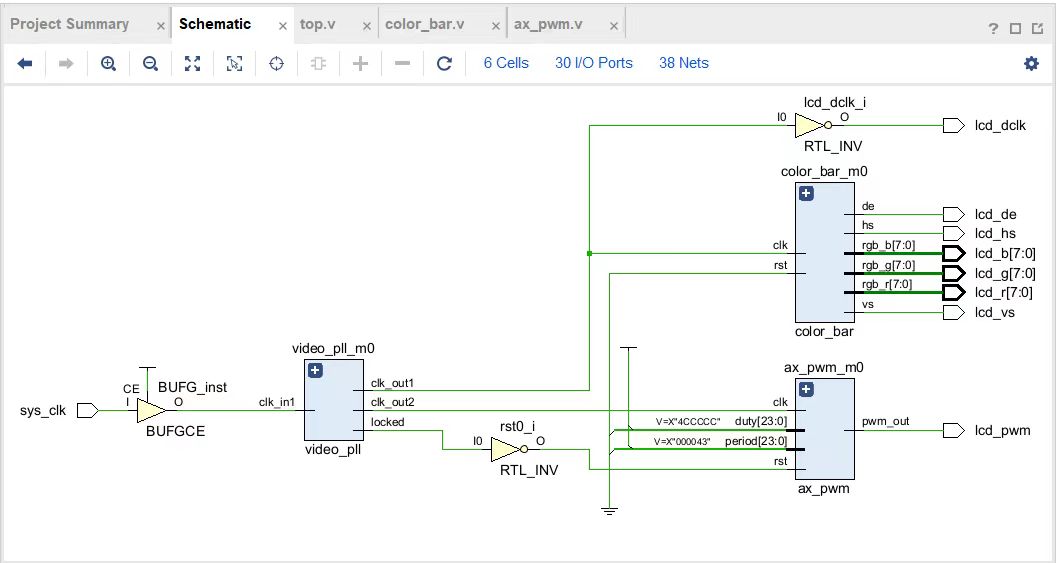
**彩条显示实验**

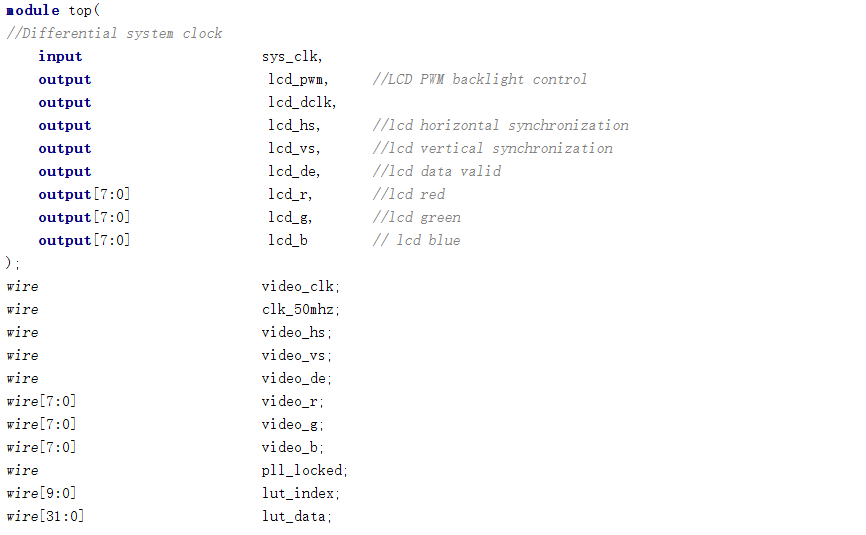
用alinx的开发板驱动LCD液晶屏显示彩条

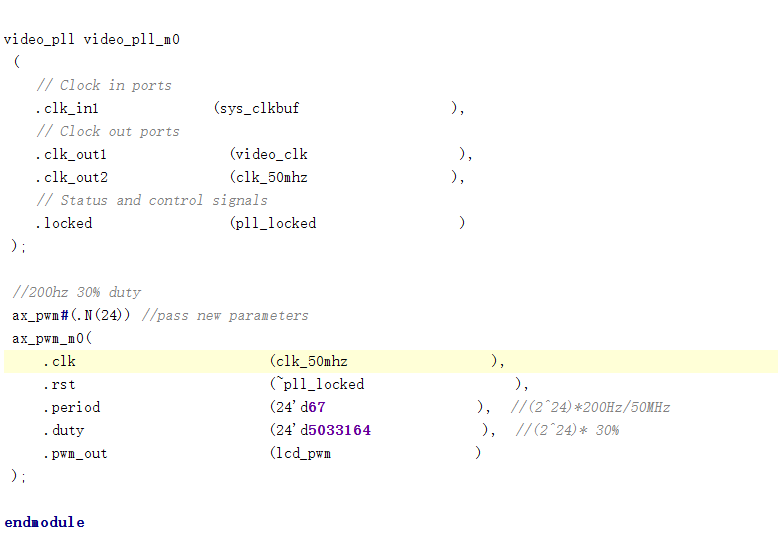
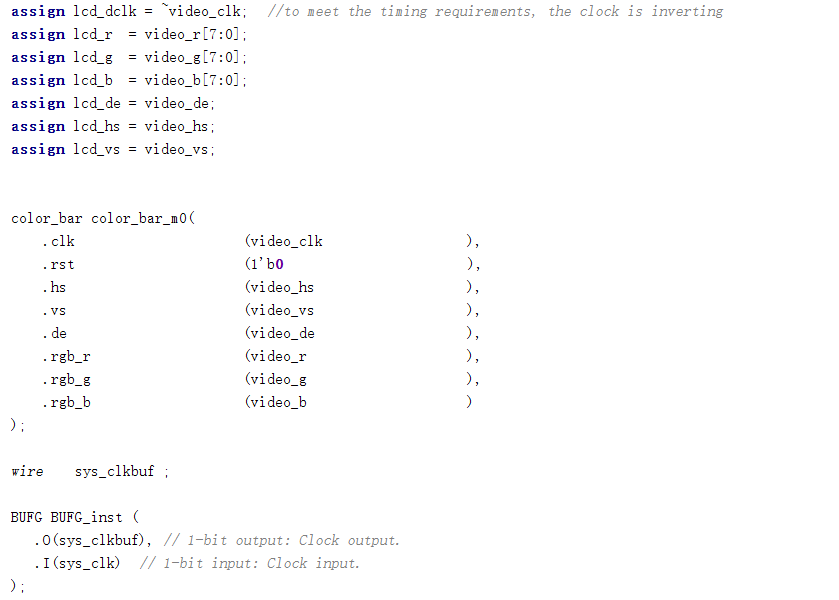
RTL结构框图：



主要包括四个模块：时钟分频模块、顶层模块、LCD驱动模块、LCD显示模块

顶层模块实现：





如果把我们写的代码抽象成黑箱，那么顶层文件就是对黑箱输入某些信号，然后得到某些信号。在对黑箱输入和输出配置完之后，我们要进到黑箱当中，对每个module进行连线，相当于是黑箱的黑箱，所以，我们需要定义wire变量，用于module之间的数据传递，在顶层文件定义wire变量，就是用来对每个module之间的input和output进行连线。

Input：

1.系统时钟

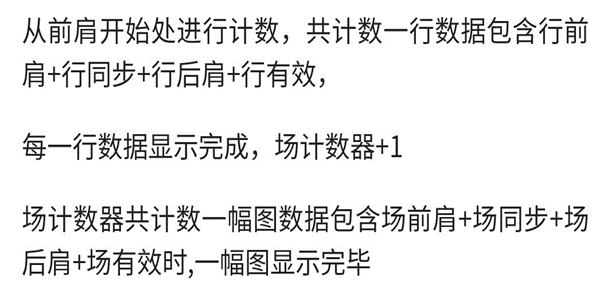
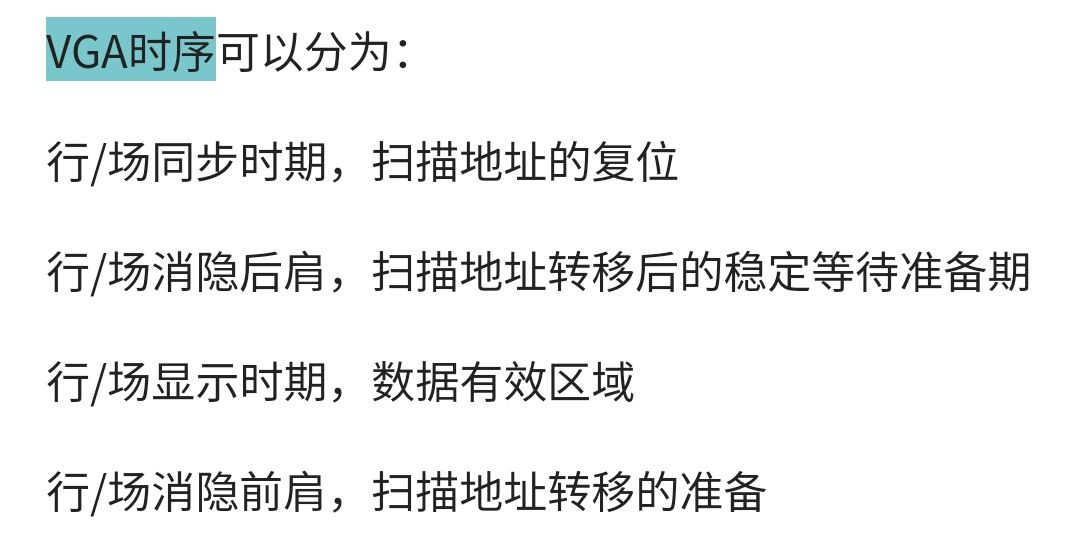
Output：

1.lcd\_pwm：调节占空比来调节lcd屏幕供电电压 调节显示亮度

2.lcd\_dclk:lcd\_dclk为video\_clk按位取反

3.lcd\_r、lcd\_g、 lcd\_b：lcd屏幕的rgb数据信号

4.lcd\_hs、lcd\_vs行场同步信号



时钟分频模块

module video\_pll\_exdes

(

// Reset that only drives logic in example design

input COUNTER\_RESET,

output [2:1] CLK\_OUT,

// High bits of counters driven by clocks

output [2:1] COUNT,

// Status and control signals

output locked,

// Clock in ports

input clk\_in1

);

// Parameters for the counters

//-------------------------------

localparam ONE\_NS = 1000;

localparam time PER1 = 10\*ONE\_NS;

localparam time PER1\_1 = PER1/2;

// Counter width

localparam C\_W = 16;

// Clock to Q delay of 100ps

localparam TCQ = 100;

localparam NUM\_C = 2;

genvar count\_gen;

// When the clock goes out of lock, reset the counters

wire reset\_int = (!locked) || COUNTER\_RESET;

(\* ASYNC\_REG = "TRUE" \*) reg [NUM\_C:1] rst\_sync;

(\* ASYNC\_REG = "TRUE" \*) reg [NUM\_C:1] rst\_sync\_int;

(\* ASYNC\_REG = "TRUE" \*) reg [NUM\_C:1] rst\_sync\_int1;

(\* ASYNC\_REG = "TRUE" \*) reg [NUM\_C:1] rst\_sync\_int2;

// Declare the clocks and counters

wire [NUM\_C:1] clk\_int;

wire [NUM\_C:1] clk;

reg [C\_W-1:0] counter [NUM\_C:1];

wire clk\_in1\_buf;

wire clk\_in2\_buf;

wire clkfb\_in\_buf;

// Insert BUFGs on all input clocks that don't already have them

//--------------------------------------------------------------

BUFG clkin1\_buf

(.O (clk\_in1\_buf),

.I (clk\_in1));

// Instantiation of the clocking network

//--------------------------------------

video\_pll clknetwork

(

// Clock out ports

.clk\_out1 (clk\_int[1]),

.clk\_out2 (clk\_int[2]),

// Status and control signals

.locked (locked),

// Clock in ports

.clk\_in1 (clk\_in1\_buf)

);

genvar clk\_out\_pins;

generate

for (clk\_out\_pins = 1; clk\_out\_pins <= NUM\_C; clk\_out\_pins = clk\_out\_pins + 1)

begin: gen\_outclk\_oddr

ODDRE1

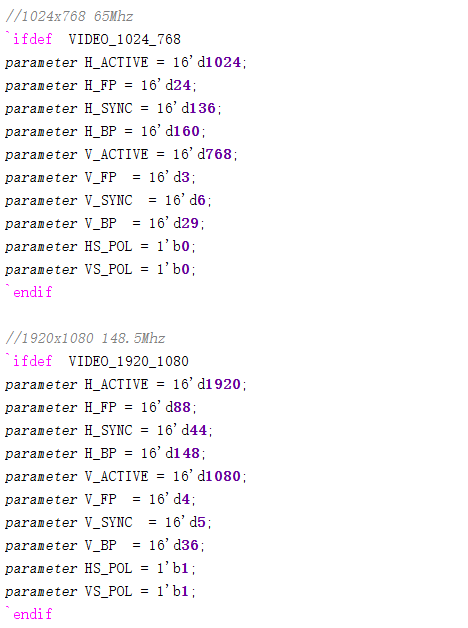
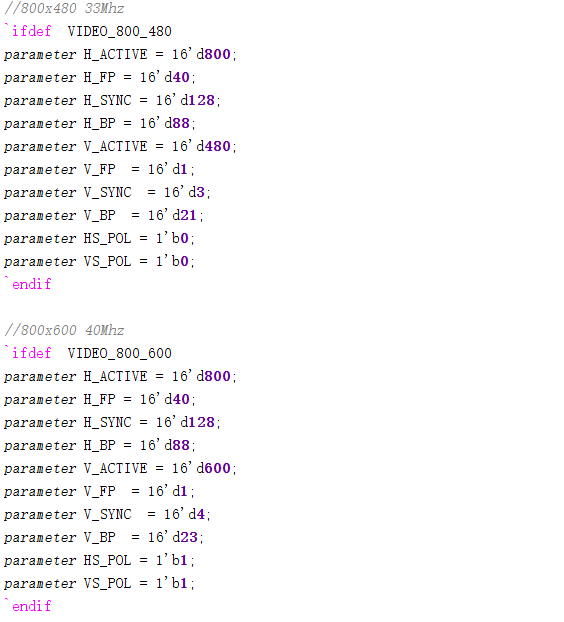
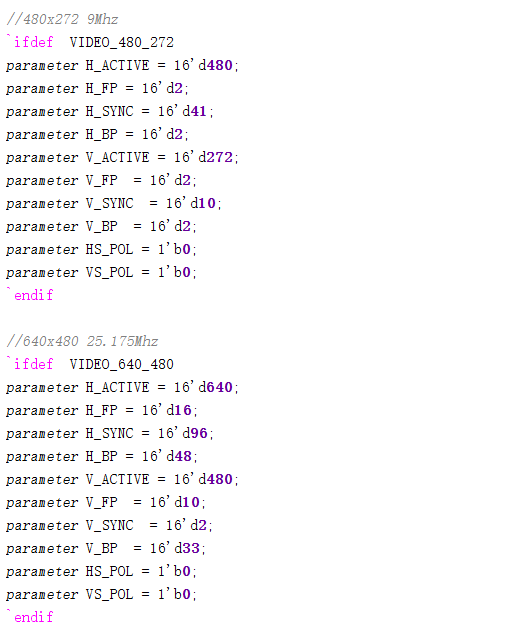
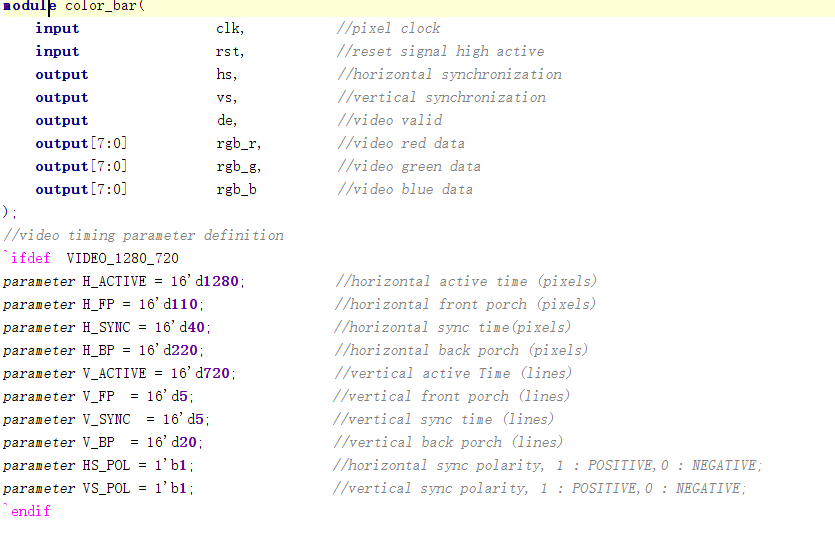
clkout\_oddr

(.Q (CLK\_OUT[clk\_out\_pins]),

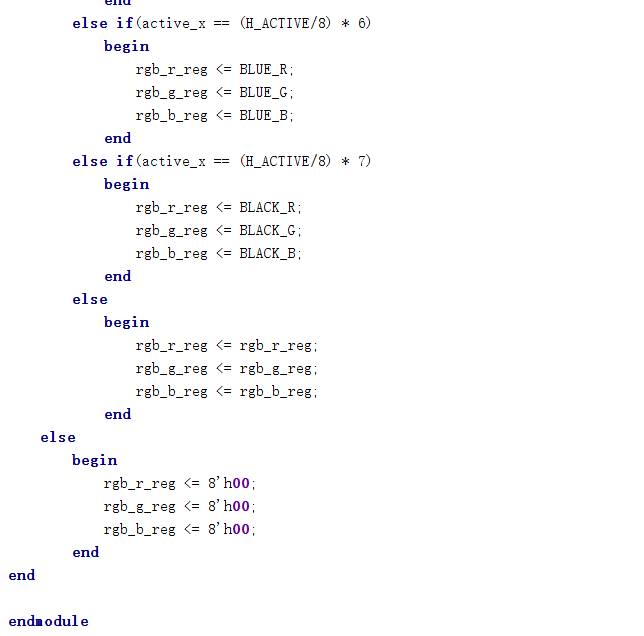
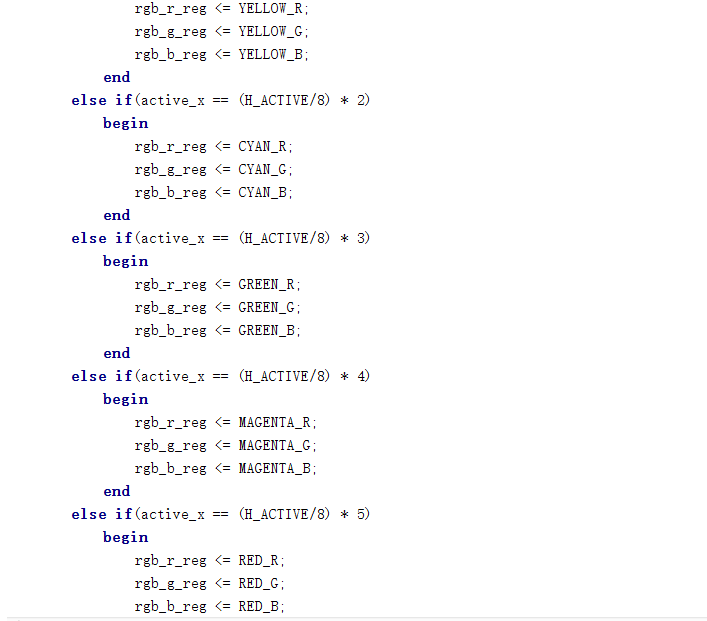
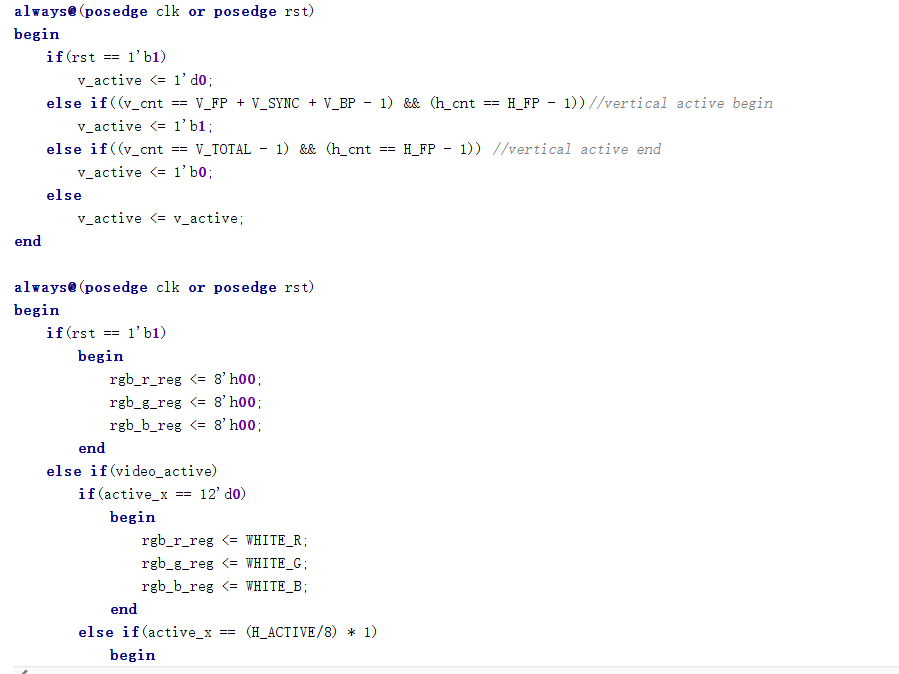
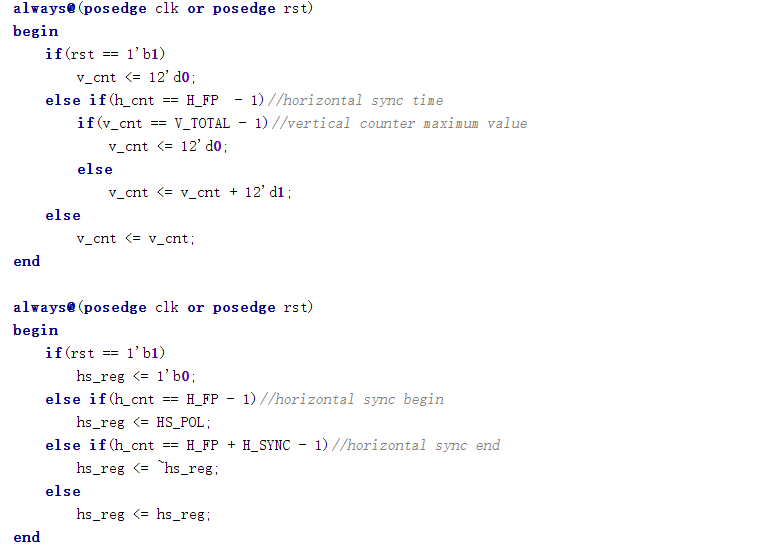
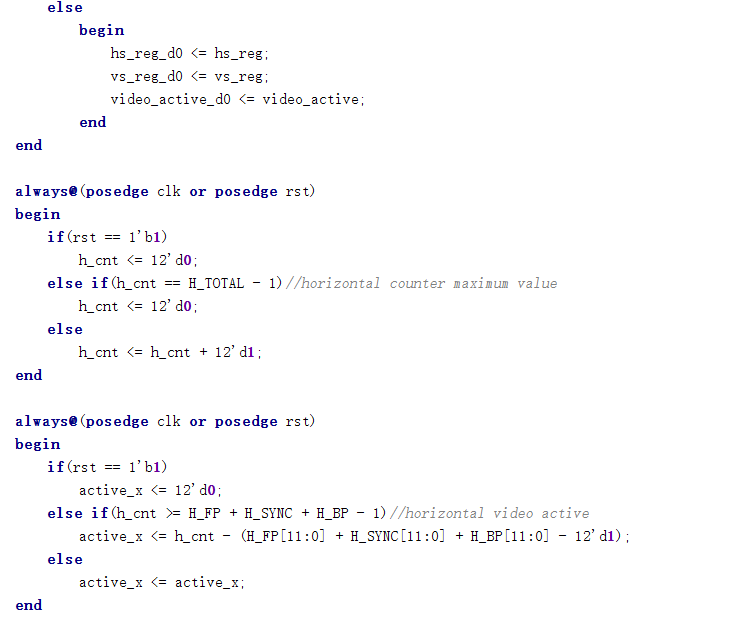
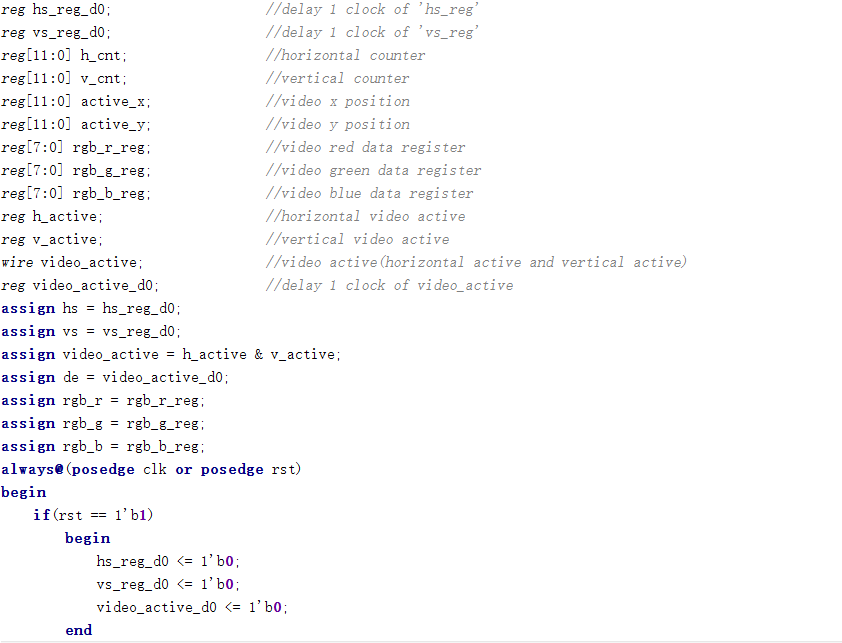
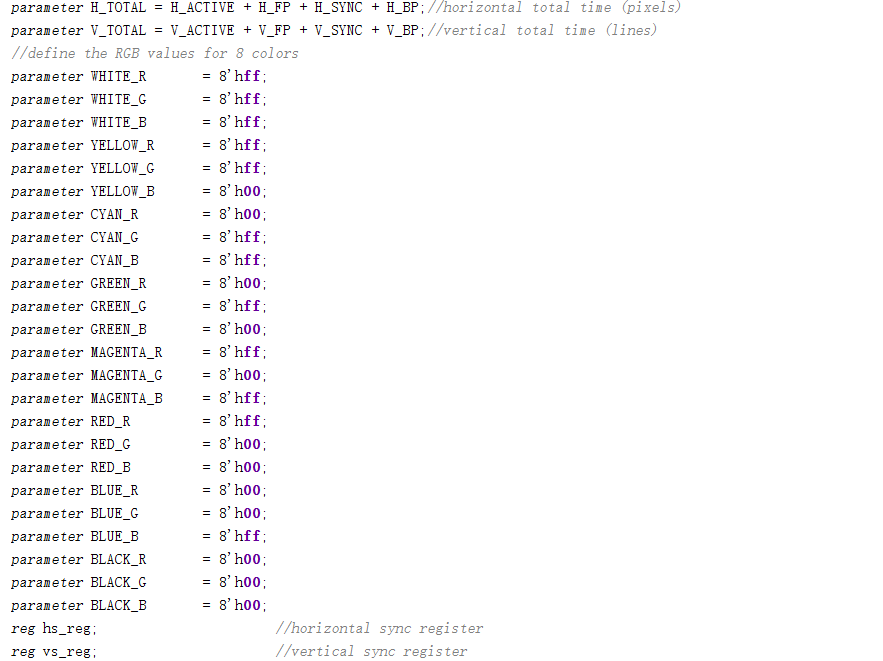
.C (clk\_int[clk\_out\_pins])

分频时钟信号clk\_out1进行时钟翻转，提高采集信号的完整度(这块是PLL的IP核，内部代码不需要掌握，可直接引用并修改输出频率)

LCD驱动模块

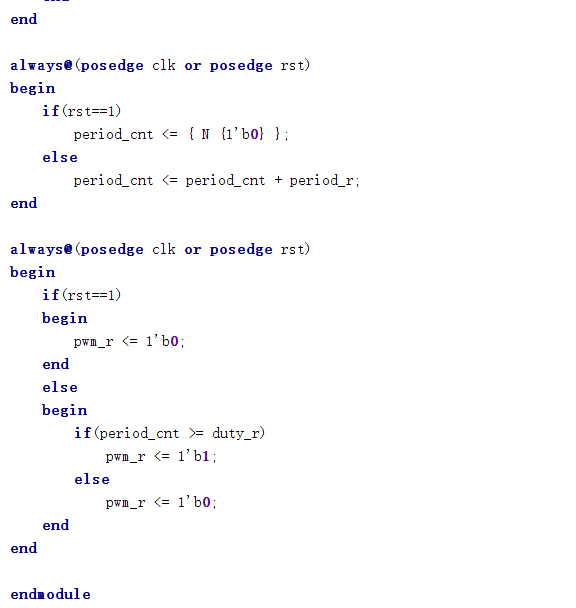
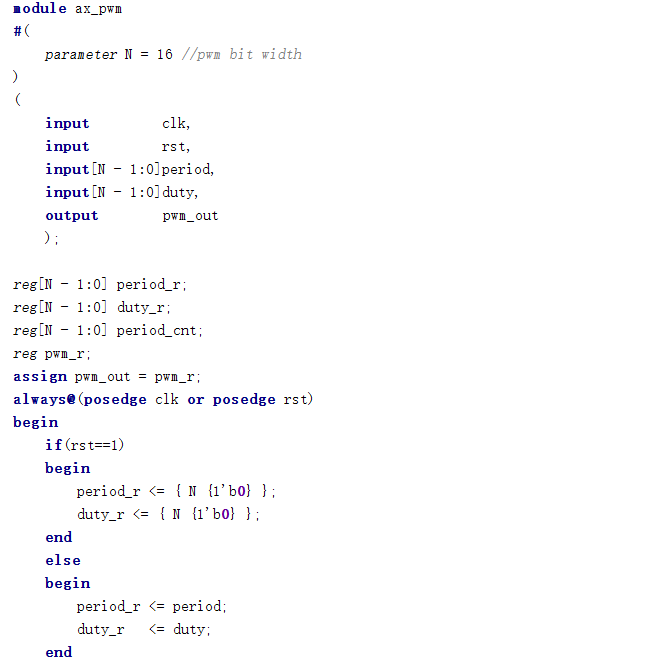


提供了多种不同的屏幕显示情况时的代码。因为这个LCD屏幕分辨率是480\*272的，因此我们选择这个。以后如果用到1920\*1080时也不用修改代码，直接在 video\_define.v处修改即可。



围绕行场同步展开。

LCD显示模块



通过pwm模块改变占空比，在单位时间输出功率不同，从而改变LCD显示亮度