中国科学技术大学计算机学院

《数字电路实验》报告

实验题目：存储器

学生姓名：柯志伟

学生学号：PB20061338

完成日期：2021 年 12 月 12 日

计算机实验教学中心制

2020 年 09 月

【实验题目】

存储器

【实验目的】

- 了解熟、悉存储器的使用并借此熟悉 Vivado IP 核的使用
- 理解 VGA 显示图像的原理，并能正确使用开发板的 VGA 接口实现相关功能
- 实现绘画功能，并借由 VGA 接口显示在显示屏上
- 选做
  - 实现不同分辨率的显示
  - 在显示屏上以十字光标标识光标的位置
  - 实现画布的背景功能, 可擦除
  - 输入数字并在显示屏上显示

【实验环境】

- PC 机
- Vivado 平台
- Nexys4 开发板

【实验原理】

- VGA 接口功能

  VGA 接口共有 15 个孔，分成 3 排，每排 5 个孔，分别传输红、绿、蓝模拟信号模拟信号以及同步信号 (水平和垂直信号)。红、蓝、绿信号的数值分别控制三种颜色的亮度。VGA 信号显示图片时，并不是直接显示所有像素点，而是一个点一个点地依次显示，由于显示频率非常高加上视觉残留，可以达到视觉上同时显示地效果。由于画面尺寸不是固定的，因此需要一个用于换行的水平同步信号（H-Sync），来向显示器传递换行信号。当绘制到画面的最后一个像素，需要一个用来回到开头的垂直同步信号（V-Sync），来传递回溯开头新号。

【实验过程】

- 用 Verilog 实现 VDT,VDS 模块, 其中 VDT 信号输出 VGA 的行列同步信号,VDS 输出要显示的 RGB 的值，并起到适配显示屏分辨率的作用 (通过精确控制要输出的颜色), 如本次实验中要将画布信息显示在显示屏上, 画布分辨率是是 200*150，而显示屏是 800*600, 因此要将画布放大 16 倍（行列均放大 4 倍），通过一个行计数器，一个列计数器，以 4 为周期, 行计数为 4 才读画布下一像素点信息，至行尾回溯 200 个像素点

直至列计数为 4, 以此将画布一个像素点信息放大为显示屏上 16 个像素点的信息
代码如下:

- VDT

```
module VDT (input pclk,
        output  hs,vs,
        output reg hen,ven
        );
    parameter henMax=1039 ,venMax=665;
    //HBP =64   ,HFP = 56,VBP = 23   ,VFP= 37 ;
    parameter UP_BOUND = 29;
    parameter DOWN_BOUND = 628;
    parameter LEFT_BOUND = 184;
    parameter RIGHT_BOUND = 983;
    parameter HSW = 120,VSW=6;
    reg [10:0] hcnt=0;
    reg [9:0] vcnt=0;

    always @(posedge pclk) begin
        if(hcnt==henMax)begin
            hcnt <= 0;
            vcnt <=(vcnt==venMax)?0:(vcnt+1);
        end
        else hcnt <= hcnt +1;
    end

    always @(posedge pclk) begin
        hen <= (hcnt>=LEFT_BOUND)&&(hcnt <=RIGHT_BOUND);
        ven <= (vcnt>=UP_BOUND)&&(vcnt <=DOWN_BOUND);
    end
    assign hs = (hcnt>(HSW-1))?0:1;
    assign vs = (vcnt>(VSW-1))?0:1;
endmodule
```

- VDS

```

module VDS( input pclk,
    input hen,ven,
    input [11:0] rdata,
    output reg [14:0] raddr,
    output reg [11:0] prgb);
        parameter HSize = 4,VSize = 4;
        reg [1:0] hscnt,vscnt;
```

```
9           reg hen_reg;
10          wire nextline;
11
12          assign nextline = hen_reg&&(!hen);
13          always @(posedge pclk) begin
14          hen_reg <= hen;
15          end
16
17          always @(posedge pclk) begin
18          if(hen)  hscnt <= (hscnt==(HSize-1))?0:(hscnt+1);
19          else  hscnt <=0;
20          end
21
22          always @(posedge pclk) begin
23          if(ven) begin
24              if(nextline)
25    vscnt <= (vscnt==(VSize-1))?0:(vscnt+1);
26          end
27          else vscnt <= 0;
28          end
29
30          always @(posedge pclk) begin
31              if(hen&&ven)begin
32              if(hscnt==(HSize-1)) raddr <= raddr+1;
33              end
34              else  begin
35              if(nextline)begin
36                  if(vscnt<(VSize-1)) raddr <= raddr -200;
37              end
38              if((!hen)&&(!ven))
39          raddr <= 15'b0000_0000_0000_000;
40              end
41              end
42
43          always @(posedge pclk) begin
44              if(hen&&ven)begin
45                  prgb <= rdata;
46              end
47              else prgb <= 12'h000;
48
49              end
50  endmodule
```

– 将 VDT,VDS 组合为 DCU

```
module DCU(input  clk,
           output  [11:0] prgb,
           output hs,vs
           );
    wire pclk;
    wire hen,ven;
    wire [5:0] raddr;
    wire [1:0] rdata;

    //获取50MHz的时钟
    clk_wiz_0 PCLK(.clk_in1(clk),.clk_out1(pclk));
    blk_mem_gen_0 RAM(.clkb(pclk),.enb(1),
    .addrb(raddr),.doutb(rdata));
    //仅使用读端口即B端口

    VDT vdt(.pclk(pclk),.hs(hs),.vs(vs),
    .hen(hen),.ven(ven));
    VDS vds(.pclk(pclk),.hen(hen),.ven(ven),.rdata(rdata),
    .raddr(raddr),.prgb(prgb));

endmodule
```

- 用 Verilog 实现画笔控制模块 PCU
  代码如下:

```

module PCU(input  clk,rstn,
    input [3:0] dir,
    input [11:0] rgb,
    input draw,
    output reg [14:0] waddr,
    output  [11:0] wdata,
    output  we );                    //绘画控制单元
    always @(posedge clk,negedge rstn) begin
    if(!rstn) begin
    waddr <= 15'd14999;
    end
    else begin
    if((!dir[0])&&(!dir[1]&&(!dir[2])&&(dir[3]))) //左
    waddr <= waddr -1;
    else if((!dir[0])&&(!dir[1]&&(dir[2])&&(!dir[3]))) //下
    waddr <= waddr + 200;
```

```
    else if((!dir[0])&&(dir[1]&&(!dir[2])&&(!dir[3]))) //右
    waddr <= waddr +1;
    else if((!dir[0])&&(dir[1]&&(!dir[2])&&(!dir[3]))) //上
    waddr <= waddr - 200;
    end
    end
    assign we = draw;
    assign wdata = rgb;

endmodule
```

- 组合 PCU,DCU, 并例化存储器 IP 核实现绘画显示功能, 带有擦除功能，背景图像, 十字光标显示画笔, 通过例化两个存储器，一个存画布信息，一个存储背景图像，实现一个选择器选择输出 VGA 接口的 RGB（初始画布信息全为 12'h000, 选择背景信息输出，否则输出画布信息），以实现画笔的十字光标显示以及背景图像, 另外输入一个 erase 信号, 有效时可擦出画迹（恢复显示背景图像，将画布信息重置为 12‘h000）
  代码如下：

```
module  painter (
            input clk, rstn,//时钟, 复位
            input [3:0] dir,
            input [11:0] rgb,
            input erase,
        //    input [3:0] red,          //画笔颜色 (rgb): 红
        //    input [3:0] green,        //画笔颜色 (rgb): 绿
        //    input [3:0] blue,         //画笔颜色 (rgb): 蓝
            input draw,          //
            output  [11:0] prgb,
        //    output [3:0] pred,        //像素颜色 (prgb): 红
        //    output [3:0] pgreen,      //像素颜色 (prgb): 绿
        //    output [3:0] pblue,       //像素颜色 (prgb): 蓝
            output hs,           //
            output vs            //
        );
        reg [11:0] mdata,mprgb;
        wire [14:0] waddr,raddr;
        reg [11:0] wdata_reg;
        wire [11:0] wdata,rdata;
        wire [14:0] raddr2;
        wire [11:0] rdata2;
        wire we;
        reg delay;
        always @(posedge clk) begin
```

```
26            if((raddr == (waddr +1))||(raddr == (waddr -1))
27            ||(raddr == (waddr +200))
28            ||(raddr == (waddr -200))||(raddr == waddr))
29              delay <= 1;
30            else delay <= 0;
31          end
32          always @(*) begin
33              begin
34                if(delay)
35                  mdata = 12'h000;
36                else if(rdata == 12'h000) mdata = rdata2;
37                else mdata = rdata;
38            end
39          end
40
41          always @(posedge clk) begin
42            mprgb <= mdata;
43          end
44
45          always @(posedge clk) begin
46            if(erase) wdata_reg <= 12'h000;
47            else wdata_reg <= wdata;
48          end
49
50          clk_wiz_0 clkmaker(.clk_in1(clk),.clk_out1(pclk));
51          PCU pcu(pclk,rstn,draw,dir,rgb,we,waddr,wdata);
52
53          DCU dcu(pclk,mprgb,hs,vs,raddr,prgb);
54          assign raddr2 = raddr;
55          blk_mem_gen_0 vram(.addra(waddr),.clka(clk),
56          .dina(wdata_reg),.ena(1),.wea(we),
57          .addrb(raddr),.clkb(pclk),.doutb(rdata),.enb(1));
58          blk_mem_gen_1 rom(.addra(raddr2),.clka(pclk),
59          .douta(rdata2),.ena(1));
60
61          endmodule
62
63
64          module PCU (
65              input clk,rstn,draw,
66              input [3:0] dir,//
67              input [11:0] rgb,
```

```
68              output wire we,
69              output reg [14:0] waddr,
70              output wire [11:0] wdata
71          );
72          reg [3:0] last_dir;
73
74              always@(posedge clk)
75              begin
76                last_dir<=dir;
77              end
78
79              always @(posedge clk or negedge rstn)
80              begin
81                if(!rstn)waddr <= 15'd14899;
82                else if((!last_dir[3]&&dir[3])||(!last_dir[2]
83                &&dir[2])||(!last_dir[1]&&dir[1])||(!last_dir[0]
84                &&dir[0]))
85                  case(dir)
86                  4'b1000:waddr <= waddr - 200;
87                  4'b0100:waddr <= waddr + 200;
88                  4'b0010:waddr <= waddr - 1;
89                  4'b0001:waddr <= waddr + 1;
90                  default :waddr <= waddr;
91                endcase
92                else waddr <= waddr;
93              end
94
95              assign we = draw;
96              assign wdata = rgb;
97
98
99          endmodule
100
101
102
103
104
105         module VDS#(parameter TIMES = 4 )(
106                   input pclk,
107                     input hen,ven,
108                     input [11:0] rdata,
109                     output reg [14:0] raddr=0,
```

```
110                 output reg [11:0] prgb
111         );
112         reg [10:0] xcnt=0,ycnt=0;
113         reg reg_hen;
114
115         //记录换行
116         always@(posedge pclk)
117         begin
118           reg_hen<=hen;
119         end
120
121         //将一个像素点扩大为TIMES^2个，cnt计数
122           always@(posedge pclk)
123           begin
124             if(ven)
125             begin
126               if(hen)
127                 begin
128                 if(xcnt==TIMES-1)
129                   xcnt<=0;
130                 else
131                   xcnt<=xcnt+1;
132                 end
133
134               if(!hen&&reg_hen)
135                 begin
136                 if(ycnt==TIMES-1)
137                 ycnt<=0;
138                 else ycnt<=ycnt+1;
139                 end
140             end
141           end
142
143         //放大
144         always@(posedge pclk)
145         begin
146           if(ven)
147           begin
148                 if(!hen&&reg_hen&&ycnt!=TIMES-1)
149             raddr<=raddr-800/TIMES;
150                 else if(hen&&xcnt==TIMES-1)raddr<=raddr+1;
151           end
```

```
152             if(!ven)
153             begin
154                 raddr<=0;
155             end
156         end
157
158         //输出prgb
159         always@*
160         begin
161             if(hen&&ven)
162             prgb=rdata;
163             else prgb=0;
164         end
165
166
167         endmodule
168
169
170         module DCU(
171             input pclk,
172             input [11:0] rdata,
173             output hs,vs,
174             output [14:0] raddr,
175             output [11:0] prgb
176         );
177         wire hen,ven;
178
179         VDS vds(pclk,hen,ven,rdata,raddr,prgb);
180
181         VDT vdt(pclk,hs,vs,hen,ven);
182
183         endmodule
184
185
186         module VDT (input pclk,
187                     output wire hs,vs,
188                     output reg hen,ven
189         );
190             parameter hmax=1040 ,vmax=666;
191             parameter UP_BOUND = 29;
192                 parameter DOWN_BOUND = 628;
193                 parameter LEFT_BOUND = 184;
```

```
194             parameter RIGHT_BOUND = 983;
195         parameter HSW = 120,VSW=6;
196 
197 
198         reg [11:0] hcnt=0;
199         reg [11:0] vcnt=0;
200 
201 
202         always@(posedge pclk)
203         begin
204             if(hcnt==hmax-1)
205                 begin
206                         hcnt=0;
207                         if(vcnt==vmax-1)vcnt<=0;
208                         else vcnt=vcnt+1;
209                 end
210             else  hcnt=hcnt+1;
211 
212         end
213 
214         assign hs=(hcnt<HSW);
215         assign vs=(vcnt<VSW);
216 
217 
218         always@(posedge pclk)
219         begin
220                 if((hcnt>=LEFT_BOUND)&&(hcnt<=RIGHT_BOUND))
221                     hen <= 1;
222                 else hen<=0;
223                 if((vcnt>=UP_BOUND)&&(vcnt<=DOWN_BOUND))
224                     ven <= 1;
225                 else ven<= 0;
226         end
227 
228     endmodule
```

- 实现显示数字的功能，实现确定各个字符的显示需要的像素值信息，选择进行输出

  由于有些行代码过长，不便放于报告，于是放入压缩包中

- 实验中的电路图、电路资源使用情况、实物图

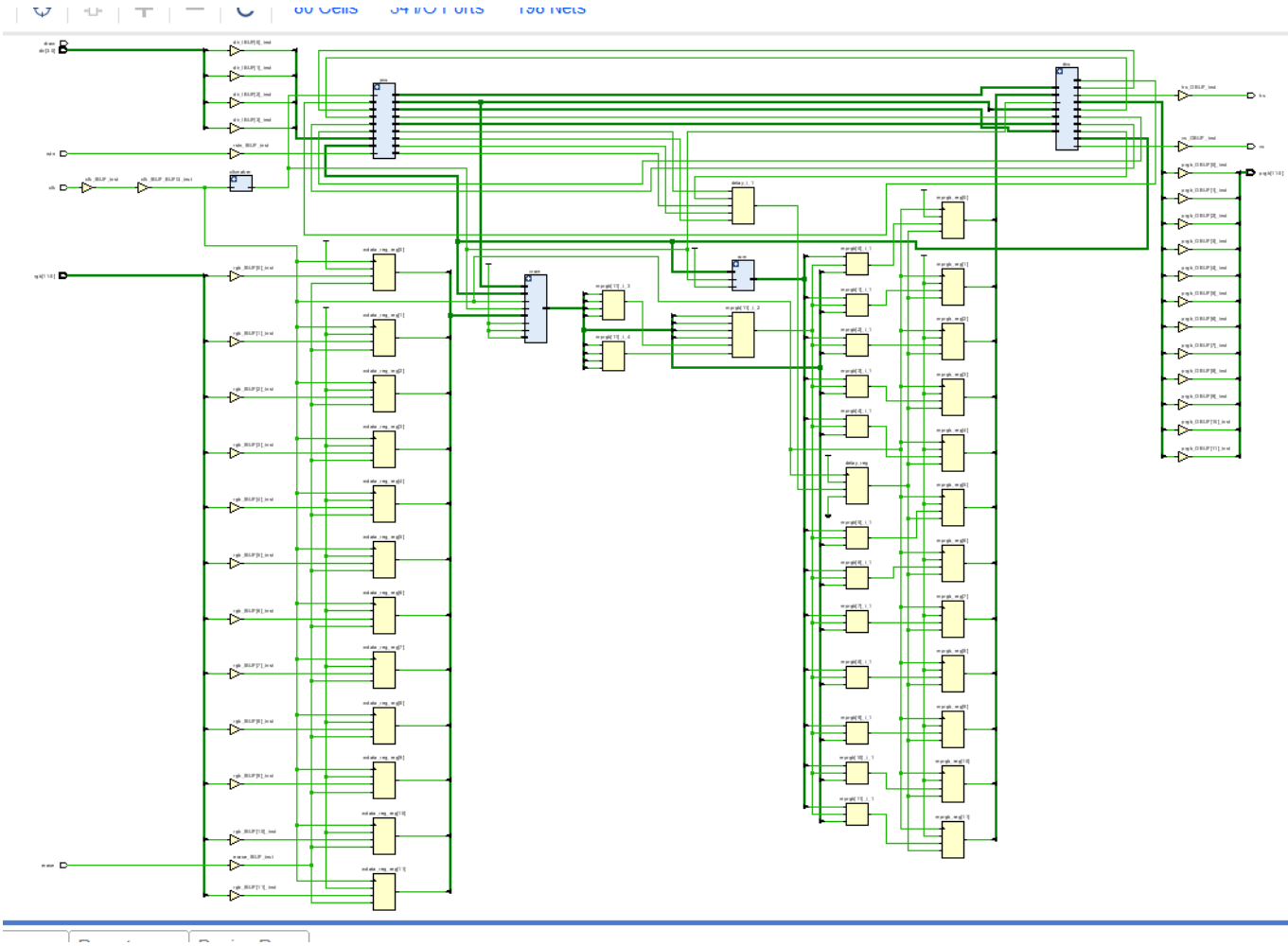

图 1: 画笔绘画并显示的电路图

—

Hierarchy

Name	Slice LUTs (63400)	Slice Registers (126800)	F7 Muxes (31700)	Block RAM Tile (135)	Bonded IOB (210)	BUFGCTRL (32)	PLLE2_ADV (6)
painter	208	102	18	22	33	4	1
clkmaker (clk_wiz_0)	0	0	0	0	0	3	1
dcu (DCU)	53	46	0	0	0	0	0
pcu (PCU)	85	19	0	0	0	0	0
rom (blk_mem_gen_1)	26	6	9	11	0	0	0
vram (blk_mem_gen_0)	34	6	9	11	0	0	0

图 2: 画笔绘画并显示的电路资源使用情况

—

图 3: 画笔绘画并显示的实物图

–

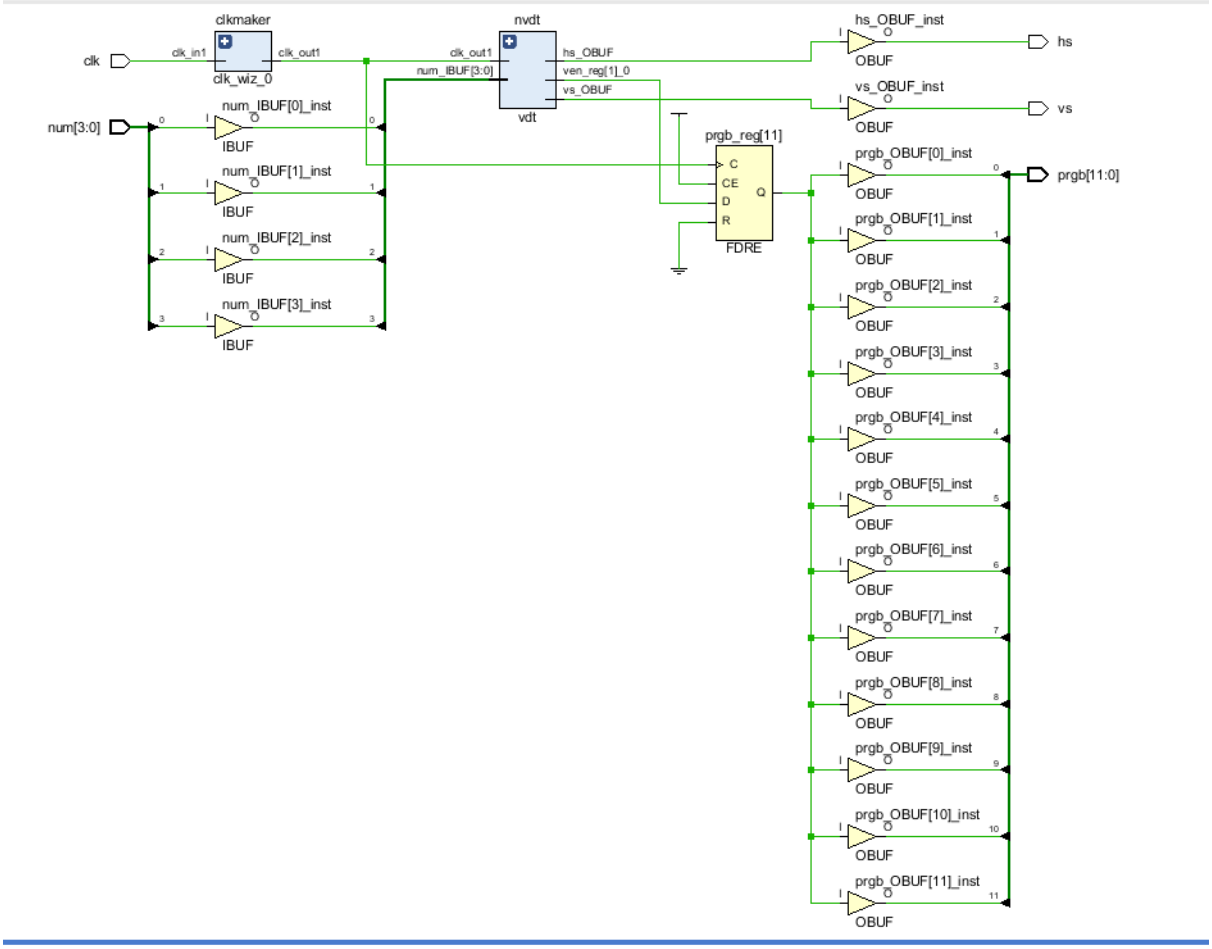

图 4: 数字显示的电路图

—

Hierarchy

Name	Slice LUTs (63400)	Slice Registers (126800)	F7 Muxes (31700)	F8 Muxes (15850)	Bonded IOB (210)	BUFGCTRL (32)	PLLE2_ADV (6)
disnum	77	33	10	5	19	2	1
clkmaker (clk_wiz_0)	0	0	0	0	0	2	1
inst (clk_wiz_0_clk_	0	0	0	0	0	2	1
nvdt (vdt)	77	32	10	5	0	0	0

图 5: 数字显示的电路资源使用情况

—

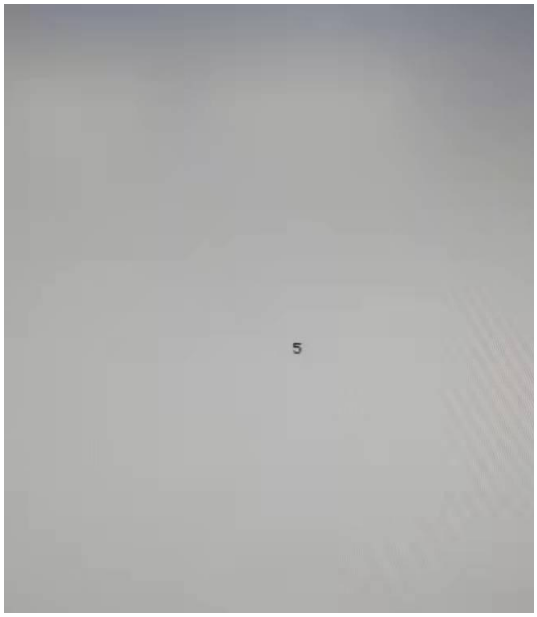

图 6: 数字显示的实物图

–

【总结与思考】

- 总结:

  通过此次实验，了解了 VGA 的显示原理，进一步运用分模块设计的思想，了解 Vivado 的 IP 核的例化使用，同时在此次实验中也遇到了一些问题: 在画笔十字光标显示时，发现画笔的位置与画笔的显示位置相离一个像素 (画笔的显示位置早了一个时钟周期)，于是尝试延迟画笔显示位置的输出，尝试了两个方法: 1. 将画笔的显示位置信息寄存，下一个时钟周期输出 2. 再使用一个 always 块 (选择器)，选择输出画笔位置或图像信息单独使用都失败了，但混合使用解决了问题

- 思考:

  这次实验中显示数字的实验，我采用很多个选择器实现逻辑电路来显示数字，代码很长，实际上可以预先在一个存储器中存储数字的显示信息，要显示是再读取，可以简化很多代码

- 建议:

  在此次实验感觉在 VGA 显示的原理方面花了很多时间研究，最后看了 VGA 的驱动代码才理解它的原理，感觉直接看原理理论不如具体的例子，建议老师在讲一些器件的原理时能具体展示几个例子