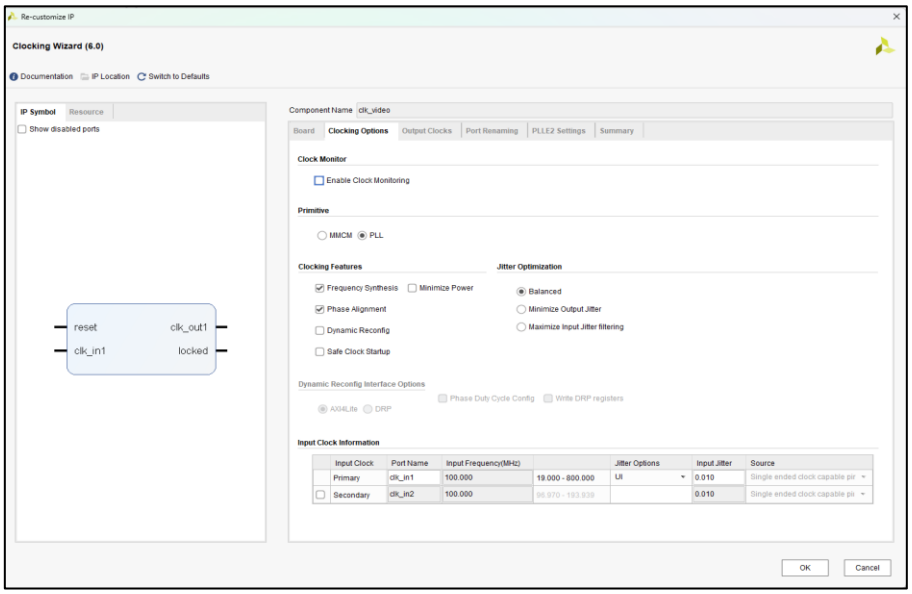

图 6.17　Clocking Wizard 的设置界面

点击 Output Clocks 标签，在页面中设置 clk_out1 为 74.28571MHz 或者 25.20325MHz，如图 6.18 所示。输出可以根据需要产生一个或多个时钟，比如，为显示器、键盘和七段数码管定义不同的时钟信号，每个时钟信号的频率可以自行设置。

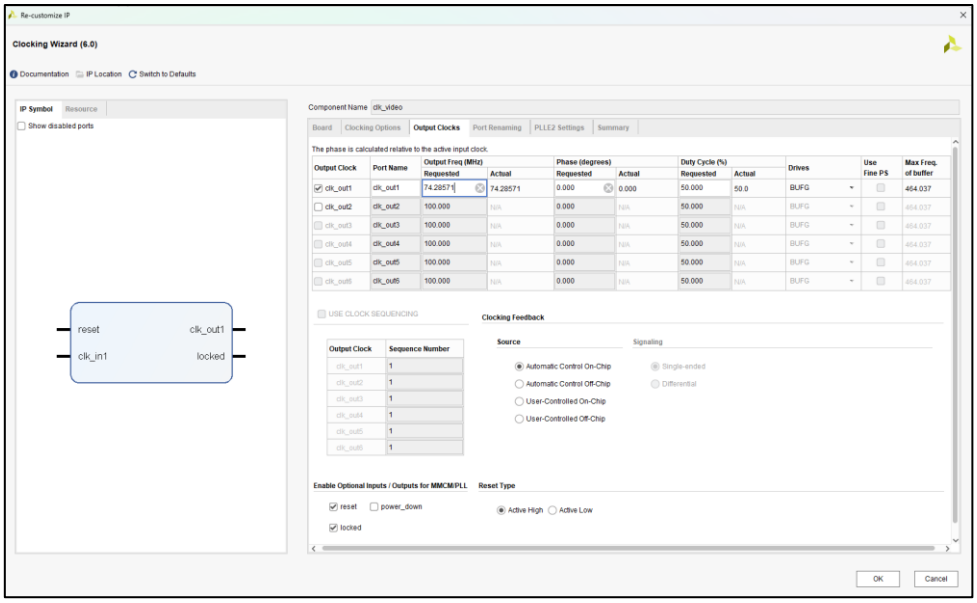

图 6.18　设置输出时钟信号频率

在完成配置后点击 OK 按钮，随后点击新对话框中的 Generate，如图 6.19 所示，就生成对应的时钟模块，在项目的源代码中新增 IP 核接口信息文件 clk_wiz_0.xci，该模块中还包括 Reset 和 Locked 等控制信号。IP 核产生的时钟有延时，可以利用 locked 信号作为标识，当时钟输入信号和输出信号保持稳定后，Locked 信号为高电平。

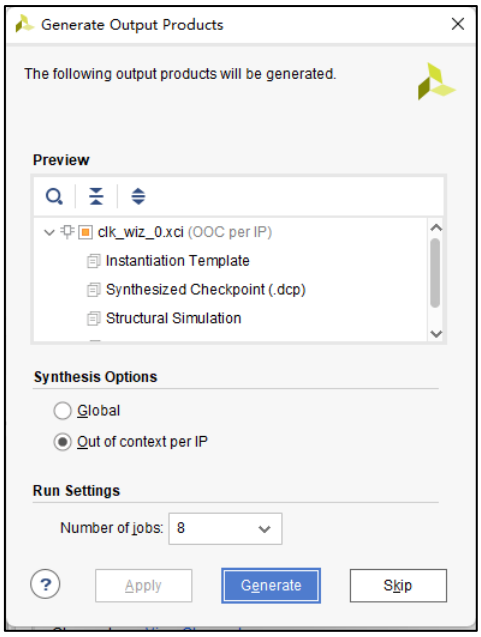

图 6.19　生成时钟信号 IP 核接口信息文件

在项目文件中通过实例来引用该 IP 核模块。如：

```
clk_wiz_0    myvgaclk(.clk_in1(CLK100MHZ),.reset(rst),.locked(locked),.clk_out1(vga_clk));
```

四、实验内容

实验要求：模仿 Window 命令行或 Linux 的字符终端，集成键盘接收模块，实现把键盘输入的字符在 VGA 显示器上回显的交互界面。支持所有大小写英文字母、数字以及可显示的字符。用显示闪烁的竖线或横线作为字符光标。支持回车、删除、退格等功能键。支持清屏和命令行滚动功能。支持系统提示符。键盘扫描码、显存和字符点阵字库等可以直接使用 FPGA 提供的分布式存储器来实现。

字符终端界面参考设计如下：

```
-------Xterminal----------------------------------------------[学号]-[姓名拼音]-[日期]-------
[G]raphics
[I]mage
[T]xt
[C]alculator
-
```

命令含义如下：

1、 当输入字符串为“[G]raphics”时，按回车后，显示一幅彩色图形，再按任意键返回菜单页。

2、 当输入字符串为“[I]mage”时，按回车后，显示一幅静态图像，再按任意键返回菜单页。

3、 当输入字符串为“[T]xt”时，按回车后，显示一段文本，支持上下左右光标键。

4、 当输入字符串为“[C]alculator”时，按回车后，输入一个四则运算表达式后（支持正负号和括号），显示运算结果。（选做）

假设 xterm 模块的接口定义如下：

```
module xterm(
    input CLK100MHZ,     //系统时钟信号
    input PS2_CLK,       //来自键盘的时钟信号
    input PS2_DATA,    //来自键盘的串行数据位
    input BTNC,          //Reset
    output [6:0]SEG,
    output [7:0]AN,        //显示扫描码和 ASCII 码
    output [15:0] LED,     //显示键盘状态
    output [3:0] VGA_R,
    output [3:0] VGA_G,
    output [3:0] VGA_B,
    output    VGA_HS,
    output    VGA_VS
);
// Add your code here
endmodule
```

请根据上述描述，按照下列步骤完成实验。

1、 使用 Vivado 创建一个新工程。

2、 点击添加设计源码文件，加入 lab6.zip 里的 xterm.v、VGASim.v、VGACtrl.v 和 VGADraw.v 等文件。

3、 点击添加约束文件，加入 lab6.zip 里的 xterm.xdc 文件。

4、 创建块存储 IP 核文件，加载显存内容。

5、 根据实验要求，完成源码文件的设计。

6、 对工程进行仿真测试。

7、 仿真通过后，进行综合、实现并生成比特流文件。

8、 生成比特流文件后，加载到实验开发板，进行调试验证，并记录验证过程。

五、思考题

1、 如何在显示器分辨率设置为1280×1024时，在屏幕中间显示640×480的图像？

2、 试试实现在屏幕上彩色字符（如图6.15所示）或实现类似电影Matrix开头的字符雨效果。

3、 说说如何字屏幕上显示汉字（实验资源中HZK16S汉字16×16点阵字库文件）？

4、 简述如何在图形界面中显示鼠标位置及其移动方向。

图 6.15 彩色字符参考设计