

计算机组成原理实验报告

- 实验题目：存储器与显示控制器
- 实验日期：2019年4月18日
- 姓名：张劲瞰
- 学号：PB16111485
- 成绩：

IP (Intellectual Property) 内核模块是一种预先设计好的甚至已经过验证的具有某种确定功能的集成电路、器件或部件。

ROM的IP核生成需要初始化文件，这个初始化的文件后缀是 .coe

其中：`memory_initialization_radix` 是数值格式 `memory_initialization_vector` 是初始化的数值向量，分别对应各个深度

注意：

- radix 其实就是进制，下面的 vector 不能出现超过 radix 的数据。
- 因为coe文件最后会被转化为mif文件，即最后都是二进制表示，所以需要考虑好数值范围的问题，否则可能出错，例如 ROM 的数据宽度为 3，初始化文件中出现 8 或以上的数据，则可能生成失败，或者只取了低3位。

对于一个刷新频率为**72Hz**，分辨率为**800X600**的SVGA显示驱动，若它的基准驱动时钟为50MHz，它的计数脉冲参数如表所示。注意列的单位为“行”，而行的单位为“基准时钟周期数”，即**50MHz时钟脉冲数**。SVGA驱动时序参数表

行/列	同步脉冲	后沿脉冲	显示脉冲	前沿脉冲	帧长
列	6	23	600	37	666
行	120	64	800	56	1040

实验目的：

1. 控制画笔在800x600分辨率的显示器上随意涂画，画笔的颜色12位(红r绿g蓝b各4位)，绘画区域位于屏幕正中部，大小为256x256
 - 画笔位置(x, y)：x = y = 0 ~ 255，复位时 (128, 128)
 - 移动画笔(dir)：上/下/左/右按钮
 - 画笔颜色(rgb)：12位开关设置
 - 绘画状态(draw)：1-是，0-否；处于绘画状态时，移动画笔同时绘制颜色，否则仅移动画笔
2. **VRAM**：视频存储器，存储256x256个像素的颜色信息，采用简单双端口存储器实现

- **paddr**, **pdata**, **we**: 地址、数据、写使能, 用于绘画 的同步写端口
- **vaddr**, **vdata**: 地址、数据, 用于显示的异步读端口

3. **PCU**: Paint Control Unit, 绘画控制单元, 修改**VRAM**中像素信息

- 通过12个拨动开关设置像素颜色 (**rgb**)
- 通过上/下/左/右(**dir**)按钮开关, 移动画笔位置(**x**, **y**)
 - 直角移动: 单一按钮按下一次, **x**或**y**增加或减小1
 - 对角移动: 两按钮同时按下一次, **x**和**y**同时加或减1
 - 连续移动: 按钮按下超过**t**秒后, 等效为**s**速率的连续 点击, 直至松开 (调试时确定合适的**t**和**s**取值)
- 绘画(**draw=1**)时, 依据**rgb**和(**x**, **y**), 通过写端口(**paddr**, **pdata**, **we**)修改**VRAM**像素信息

4. **DCU**: Display Control Unit, 显示控制单元, 显示**VRAM**中像素信息

- 通过读端口(**vaddr**, **vdata**)取出**VRAM**信息并显示
- **vrgb**, **hs**, **vs**: 显示器接口信号显示模式: 分辨率800x600, 刷新频率72Hz, 像素时钟频率50MHz
VRAM中的1个像素对应显示屏上1个像素
- 在屏幕上显示十字光标, 指示画笔当前位置 (**x**, **y**)

实验设计简述与核心代码:

顶层单元(VGADrawer.v)

```

`verilog
1  module VGADrawer
2  (
3  //-----
4      input          clk      ,
5      input          rst      ,
6      input  [11:0]  rgb      ,
7      input  [3:0]   dir      ,
8      input          draw     ,
9  //-----
10     output          hs      ,
11     output          vs      ,
12     output  [11:0]  vrgb
13 //-----
14 );
15 wire [15:0] paddr  ;
16 wire [11:0] pdata  ;
17 wire       we      ;
18 wire [7:0]  x      ;
19 wire [7:0]  y      ;
20 wire [15:0] vaddr  ;
21 wire [11:0] vdata  ;
22 //-----
23 DCU dcu_1
24 (
25     .clk(clk)          ,
26     .rst(rst)          ,
27     .x(x)              ,
28     .y(y)              ,
29     .vdata(vdata)      ,
30     .hs(hs)            ,

```

```

31     .vs(vs)          ,
32     .red  ( vrgb[ 3:0] ),
33     .green( vrgb[ 7:4] ),
34     .blue ( vrgb[11:8] ),
35     .vaddr(vaddr)
36 );
37 //-----
38 PCU pcu_1
39 (
40     .clk(clk)          ,
41     .rst(rst)          ,
42     .rgb(rgb)          ,
43     .dir(dir)          ,
44     .draw(draw)        ,
45     .x(x)              ,
46     .y(y)              ,
47     .we(we)            ,
48     .paddr(paddr)      ,
49     .pdata(pdata)
50 );
51 //-----
52 VRAM vram_1
53 (
54     .clk(clk)          ,
55     .rst(rst)          ,
56     .paddr(paddr)      ,
57     .pdata(pdata)      ,
58     .we(we)            ,
59     .vaddr(vaddr)      ,
60     .vdata(vdata)
61 );
62 //-----
63 endmodule

```

视频存储器 (VRAM.v) (可以在复位时自动清屏)

```

`verilog
1  module VRAM
2  (
3  //-----
4      input          clk      ,
5      input          rst      ,
6      input  [15:0]  paddr    ,
7      input  [11:0]  pdata    ,
8      input          we       ,
9      input  [15:0]  vaddr    ,
10 //-----
11      output [11:0]  vdata
12 //-----
13 );
14 reg  [15:0] a;
15 wire [11:0] d;
16 wire we_m;
17 //-----
18 assign we_m = rst ? 1'b1 : we;
19 always @ (posedge clk)
20     begin
21         if (rst)    a[15:0] <= a[15:0] + 16'd1;

```

```

22         else          a[15:0] <= paddr[15:0];
23     end
24     assign d = rst ? 12'b1111_1111_1111 : pdata[11:0];
25     //-----
26     dist_mem_gen_0 vram_core (
27         .a(a),          // input wire [15 : 0] a
28         .d(d),          // input wire [11 : 0] d
29         .dpra(vaddr),   // input wire [15 : 0] dpra
30         .clk(clk),      // input wire clk
31         .we(we_m),      // input wire we
32         .dpo(vdata)     // output wire [11 : 0] dpo
33     );
34     //-----
35 endmodule

```

绘画控制单元 (PCU.v)

```

`verilog
1  module PCU
2  (
3      //-----
4      input          clk          ,
5      input          rst          ,
6      input          [11:0]      rgb      ,
7      // rgb[ 3:0] := red  [3:0],
8      // rgb[ 7:4] := green[3:0],
9      // rgb[11:8] := blue [3:0],
10     input          [3:0]      dir      ,
11     // dir[0] := Up,      dir[1] := Left,
12     // dir[2] := Down,   dir[3] := Right,
13     input          draw          ,
14     //-----
15     output reg [7:0] x          ,
16     output reg [7:0] y          ,
17     output          we          ,
18     output reg [15:0] paddr      ,
19     output reg [11:0] pdata
20     //-----
21 );
22 //-----
23 reg          clk_slow      ;   initial clk_slow = 1'b0;
24 reg [23:0]    clk_count    ;   initial clk_count = 24'd0;
25 reg          Pixel_Clk     ;   // 50MHz像素时钟
26 //-----
27 // 产生50MHz像素时钟
28 always @ (posedge clk)      begin   Pixel_Clk  <= ~Pixel_Clk ;   end
29 //-----
30 always @ (posedge Pixel_Clk)
31     begin
32         if (clk_count == 24'd100_0000 - 24'd1)
33             begin   clk_slow <= 1'b1;   clk_count <= 24'd0;           end
34         else
35             begin   clk_slow <= 1'b0;   clk_count <= clk_count + 24'd1; end
36         end
37     //-----
38     assign we = draw & clk_slow;
39     //-----
40     always @ (posedge clk_slow or posedge rst)

```

```

41     begin
42     //-----
43         if(rst)                begin    x <= 8'd128;    y <= 8'd128;    end
44     //-----
45         else
46             begin
47                 case (dir)
48                     4'b0001:    begin    x <= x + 8'd0;    y <= y - 8'd1;    end // Up
49                     4'b0010:    begin    x <= x - 8'd1;    y <= y + 8'd0;    end // Left
50                     4'b0100:    begin    x <= x + 8'd0;    y <= y + 8'd1;    end // Down
51                     4'b1000:    begin    x <= x + 8'd1;    y <= y + 8'd0;    end // Right
52                     4'b1001:    begin    x <= x + 8'd1;    y <= y - 8'd1;    end // UpRight
53                     4'b0011:    begin    x <= x - 8'd1;    y <= y - 8'd1;    end // UpLeft
54                     4'b0110:    begin    x <= x - 8'd1;    y <= y + 8'd1;    end // DownLeft
55                     4'b1100:    begin    x <= x + 8'd1;    y <= y + 8'd1;    end //
DownRight
56                     default:    begin    x <= x + 8'd0;    y <= y + 8'd0;    end // Illegal
57                 endcase
58             end
59     //-----
60         paddr[15:0] <= {y[7:0], x[7:0]} ;
61         pdata[11:0] <= rgb[11:0] ;
62     //-----
63     end
64 //-----
65 endmodule
66

```

显示控制单元 (DCU.v)

```

`verilog
1  module DCU
2  (
3  //-----
4      input                clk        ,
5      input                rst        ,
6      input    [7:0]       x          ,
7      input    [7:0]       y          ,
8      input    [11:0]      vdata      ,
9  //-----
10     output                hs        ,
11     output                vs        ,
12     output reg [3:0]      red       ,
13     output reg [3:0]      blue      ,
14     output reg [3:0]      green     ,
15     output reg [15:0]     vaddr     ,
16 //-----
17 );
18 //-----
19 // 分辨率为800*600, 刷新频率72Hz, 像素时钟50MHz, 行时序参数定义, 行的单位是"标准时钟周期"
20 parameter Horizontal_Sync_Pulse = 120 , // 行同步宽度
21           Horizontal_Back_Porch  = 64  , // 行消隐宽度
22           Horizontal_Front_Porch = 56  , // 行前肩宽度
23           Horizontal_Active_Time  = 800 , // 行视频有效宽度
24           Horizontal_Line_Period  = 1040; // 行宽度
25 //-----
26 // 分辨率为800*600, 刷新频率72Hz, 像素时钟50MHz, 场时序参数定义, 场的单位是"行周期"
27 parameter Vertical_Sync_Pulse   = 6    , // 场同步宽度

```

```

28         Vertical_Back_Porch      = 23 ,    // 场消隐宽度
29         Vertical_Front_Porth     = 37 ,    // 场前肩宽度
30         Vertical_Active_Time      = 600 ,   // 场视频有效宽度
31         Vertical_Frame_Period     = 666 ;   // 场宽度
32 //-----
33 reg [11:0] H_Count      ;    // 行时序计数器
34 reg [11:0] V_Count      ;    // 场时序计数器
35 reg        Pixel_Clk    ;    // 50MHz像素时钟
36 wire       Active_Flag  ;    // 显示激活标志
37 wire       Cross_Sign   ;    // 十字光标区域标志
38 //-----
39 assign Active_Flag =
40     (H_Count >= (Horizontal_Sync_Pulse + Horizontal_Back_Porch
41         )) &&
42     (H_Count <= (Horizontal_Sync_Pulse + Horizontal_Back_Porch +
43         Horizontal_Active_Time
44         )) &&
45     (V_Count >= (Vertical_Sync_Pulse + Vertical_Back_Porch
46         )) &&
47     (V_Count <= (Vertical_Sync_Pulse + Vertical_Back_Porch + Vertical_Active_Time
48         )) ;
49 //-----
50 assign Show_Flag =
51     (H_Count >= (Horizontal_Sync_Pulse + Horizontal_Back_Porch
52         )) &&
53     (H_Count <= (Horizontal_Sync_Pulse + Horizontal_Back_Porch +
54         255
55         )) &&
56     (V_Count >= (Vertical_Sync_Pulse + Vertical_Back_Porch
57         )) &&
58     (V_Count <= (Vertical_Sync_Pulse + Vertical_Back_Porch +
59         255
60         )) ;
61 //-----
62 assign Cross_Sign = (
63     (H_Count <= (Horizontal_Sync_Pulse + Horizontal_Back_Porch + x + 5
64         )) &&
65     (H_Count >= (Horizontal_Sync_Pulse + Horizontal_Back_Porch + x - 5
66         )) &&
67     (V_Count == (Vertical_Sync_Pulse + Vertical_Back_Porch + y
68         ))
69     ) ||
70     (
71     (H_Count == (Horizontal_Sync_Pulse + Horizontal_Back_Porch + x
72         )) &&
73     (V_Count >= (Vertical_Sync_Pulse + Vertical_Back_Porch + y - 7
74         )) &&
75     (V_Count <= (Vertical_Sync_Pulse + Vertical_Back_Porch + y + 7
76         ))
77     );
78 //-----
79 // 产生50MHz像素时钟
80 always @ (posedge clk) begin Pixel_Clk <= ~Pixel_Clk ; end
81 //-----
82 // 产生行时序
83 always @ (posedge Pixel_Clk or posedge rst)
84 begin
85     if (rst)
86         H_Count <= 12'd0 ;
87     else if (H_Count == Horizontal_Line_Period - 1'b1)
88         H_Count <= 12'd0 ;
89     else
90         H_Count <= H_Count + 12'd1 ;
91 end
92 //-----
93 // 产生场时序
94 always @ (posedge Pixel_Clk or posedge rst)

```

```

74 begin
75     if (rst)                                V_Count <= 12'd0 ;
76     else if (V_Count == Vertical_Frame_Period - 1'b1) V_Count <= 12'd0 ;
77     else if (H_Count == Horizontal_Line_Period - 1'b1) V_Count <= V_Count + 1'b1 ;
78     else                                V_Count <= V_Count ;
79 end
80 //-----
81 assign hs = (H_Count < Horizontal_Sync_Pulse) ? 1'b0 : 1'b1 ;
82 assign vs = (V_Count < Vertical_Sync_Pulse ) ? 1'b0 : 1'b1 ;
83 //-----
84 // 产生视频显示
85 always @ (posedge Pixel_Clk or posedge rst)
86 begin
87     if (rst)                                vaddr <= 16'd0;
88     else if (Active_Flag)
89         begin
90             if (Show_Flag)
91                 begin
92                     if (Cross_Sign)
93                         begin
94                             red    <= 4'b0000 ;
95                             green  <= 4'b1111 ;
96                             blue   <= 4'b0000 ;
97                             vaddr <= vaddr + 1'b1 ;
98                         end
99                     else
100                         begin
101                             red    <= vdata[ 3:0] ;
102                             green  <= vdata[ 7:4] ;
103                             blue   <= vdata[11:8] ;
104                             vaddr <= vaddr + 1'b1 ;
105                         end
106                     end
107                 end
108             begin
109                 red    <= 4'b0000 ;
110                 green  <= 4'b0000 ;
111                 blue   <= 4'b0000 ;
112                 vaddr <= vaddr ;
113             end
114         end
115     else
116         begin
117             red    <= 4'b0000 ;
118             green  <= 4'b0000 ;
119             blue   <= 4'b0000 ;
120             vaddr <= vaddr ;
121         end
122 end
123 //-----
124 endmodule
125

```

实验结果：

现场烧录检查：已通过

实现资源消耗与性能统计：

Utilization

Post-Synthesis | Post-Implementation

Graph | Table

Resource	Utilization	Available	Utilization %
LUT	26164	63400	41.27
LUTRAM	16384	19000	86.23
FF	1010	126800	0.80
IO	33	210	15.71
BUFG	3	32	9.38

Summary

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power: 386.907 W (Junction temp exceeded!)

Design Power Budget: Not Specified

Power Budget Margin: N/A

Junction Temperature: 125.0°C

Thermal Margin: -1705.4°C (-373.2 W)

Effective θ_{JA} : 4.6°C/W

Power supplied to off-chip devices: 0 W

Confidence level: Low

[Launch Power Constraint Advisor](#) to find and fix invalid switching activity

On-Chip Power

99%

Dynamic: 386.118 W (99%)

72%

28%

0%

1%

Signals: 276.483 W (72%)

Logic: 108.294 W (28%)

I/O: 1.340 W (0%)

Device Static: 0.797 W (1%)

片外器件供电

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	inf	Worst Hold Slack (WHS):	inf	Worst Pulse Width Slack (WPWS):	NA
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	NA
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	NA
Total Number of Endpoints:	128159	Total Number of Endpoints:	128159	Total Number of Endpoints:	NA
There are no user specified timing constraints.					

仿真测试结果：

因为本实验需要大量降频和视频控制操作(刷新)，不便仿真观察结果，故未仿真，现场烧录检查通过，没有任何问题。

实验总结与感想：

1. 通过实验了解了存储器与显示控制器的设计实现，了解了存储器与显示控制器的简单应用。
2. 复习了Verilog语法，提高了编程实践能力。