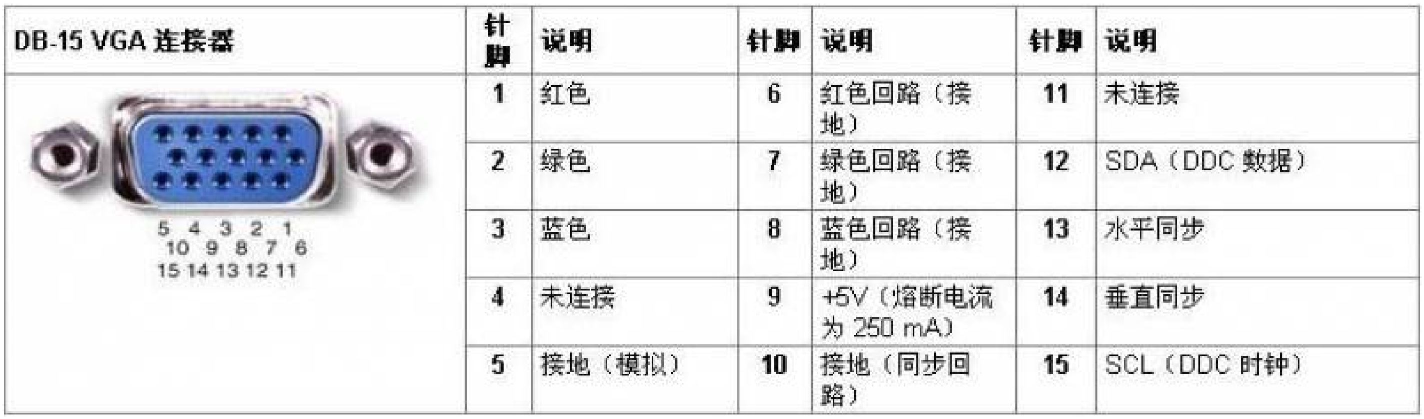
8-color VGA function driver based on STEP FPGA

In this section, we will use the 8-color VGA interface on the FPGA driver backplane to realize the 8-color bar display function.

====Hardware description====

VGA (video graphics array) is a video graphics array, which is a video transmission standard that uses analog signals that IBM introduced with PS/2 in 1987. The VGA interface is divided into male and female ports, as shown in the figure below: VGA interface pins are defined as follows: A standard VGA interface should have the following ports:

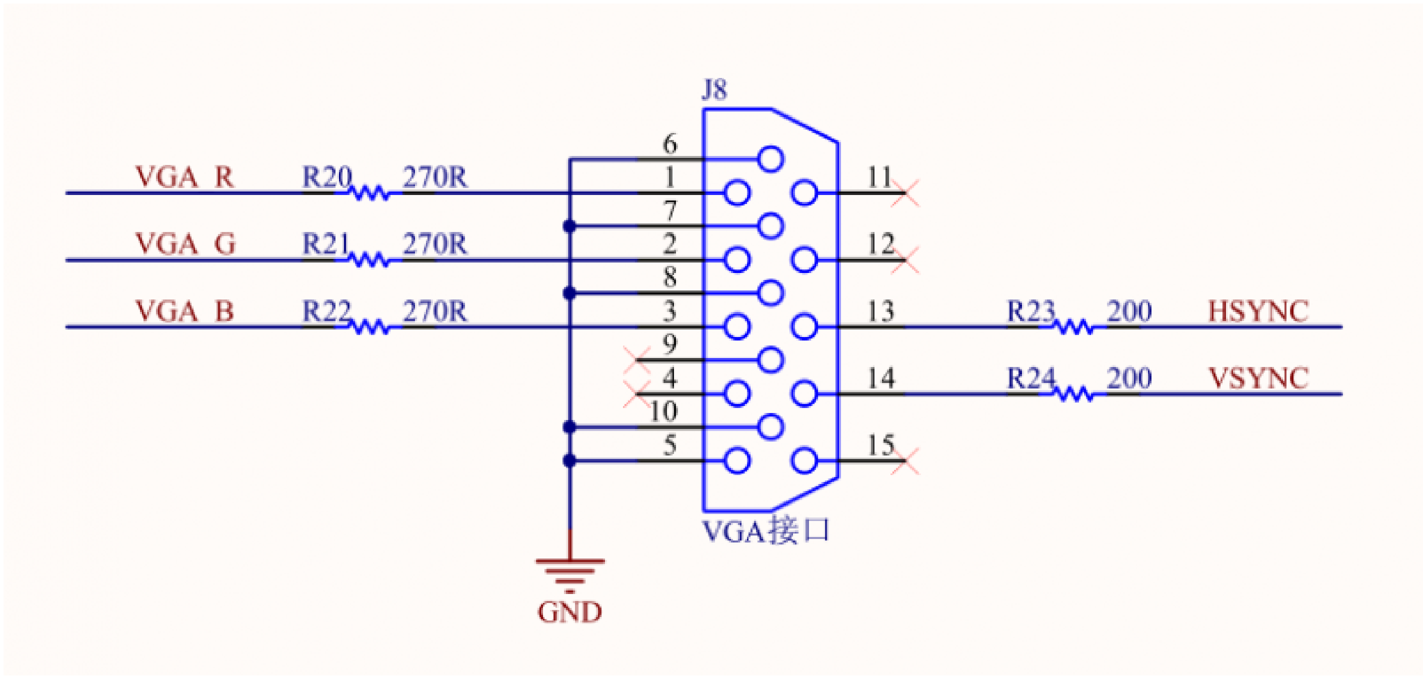


DB-15 VGA 连接器	针脚	说明	针脚	说明	针脚	说明
5 4 3 2 1 10 9 8 7 6 15 14 13 12 11	1	红色	6	红色回路（接地）	11	未连接
	2	绿色	7	绿色回路（接地）	12	SDA（DDC 数据）
	3	蓝色	8	蓝色回路（接地）	13	水平同步
	4	未连接	9	+5V（熔断电流为 250 mA）	14	垂直同步
	5	接地（模拟）	10	接地（同步回路）	15	SCL（DDC 时钟）

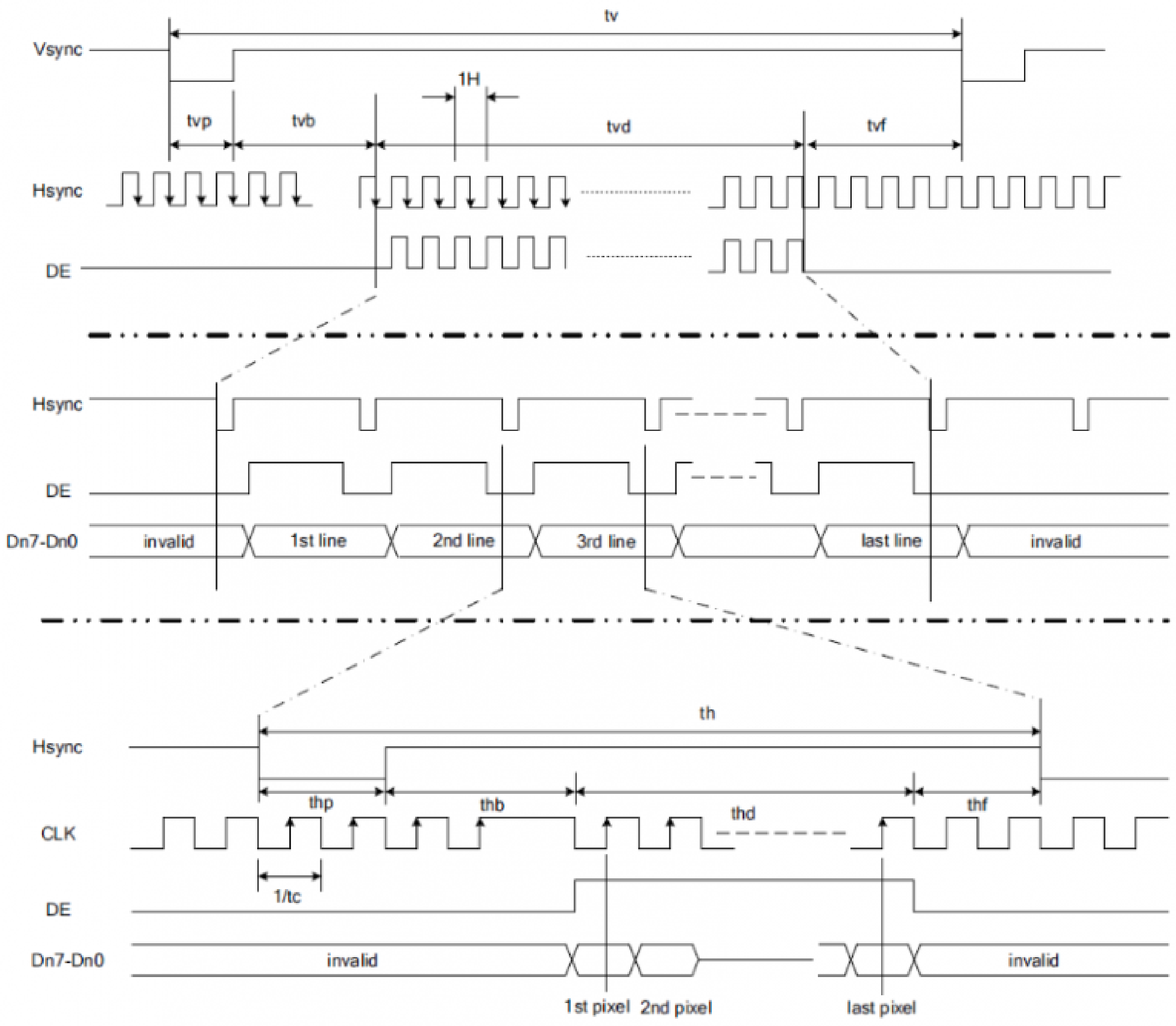
- Red, green and blue signal (R\G\B)
- Line and field synchronization signal (HS\VS)
- And a lot of ground shielding;

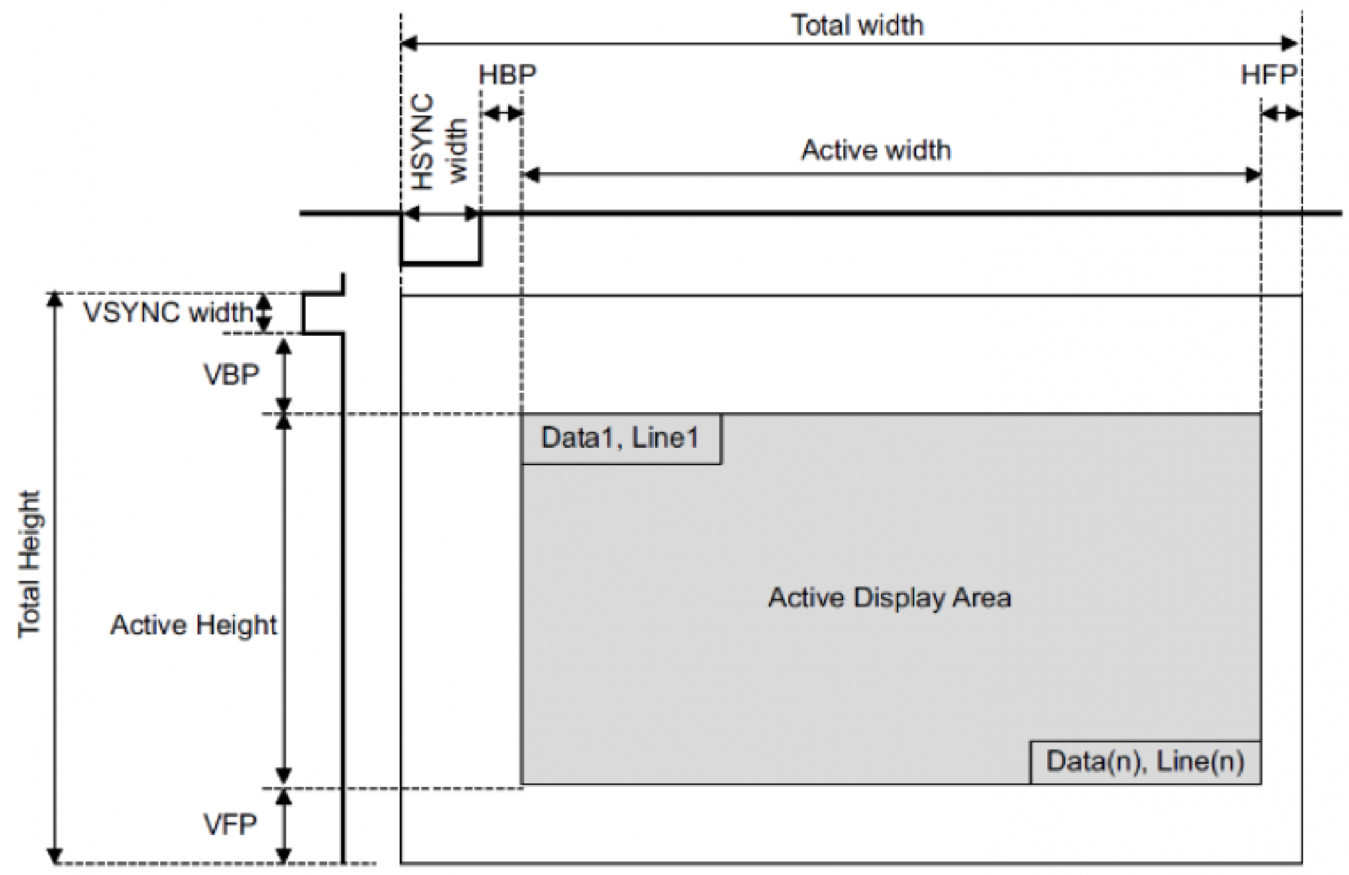
The three-color signals are all analog signals,
and the horizontal and vertical synchronization signals are all digital signals; for the VGA interface analog voltage, it is 0~0.714V, 0 means colorless, 0.714 means full color, FPGA outputs 3.3V, so it must be passed DAC conversion. There are two more mature methods today: the resistor divider method and the DAC conversion method.
Our bottom board uses a resistor divider method. Because there is a 75 ohm pull-down resistor on the VGA display end, in order to obtain a voltage of 0.714V, we put a 270 ohm resistor into the RGB signal line in series, 3.3V*75/(270 +75)=0.717V. The following VGA drives the display using a scanning method. The HS (Horizontal Synchronization)

progressive scan starts from the upper left corner of the screen and scans from left to right point by point. After scanning one line, the electron beam returns The starting position of the next line on the left side of the screen. During this period, the CRT (Cathode Ray Tube) blanks the electron beam. At the end of each line, the line synchronization signal is used for synchronization; when all lines are scanned, a frame is formed, and the field synchronization is used The signal is synchronized, and the scan returns to the upper left of the screen, while vertical blanking is performed to start the next frame. VGA is scanning all the time, and the scanning of each field includes several lines of scanning, which are cycled in turn; VGA display sequence is as follows: VGA display area and blanking area: common VGA display mode:



INPUT SIGNAL TIMING DIAGRAM





模式	像素频率（MHz）	横向				纵向			
		有效线数	前廊	同步脉冲	后廊	有效线数	前廊	同步脉冲	后廊
640x480, 60Hz	25.175	640	16	96	48	480	11	2	31
640x480, 72Hz	31.500	640	24	40	128	480	9	3	28
640x480, 75Hz	31.500	640	16	96	48	480	11	2	32
640x480, 85Hz	36.000	640	32	48	112	480	1	3	25
800x600, 56Hz	38.100	800	32	128	128	600	1	4	14
800x600, 60Hz	40.000	800	40	128	88	600	1	4	23
800x600, 72Hz	50.000	800	56	120	64	600	37	6	23
800x600, 75Hz	49.500	800	16	80	160	600	1	2	21
800x600, 85Hz	56.250	800	32	64	152	600	1	3	27
1024x768, 60Hz	65.000	1024	24	136	160	768	3	6	29
1024x768, 70Hz	75.000	1024	24	136	144	768	3	6	29
1024x768, 75Hz	78.750	1024	16	96	176	768	1	3	28
1024x768, 85Hz	94.500	1024	48	96	208	768	1	3	36

====Verilog code====

```
// --------------------------------------------------- --------------------
// >>>>>>>>>>>>>>>>>>>>>>>>> COPYRIGHT NOTICE < <<<<<<<<<<<<<<<<<<<<<<<
// ------------------------ ----------------------------------------------
// Module: Param_define
/ /
// Author: Step
//
// Description: Param_define
//
// Web: www.stepfpga.com
//
// ---------------------- ------------------------------------------------
// Code Revision History:
// ------------------------------------------------- ---------------------
// Version: |Mod. Date: |Changes Made:
// V1.1 |2016/10/30 |Initial ver
// ----------------------------------------- ------------------------------
`timescale  1ns  /  1ns

//VGA display driver only needs 5 signals (line sync, field sync, red, green, blue signal)
//
Red, green and blue signals are analog signals, and the input voltage range is 0.0V~0.7V // In
VGA timing, the horizontal synchronization and vertical synchronization are divided into four
 stages (synchronization pulse, back porch, number of effective lines, front porch)
//VGA display has many modes, and each mode has fixed clock and timing parameters, according t
o Request control

`ifdef VGA_800X600_60Hz  //Corresponding parameters for different VGA display modes
//----------------------------------- ---------------------------------------
//-- Horizonal timing information
`define HSYNC_A    1 6'd128     // 128
`define HSYNC_B    1 6'd216     // 128 + 88
`define HSYNC_C    1 6'd1016    // 128 + 88 + 800
`define HSYNC_D    1 6'd1056    // 128 + 88 + 800 + 40 // Line sync pulse + back porch + numbe
r of effective lines + front porch
//-- Vertical timing information
`define VSYNC_O    1 6'd4               // 4
`define VSYNC_P    16'd27       // 4 + 23
`define VSYNC_Q    1 6'd627     // 4 + 23 + 600
`define VSYNC_R    1 6'd628     // 4 + 23 + 600 + 1 //field sync pulse + back gallery + number
of valid lines +front porch
//----------------------------------------------- -------------------------------
`endif

`ifdef VGA_640X480_85Hz  //corresponding parameters for different VGA display modes
//----------------------------------- ---------------------------------------
//-- Horizonal timing information
`define HSYNC_A    1 6'd48      // 48
`define HSYNC_B    1 6'd160     // 48 + 112
`define HSYNC_C    1 6'd800     // 48 + 112 + 640
`define HSYNC_D    1 6'd832     // 48 + 112 + 640 + 32 // Line sync pulse + back porch + numbe
r of effective lines + front porch
```

```
//-- Vertical timing information
`define VSYNC_O    1 6'd3               // 3
`define VSYNC_P    16'd28       // 3 + 25
`define VSYNC_Q    1 6'd508     // 3 + 25 + 480
`define VSYNC_R    1 6'd509     // 3 + 25 + 480 + 1 //field sync pulse + back gallery + number
of effective lines +front porch
//----------------------------------------------- ------------------------------
`endif
```

```
// -------------------------------------------------- --------------------
// >>>>>>>>>>>>>>>>>>>>>>>>> COPYRIGHT NOTICE < <<<<<<<<<<<<<<<<<<<<<<<
// ------------------------ --------------------------------------------
// Module: Vga_Module
/ /
// Author: Step
//
// Description: Vga_Module
//
// Web: www.stepfpga.com
//
// ---------------------- ----------------------------------------------
// Code Revision History:
// ------------------------------------------------ ---------------------
// Version: |Mod. Date: |Changes Made:
// V1.1 |2016/10/30 |Initial ver
// -------------------------------------- -------------------------------
`define                 VGA_800X600_60Hz         //Define the VGA display mode used

`ifdef          VGA_800X600_60Hz         //Call the corresponding parameters according to the
 definition of VGA display mode
` include       "Param_define.v"        //Call the global definition in the Param_define.v fil
e
`endif

module Vga_Module
 (
input                               clk_in ,                              //40MHz system
clock
input                               rst_n_in ,                    //System reset, low ef
fective
output          reg                         sync_v ,                               //VGA
 field sync sync_v
output          reg                         sync_h ,                               //VGA
 line sync sync_h
output   reg            [ 2 : 0 ]       vga_data                        / /VGA data MSB~LSB =
{R,G,B}
) ;

reg                     [ 15 : 0 ]      x_cnt ;
reg                     [ 15 : 0 ]      y_cnt ;
reg                                     vga_valid ;

//Count the clock to identify the time required for one line scan of VGA
always  @  ( posedge clk_in or  negedge rst_n_in )
        if ( ! Rst_n_in ) x_cnt <=  1 6'd0 ;    //Initial value at reset
        else  if ( x_cnt >= `HSYNC_D ) x_cnt <=  1 6'd0 ;       //One line scan requires 1056
 clocks (128+88+800+40)
        else x_cnt <= x_cnt +  1'b1 ;

//Count the line scan to identify the time required for one field scan of VGA
always  @  ( posedge clk_in or  negedge rst_n_in )
```

```
        if ( ! Rst_n_in ) y_cnt <=  1 6'd0 ;    //Initial value at reset
        else  if ( x_cnt == `HSYNC_D )  begin   //Each line scan
                if ( y_cnt >= `VSYNC_R ) y_cnt <=  1 6'd0 ;     //Each field scan contains 628
line scans
                else y_cnt <= y_cnt +  1'b1 ;
        end  else y_cnt <= y_cnt ;      //The field scan counter remains unchanged during each
line scan

//According to the parameters of the display mode to generate the pulse of line synchronizatio
n scanning
always  @  ( posedge clk_in or  negedge rst_n_in )
        if ( ! Rst_n_in ) sync_h <=  1'b1 ;
        else  if ( x_cnt < `HSYNC_A ) sync_h <=  1'b0 ;
        else sync_h <=  1'b1 ;

//Generate
 vertical sync scan pulses according to the parameters of the display mode always  @  ( posedg
e clk_in or  negedge rst_n_in )
        if ( ! Rst_n_in ) sync_v <=  1'b1 ;
        else  if ( y_cnt < `VSYNC_O ) sync_v <=  1'b0 ;
        else sync_v <=  1'b1 ;

//Determine the effective display area according to the number of effective lines of the
 horizontal and vertical synchronization signal always  @  ( posedge clk_in or  negedge rst_n_
in )
        if ( ! Rst_n_in )
                vga_valid <=  1'b0 ;
        else  if ( ( x_cnt > `HSYNC_B )  &&  ( x_cnt < ` HSYNC_C )  &&  ( y_cnt > `VSYNC_P )
&&  ( y_cnt < `VSYNC_Q ) )
                vga_valid <=  1'b1 ; //The vga_valid flag in the effective display area is 1
        else
                vga_valid <=  1'b0 ;

//Display different colors in different segments of the VGA effective display area
always  @  ( posedge clk_in or  negedge rst_n_in )
begin
        if ( ! Rst_n_in ) vga_data =  3'b111 ;
        else  if ( vga_valid ) begin    //In the effective display area
                if ( ( x_cnt > `HSYNC_B )  &&  ( x_cnt <= `HSYNC_B +  1 0'd100 ) )
                        vga_data =  3'b100 ;                    //Red
                else  if ( ( x_cnt > `HSYNC_B +  1 0'd100 )  &&  ( x_cnt <= `HSYNC_B +  1 0'd2
00 ) )
                        vga_data =  3'b010 ;                    //green
                else  if ( ( x_cnt > `HSYNC_B +  1 0'd200 )  &&  ( x_cnt <= `HSYNC_B +  1 0'd3
00 ) )
                        vga_data =  3'b001 ;                    //blue
                else  if ( ( x_cnt > `HSYNC_B +  1 0'd300 )  &&  ( x_cnt <= `HSYNC_B +  1 0'd4
00 ) )
                        vga_data =  3'b110 ;                    //yellow
                else  if ( ( x_cnt > `HSYNC_B +  1 0'd400 )  &&  ( x_cnt <= `HSYNC_B +  1 0'd5
00 ) )
                        vga_data =  3'b101 ;                    //purple
                else  if ( ( x_cnt > `HSYNC_B +  1 0'd500 )  &&  ( x_cnt <= `HSYNC_B +  1 0'd6
00 ) )
                        vga_data =  3'b011 ;                    //Cyan
                else  if ( ( x_cnt > `HSYNC_B +  1 0'd600 )  &&  ( x_cnt <= `HSYNC_B +  1 0'd7
00 ) )
                        vga_data =  3'b111 ;                    //white
                else  if ( ( x_cnt > `HSYNC_B +  1 0'd700 )  &&  ( x_cnt <= `HSYNC_B +  1 0'd8
00 ) )
                        vga_data =  3'b000 ;                    //black
                else
                        vga_data =  3'b111 ;                    //white
        end  else
                vga_data =  3'b111 ;                            //white
end

endmodule
```

====Summary====

This section mainly explains the principle, timing and software design of VGA display for everyone. You need to create your own project while mastering it, and generate FPGA configuration file loading test through the entire design process. If you are not familiar with the use of Diamond software, please refer to here: Use of Diamond .

====Related Information====

Use STEP-MXO2 second generation VGA display driver: download link will be updated, the subsequent
use of STEP-MAX10 VGA display driver: subsequent download link will be updated