一、实验内容

1.项目内容概括

基于 FPGA 和 MATLAB 处理的的《父与子》漫画阅读器

2.操作说明

连接开发板，上推 J15 开关，阅读器、数码管启动，VGA 显示屏显示漫画封面，数码管显示当前页数 1，正常频次力度按压按键 P17 可上翻漫画，按压 M17 可下翻漫画，与此同时数码管输出当前页数(PAgE+四位数字)，下推开关 J15 关闭漫画显示屏和数码管，再次上推二者对应显示上次关闭前的状态。

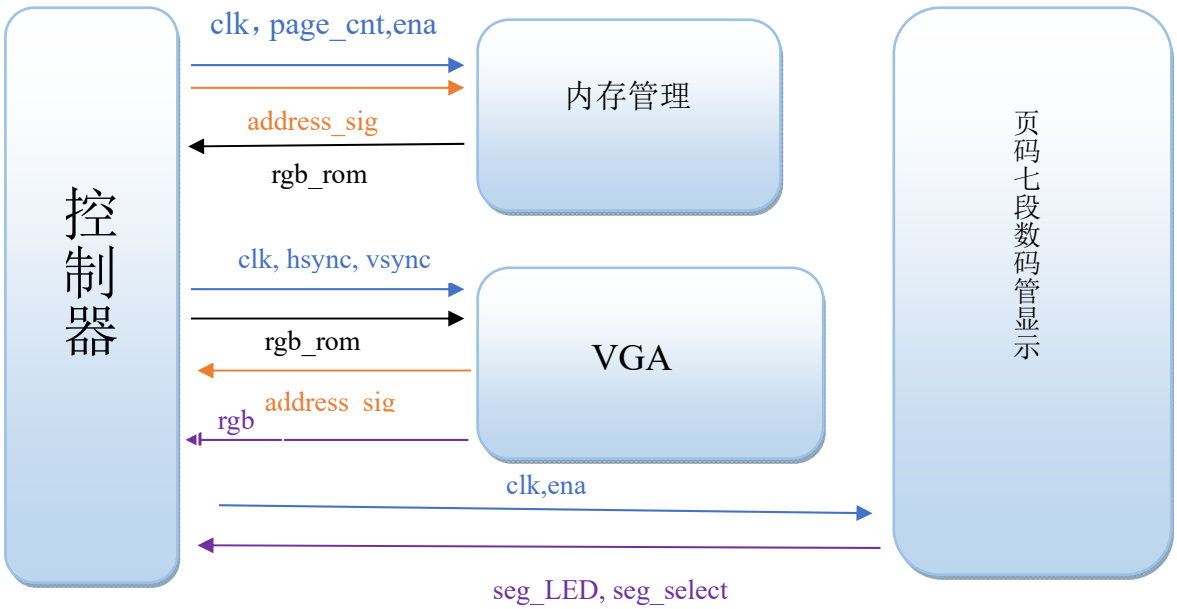
3.器件简介

VGA——VGA(Video Graphics Array)是 IBM 在 1987 年随 PS/2 机一起推出的一种视频传输标准，具有分辨率高、显示速率快、颜色丰富等优点；

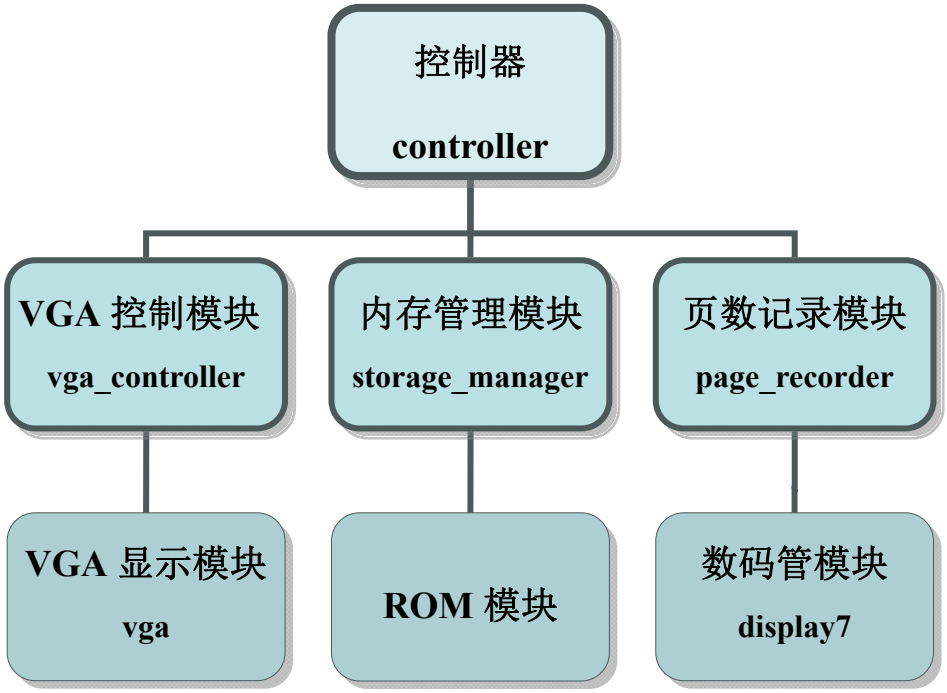
Nexys 4 DDR Artix-7——由 Xilinx 公司开发出的一款现场可编程门阵列（FPGA）开发板。

二、漫画阅读器数字系统总框图

1. 逻辑框图



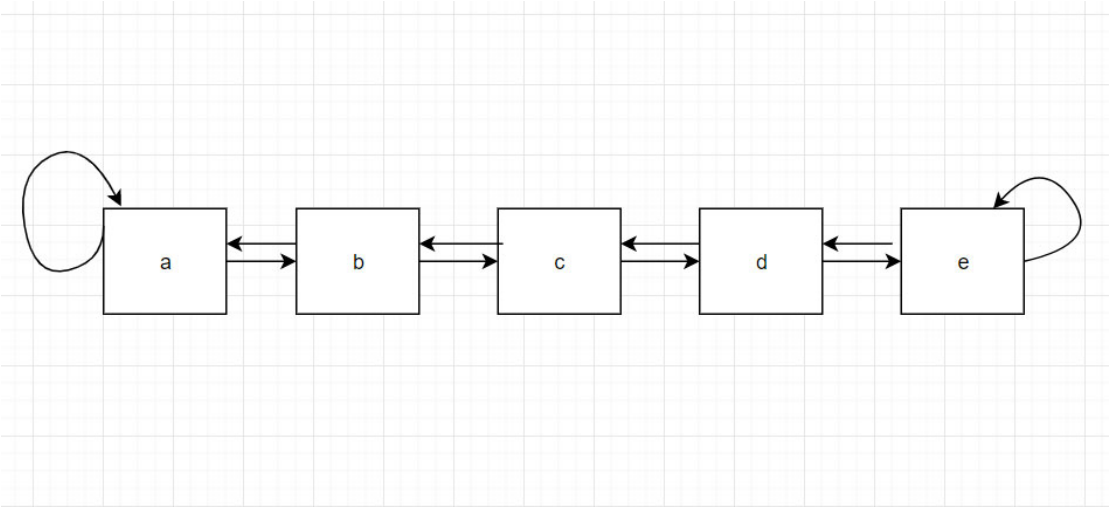
2. 模块构架



三、系统控制器设计

状态转移图：

ena=0 的情况下，状态保持不变；ena==1 的情况下，如图：



状态转移真值表：

A(n	B(n	C(n	ena=1	ena=0||

)	)	)	pgup=1			pgup=0&&pgdown=1			pgup=0&&pgdown=0		
			A(n+1)	B(n+1)	C(n+1)	A(n+1)	B(n+1)	C(n+1)	A(n+1)	B(n+1)	C(n+1)
0	0	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	1	0	0	0	1
0	1	0	0	0	1	0	1	1	0	1	0
0	1	1	0	1	0	1	0	0	0	1	1
1	0	0	0	1	1	1	0	0	1	0	0

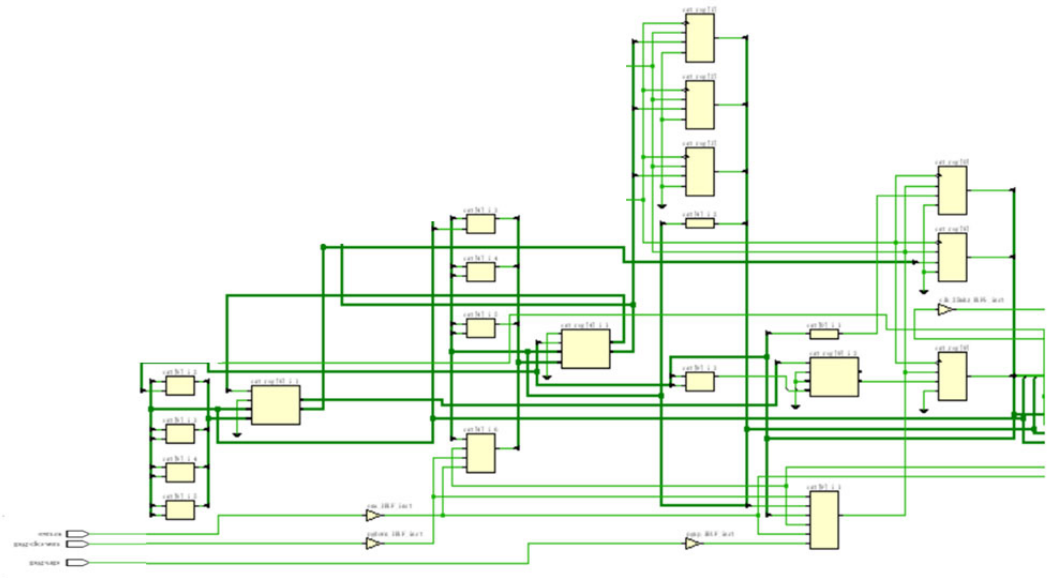
次态激励函数表达式:

A(n+1)=(1-pgup*ena)A(n)not(B(n))notC(n)+enaNot(A(n))B(n)C(n)Not(pgup) pgdown

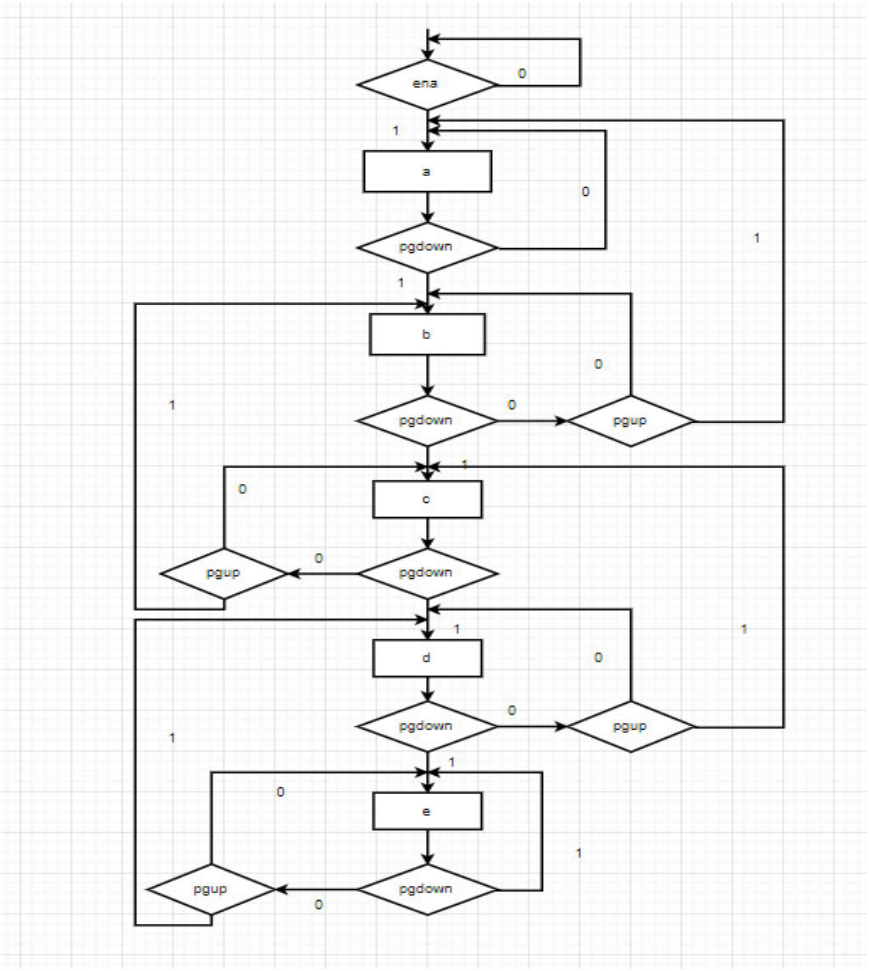
B(n+1)=ena*pgup(A(n)not(B(n))notC(n)+Not(A(n))B(n)C(n))+Not(pgup)*pgdown enaNot(A(n))(B(n)XorC(n)))+(Not(ena)+Not(pgup)Not(pgdown))(Not(A(n)) B(n))

C(n+1)=C(n)*(Not(ena)+Not(pgup)Not(pgdown))+Not(C(n))*(ena*pgup*(A(n)Xor B(n))+enaNot(pgup) Not(pgdown)Not(A(n)))

系统控制器逻辑方案图:

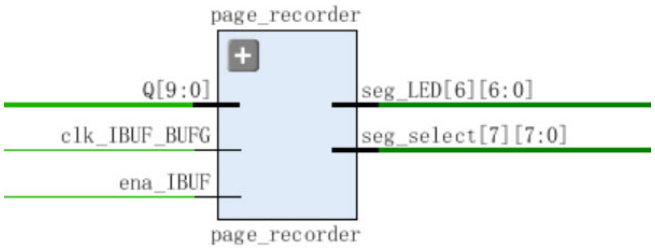


ASM 流程图：



四、子系统模块建模

1.页码显示模块




```
module page_recorder(
        input clk100mhz,
        input [9:0]page,                        //页码数
        input ena,                              //使能信号高电平有效
        output reg[6:0] seg_LED,
        output reg[7:0] seg_select
        );
```

```
wire [3:0] num0;
wire [3:0] num1;
wire [3:0] num2;
wire [3:0] num3;
reg [17:0] clk_cnt = 0;
reg pclk = 0;

assign num0=page%10;
assign num1=page/10%10;
assign num2=page/100%10;
assign num3=page/1000%10;                    //位数计算

always@(posedge clk100mhz)                   //分频
    begin
        if(clk_cnt == 80000)
        begin
            pclk <= ~pclk;
            clk_cnt <= 0;
        end
        else
            clk_cnt <= clk_cnt + 1;
    end

wire [6:0] out0;
wire [6:0] out1;
wire [6:0] out2;
wire [6:0] out3;
wire [6:0] out4;//E
wire [6:0] out5;//g
wire [6:0] out6;//A
wire [6:0] out7;//p

display7 seg0(.ena(ena),.iData(num0),.oData(out0));
display7 seg1(.ena(ena),.iData(num1),.oData(out1));
display7 seg2(.ena(ena),.iData(num2),.oData(out2));
display7 seg3(.ena(ena),.iData(num3),.oData(out3));
assign out4=(ena==0)?7'b1111111:7'b0000110;
assign out5=(ena==0)?7'b1111111:7'b0010000;
assign out6=(ena==0)?7'b1111111:7'b0001000;
assign out7=(ena==0)?7'b1111111:7'b0001100;

always@(posedge pclk)
begin
        cnt <= cnt + 1;
```

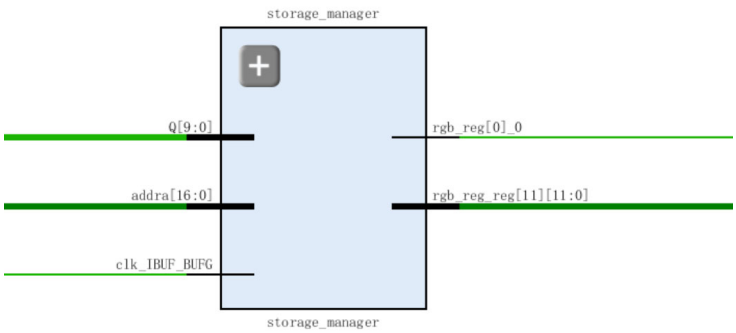
```
        case(cnt)                           //扫描显示  循环点亮
        3'b000:
        begin
            seg_LED<= out0;
            seg_select <= 8'b11111110;
        end
        3'b001:
        begin
            seg_LED<= out1;
            seg_select <= 8'b11111101;
        end
        3'b010:
        begin
            seg_LED<= out2;
            seg_select <= 8'b11111011;
        end
        3'b011:
        begin
            seg_LED <= out3;
            seg_select <= 8'b11110111;
        end
        3'b100:
        begin
            seg_LED <= out4;
            seg_select <= 8'b11101111;
        end
        3'b101:
        begin
            seg_LED <= out5;
            seg_select <= 8'b11011111;
        end
        3'b110:
        begin
            seg_LED <= out6;
            seg_select <= 8'b10111111;
        end
        3'b111:
        begin
            seg_LED <= out7;
            seg_select <= 8'b01111111;
            cnt<=0;
        end
        default:;
        endcase
    end
```

```
module display7(
    input ena,
    input[3:0]iData,
    output reg[6:0] oData
    );
    always@(*)
    if(ena)
    case({iData})
    4'b0000:oData=7'b1000000;
    4'b0001:oData=7'b1111001;
    4'b0010:oData=7'b0100100;
    4'b0011:oData=7'b0110000;
    4'b0100:oData=7'b0011001;
    4'b0101:oData=7'b0010010;
    4'b0110:oData=7'b0000010;
    4'b0111:oData=7'b1111000;
    4'b1000:oData=7'b0000000;
    4'b1001:oData=7'b0010000;
    default:oData=7'b0001001;
    endcase
    else
    oData=7'b1111111;
    endmodule
```

设计思路：

实验使用的多个数码管的连接并不是把每个数码管都独立的与可编程逻辑器件连接，而是把所有的 LED 管的输入连在一起。要使 8 个七段数码管每个显示不一致，必须对 LED 管进行扫描，即依次并循环地点亮各个 LED 管。将时钟分解为合适的频率，利用人眼的视觉暂停效应，在一定的扫描频率下，人眼就会看见多个 LED 一起点亮。该模块实例化 display7 模块，使其能根据输入数字在数码管上显示对应数字。

2.内存管理模块



```
module storage_manager(
          input clk,
          input [9:0]cnt,                          //依其选择对应的 ROM
          input [16:0]address_sig,                 //传给 ROM 对应的地址
          output reg [11:0]rgb                     //输出 ROM 中所存数据
      );
         wire [11:0] page0;
         wire [11:0] page1;
         wire [11:0] page2;
         wire [11:0] page3;
         wire [11:0] page4;

         frontpage frontpage(.clka(clk),.addra(address_sig),.douta(page0));
         pg1 pg1(.clka(clk),.addra(address_sig),.douta(page1));
         pg2 pg2(.clka(clk),.addra(address_sig),.douta(page2));
         pg3 pg3(.clka(clk),.addra(address_sig),.douta(page3));
         pg4 pg4(.clka(clk),.addra(address_sig),.douta(page4));

         always@(posedge clk)
         begin
            if(cnt==10'd0)
                  rgb<=page0;
            else if(cnt==10'd1)
                  rgb<=page1;
            else if(cnt==10'd2)
                  rgb<=page2;
            else if(cnt==10'd3)
                  rgb<=page3;
            else
                  rgb<=page4;
         end
endmodule
```

设计思路：

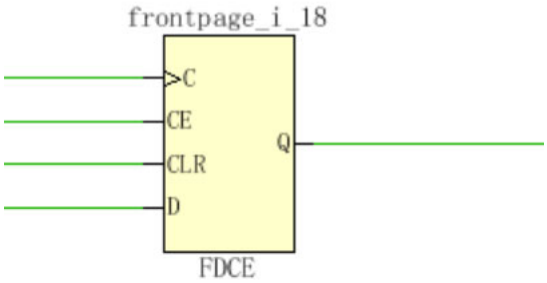
2.1 将图片内容存入内存中

先将图片转换为 ROM 可以识别的.coe 文件格式,考虑到板上的 ROM 的大小和漫画每页的大小，我先将漫画大小统一设置为 320*240 像素，生成 24 位的 bmp 文件格式，再利用 MATLAB 软件，将其转换为.coe 文件，MATLAB 源码如下：

```
function imgcoe(path,name)
    % 利用 imread 函数把图片转化为一个三维矩阵
    image_array = imread(path);
```

```
% 利用 size 函数把图片矩阵的三个维度大小计算出来
[height,width,z]=size(image_array);

red = image_array(:,:,1); % 提取红色分量，数据类型为 uint8
green = image_array(:,:,2); % 提取绿色分量，数据类型为 uint8
blue = image_array(:,:,3); % 提取蓝色分量，数据类型为 uint8

% 把上面得到了各个分量重组成一个 1 维矩阵，由于 reshape 函数重组矩阵的时候是
%按照列进行重组的，所以重组前需要先把各个分量矩阵进行转置以后再重组
% 为了避免溢出这里把 uint8 类型的数据扩大为 uint32 类型
r = uint32(reshape(red'   , 1 ,height*width));
g = uint32(reshape(green' , 1 ,height*width));
b = uint32(reshape(blue'  , 1 ,height*width));

% 初始化要写入.coe 文件中的 RGB 颜色矩阵
rgb=zeros(1,height*width);

% 导入的图片是 24-bit 真彩色图片，每个像素占用 24-bit，其中 RGB 分别占用 8-bit
% 需要的是 12-bit，RGB 分别占用 4-bit，所以需要在这里对 24-bit 的数据进行重组拼接
% bitshift()函数的作用是对数据进行移位操作，其中第一个参数是要进行移位的数据，
%第二个参数为负数表示向右移，为正数表示向左移
for i = 1:height*width
rgb(i) = bitshift(bitshift(r(i),-4),0) + bitshift(bitshift(g(i),-4),4) +
bitshift(bitshift(b(i),-4),8);
end

fid = fopen( name , 'w+' );

% 16 表示 16 进制
fprintf( fid, 'memory_initialization_radix=16;\n');
fprintf( fid, 'memory_initialization_vector =\n');

% 把 rgb 数据的前 height*width-1  个数据写入.coe 文件中，每个数据之间用逗号隔开
fprintf( fid, '%x,\n',rgb(1:end-1));

% 把 rgb 数据的最后一个数据写入.coe 文件中，并用分号结尾
fprintf( fid, '%x;',rgb(end));

fclose( fid ); % 关闭文件指针
end
```

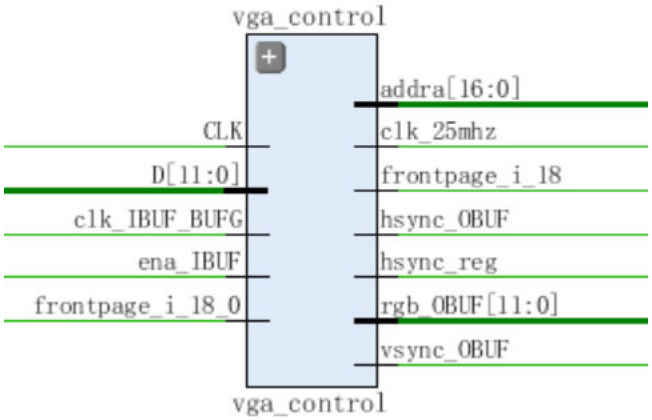
得到图片后，生成对应的 IP 核(数量和漫画页数相同)。



2.2IP 核的选择

根据传入的 cnt 选择对应的 ROM，将其所存数据传给 reg 接口

3.VGA 控制模块




```
module vga_control(
        input wire clk100mhz,
        input [11:0]qsig,
        input    sys_rst_n,                          //WHEN 1, DISPLAY
        output HSync,
        output VSync,
        output [11:0] rgb,
        output wire[16:0]address_sig,
        output [10:0]pixel_x,
        output [10:0]pixel_y
        );

reg clk_25mhz;
reg [11:0]    rgb_reg;
reg [18:0]address_sig_19;

assign address_sig=address_sig_19[16:0];
assign rgb = (video_en == 1'b1) ? rgb_reg:12'b0;
```

```
always @ (posedge clk_25mhz or negedge sys_rst_n)
    if(!sys_rst_n)
        begin
            rgb_reg <= 12'b0;
        end
    else
        rgb_reg[11:0] <= qsig[11:0];

always @ (posedge clk_25mhz or negedge sys_rst_n)
    if(!sys_rst_n)
        begin
            address_sig_19 <= 19'b0;
        end
    else
    begin
        if(pixel_x>=0 && pixel_x<= 639 && pixel_y>=0 && pixel_y<=479)
        address_sig_19 = (pixel_x/2 + pixel_y/2*320);
    end

    //分频
    reg clk50mhz;
  always@(posedge(clk100mhz))
            begin
                clk50mhz <= ~clk50mhz;
            end
            always@(posedge(clk50mhz))
                begin
                    clk_25mhz <= ~clk_25mhz;
                end

vga vga_uut(
            .clk            (clk_25mhz),
            .rst_n          (sys_rst_n),
            .video_en       (video_en),
            .hsync          (HSync),
            .vsync          (VSync),
            .pixel_x        (pixel_x),
            .pixel_y        (pixel_y)
        );

endmodule
```

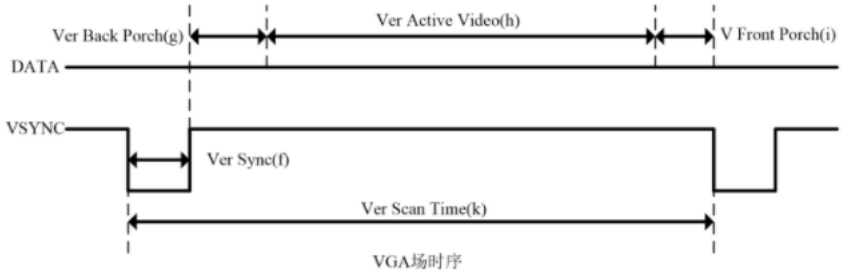
```
module vga(
        input   wire            clk,
        input   wire            rst_n,
        output  wire            video_en,           //数据有效
        output  reg             hsync,              //场同步信号
        output  reg             vsync,              //行同步信号
        output  wire    [10:0]  pixel_x,            //待显示像素的 x 坐标
        output  wire    [10:0]  pixel_y             //待显示像素的 y 坐标
        );

        reg     [10:0] x_cnt;
        reg     [10:0] y_cnt;

        reg             v_video_en;                 //竖直有效
        reg             h_video_en;                 //水平有效
        parameter PAL = 640;                        //Pixels/Active Line (pixels)
        parameter LAF = 480;                        //Lines/Active Frame (lines)
        parameter PLD = 800;                        //Pixel/Line Divider
        parameter LFD = 525;                        //Line/Frame Divider

        always @(posedge clk or negedge rst_n)
        begin
            if(!rst_n)
                x_cnt <= 10'b0;
            else
            begin
                if( x_cnt == PLD-1 )                //have reached the edge of one line
                begin
                   x_cnt <= 0;                      //reset the horizontal counter
                if( y_cnt == LFD-1 )
              //only when horizontal pointer reach the edge can the vertical counter ++
                    y_cnt <=0;
                else
                    y_cnt <= y_cnt + 1;
                end
                else
                    x_cnt <= x_cnt + 1;
              end
          end

        always @(posedge clk or negedge rst_n)
            if(!rst_n)
            begin
```

```
                h_video_en <= 1'b1;
                hsync <= 1'b1;
            end
            else
            begin
              case (x_cnt)
                10'd0:
                  h_video_en <= 1'b1;
                PAL-1:
                  h_video_en <= 1'b0;
                PAL+15:
                  hsync <= 1'b0;
                PAL+111:
                  hsync <= 1'b1;
               default:;
                        endcase
                  end

        always @(posedge clk or negedge rst_n)
            if(!rst_n)
            begin
                v_video_en <= 1'b1;
                vsync <= 1'b1;
            end
            else
                case(y_cnt)
                  10'd0:
                          v_video_en <= 1'b1;
                  LAF-1:
                            v_video_en <= 1'b0;
                  LAF+9:
                            vsync <= 1'b0;
                  LAF+11:
                            vsync <= 1'b1;
                  default:;
                endcase

        assign pixel_x = x_cnt;
        assign pixel_y = y_cnt;

    assign video_en = ((h_video_en == 1'b1) &&    (v_video_en == 1'b1)); //水平竖直都有效整体才有效
endmodule
```

设计思路：

显示器扫描方式分为逐行扫描和隔行扫描：逐行扫描是扫描从屏幕左上角一点开始，从左向右逐点扫描，每扫描完一行,电子束回到屏幕的左边下一行的　起始位置，在这期间，CRT 对电子束进行消隐，每行结束时，用行同步信号进行同步；当扫描完所有的行，形成一帧，用场同步信号进行场同步，并使扫描回到屏幕左上方，同时进行场消隐,开始下一帧。隔行扫描是指电子束扫描时每隔一行扫一线，完成一屏后在返回来扫描剩下的线。

VGA行时序



VGA场时序

在有横纵坐标范围的区域上的每一个坐标点就是一个像素点，可以给这个像素点特定的 rgb 色彩，既可以通过自定义 rgb，也可以去取图像某个像素点的 rgb。

VGA 常用分辨率时序参数

显示模式	时钟	行时序参数(单位：像素)					列时序参数(单位：行)				
	/MHz	a	b	c	d	e	f	g	h	i	k
640x480@60Hz	25.175	96	48	640	16	800	2	33	480	10	525
800x600@60Hz	40	128	88	800	40	1056	4	23	600	1	623
1024x768@60Hz	65	136	160	1024	24	1344	6	29	768	3	806
1280x720@60Hz	74.25	40	220	1280	110	1650	5	20	720	5	750
1280x1024@60Hz	108	112	248	1280	48	1688	3	38	1024	1	1066
1920x1080@60Hz	148.5	44	148	1920	88	2200	5	36	1080	4	1125

根据参数将时钟分频为 25mhz

五、测试模块建模

1.VGA 测试模块

```
module vga_test(
                        input    wire              sys_clk,
                        input    wire              sys_rst_n,
                        output   wire              hsync,
                        output   wire              vsync,
                        output   wire    [11:0]    rgb
                    );

wire      [9:0]    pixel_x;
wire      [9:0]    pixel_y;
reg               clk_25mhz;
reg               clk_50mhz;
reg       [11:0]   rgb_reg;
reg       [18:0]   address_sig;
wire      [11:0]   q_sig;

   always@(posedge(sys_clk))
               begin
                    clk_50mhz <= ~clk_50mhz;
               end
               always@(posedge(clk_50mhz))
                    begin
                         clk_25mhz <= ~clk_25mhz;
                    end

//显示静态图像
always @ (posedge clk_25mhz or negedge sys_rst_n)
      if(!sys_rst_n)
           begin
                rgb_reg <= 12'b0;
           end
      else
           begin                                        //显示图像
           //rgb_reg <= q_sig;
          rgb_reg[11:0] <=q_sig[11:0];
      end
always @ (posedge clk_25mhz or negedge sys_rst_n)
      if(!sys_rst_n)
           begin
                address_sig <= 19'b0;
           end
```

```
            end
        else
        begin
            if(pixel_x>=0 && pixel_x<= 639 && pixel_y>=0 && pixel_y<=479)
            address_sig = (pixel_x/2 + pixel_y/2*320);
        end

assign rgb = (video_en == 1'b1) ? rgb_reg:12'b0;

vga_sync vga_syn_inst(
                    .clk                (clk_25mhz),
                    .rst_n              (sys_rst_n),
                    .video_en           (video_en),
                    .hsync              (hsync),
                    .vsync              (vsync),
                    .pixel_x            (pixel_x),
                    .pixel_y            (pixel_y)
            );

blk_mem_gen_2 blk_mem_gen_2 (
   .clka(clk_25mhz),                                // input wire clka
   .addra(address_sig),                              // input wire [18 : 0] addra
   .douta(q_sig)                                     // output wire [11 : 0] douta
);

endmodule
```

2.数码管模块

2.1display7 模块

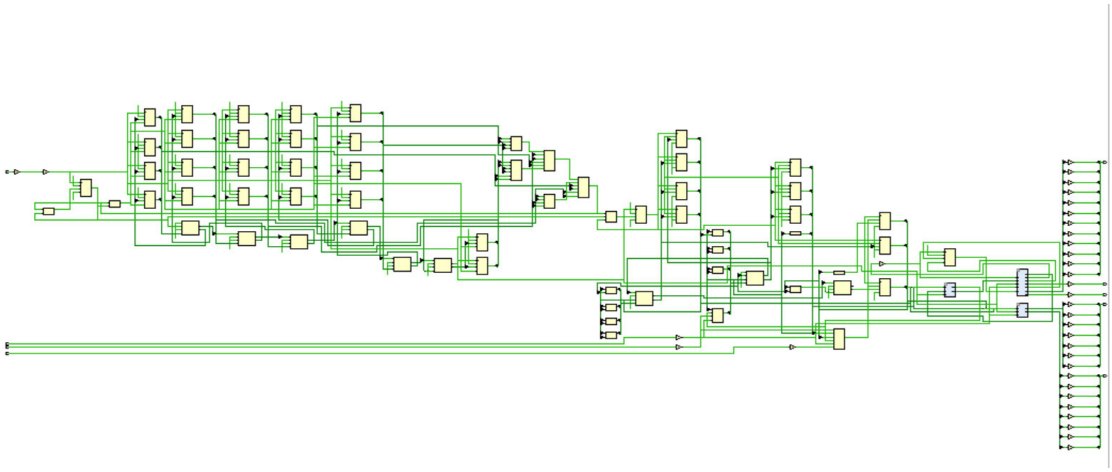
```
`timescale 1ns / 1ps
module display7_tb;
reg[3:0]iData;
wire[6:0] oData;
display7 uut(
.iData(iData),.oData(oData));
initial
begin
iData=4'b0000;
   #20 iData=4'b0001;
   #20 iData=4'b0010;
   #20 iData=4'b0011;
   #20 iData=4'b0100;
   #20 iData=4'b0101;
   #20 iData=4'b0110;
```

```
#20 iData= 4'b0111;
   #20 iData=4'b1000;
   #20 iData=4'b1001;
   #20 iData=4'b1010;
end
endmodule
```

剩余模块直接下板验证，未生成 test bench。

六、实验结果

1.RTL 模块：



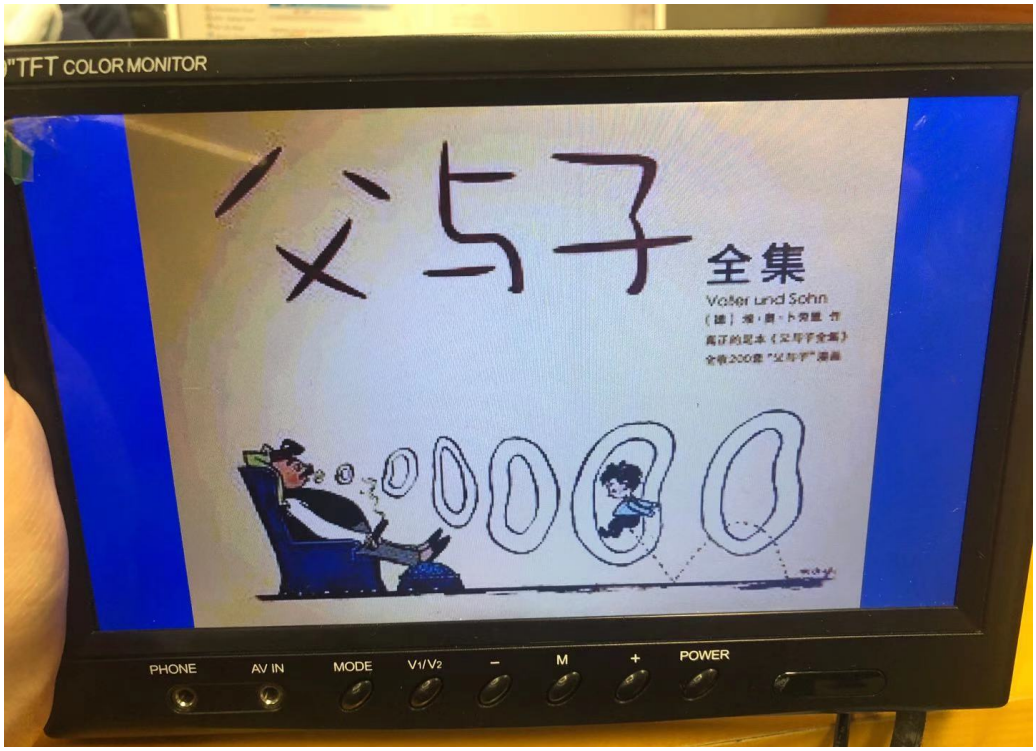
1.VGA 测试模块下板：

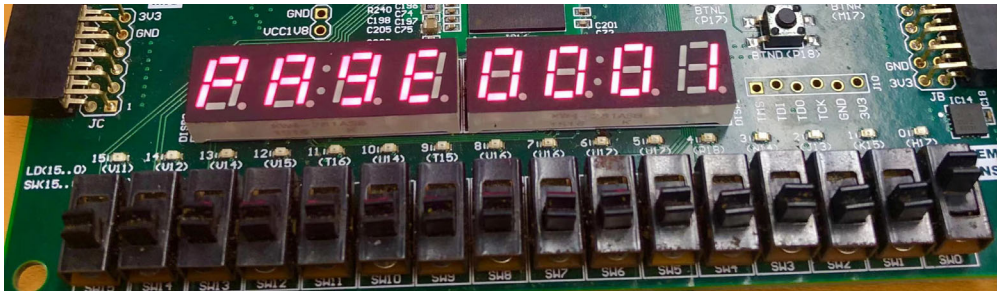
（任意一张图片）

3.综合下板后的演示

3.1 生成 bit 流后打开开关

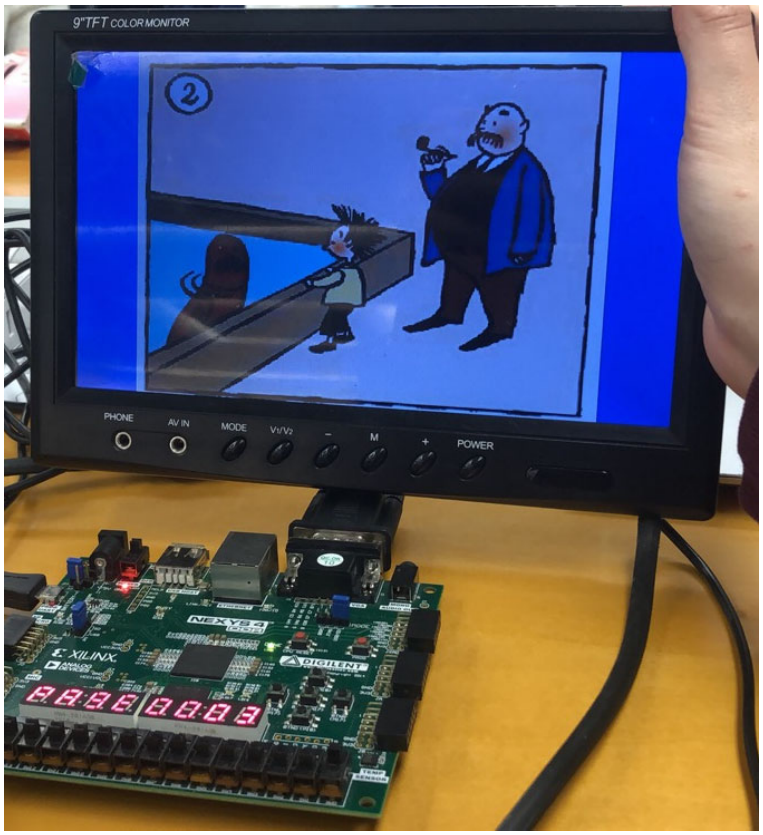
显示首页和对应页码





3.2 按下 P17 上翻漫画，按 M17 可对应下翻页

页码对应显示正确，VGA 对应显示正确，两者相互匹配



3.3 下推开关，即关闭设备，此时数码管和显示屏均无信号