数字逻辑课程综合实验报告

学　　号 xxxxxxx

姓　　名 xxxxxx

专　　业 工科试验班（电子类）

授课老师 xxxxxx

一、实验内容

制作 MP3 播放器，用到的外围部件有 MP3 模块、VGA 显示屏模块，在 Xilinx 开发板进行调试运行。

1. VGA 工作原理

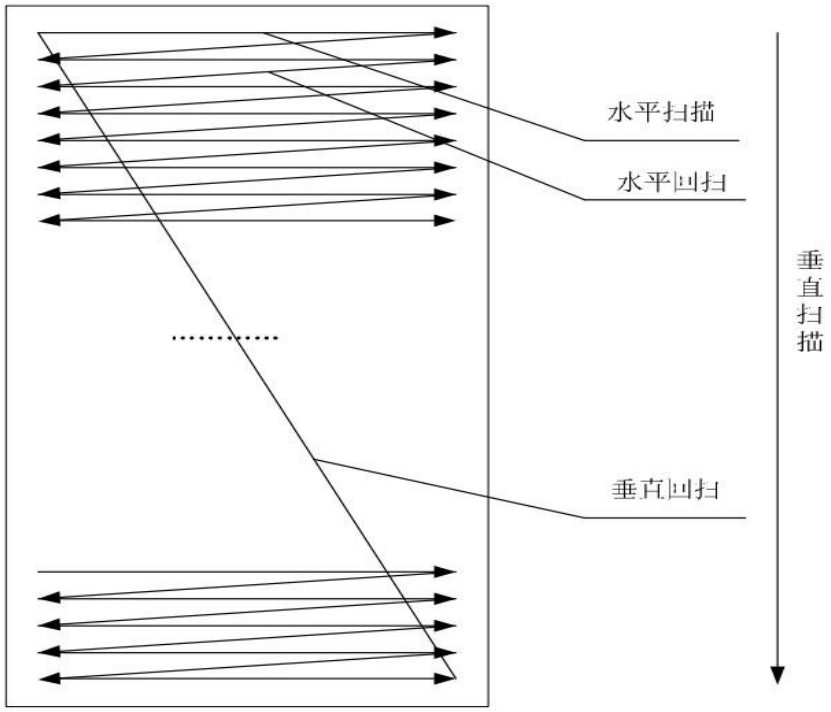
（1） 显示器扫描方式——逐行扫描

① 扫描从屏幕左上角开始，从左向右逐点扫描。

② 每扫描完一行，电子束回到下一行的起始位置，期间 CRT 对电子束进行消隐。

③ 每行结束时，用行同步信号进行同步，当扫描完全部的行，形成一帧，用场同步信号进行同步，回到屏幕左上角，同时进行场消隐，开始下一帧。

④ 下图为扫描方式示意图：



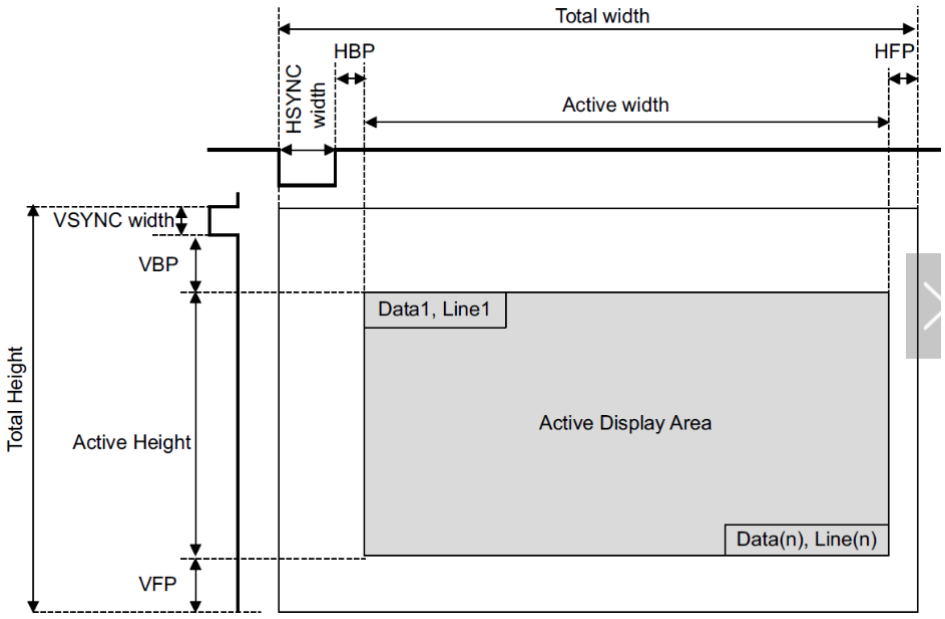
（2） VGA 扫描时序控制

① 行时序和场时序都需要同步脉冲（Sync）、显示后沿、显示时序段和显示前沿四部分。VGA 工业标准显示模式要求：行同步，场同步都为负极性，即同步脉冲要求是负脉冲。

② 由 VGA 的行时序可知：没一行都有一个负极性行同步脉冲（Sync a），是数据行的结束标志，同时也是下一行的开始标志。在同步脉冲之后为显示后沿，在显示时序段显示器为亮的过程，RGB 数据驱动一行上的每一个像素点，从而显示一行。在一行的最后为显示前沿。在显示时间段之外没有图像投射到屏幕是插入消隐信号。同步脉冲、显示后沿和显示前沿都是在行消隐间隔内，当消隐有效时，RGB 信号无效，屏幕不显示数据。

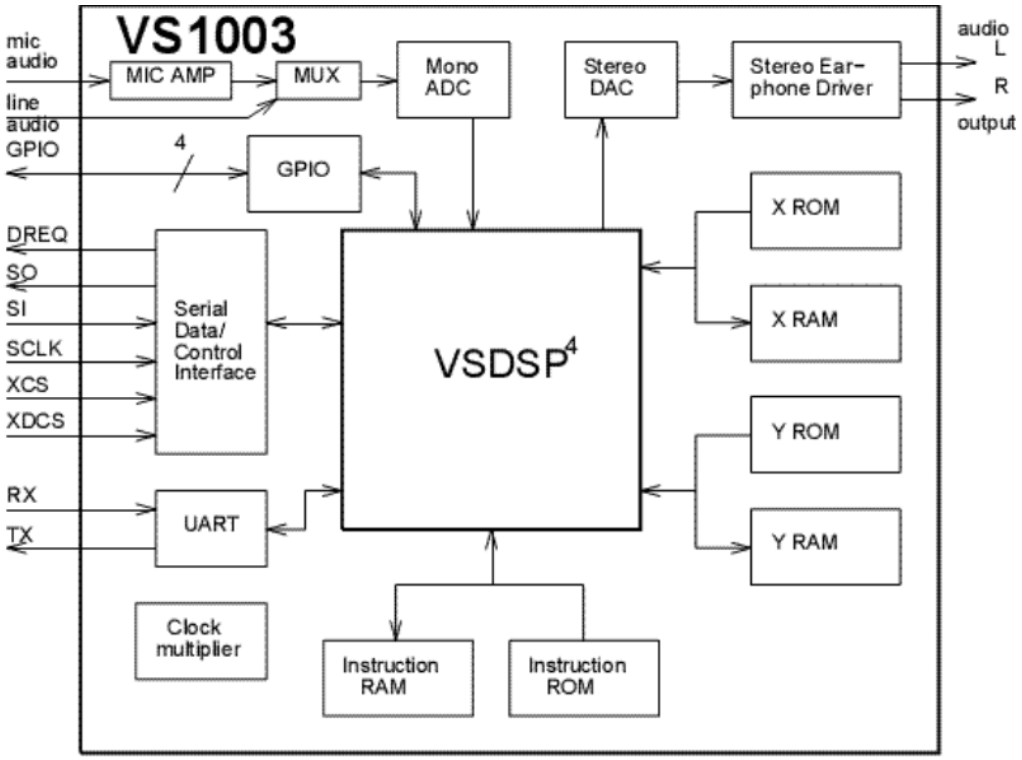
③ VGA 的场时序与行时序基本一样，每一帧的负极性脉冲是一帧的结束标志，同时也是下一帧的开始标志。而显示数据是一帧的所有行数据。

④ 下图为 VGA 显示显示屏部件的构成



2. MP3 通信原理

(1) VS1003 模块内部构造



(2) **MP3 模块引脚说明**

序号	名称	说明
1	GND	地
2	5V	5V 供电口，只可以供电
3	3.3V	3.3V 供电口，当使用 5V 供电的时候，这里可以输出 3.3V 电压给外部使用
4	XCS	片选输入(低有效)
5	XDCS	数据片选/字节同步
6	SCK	SPI 总线时钟线
7	SI	SPI 总线数据输入线
8	SO	SPI 总线数据输出线
9	DREQ	数据请求
10	RST	复位引脚(硬复位，低电平有效)

(3) **SPI 数据传输协议：**

① SPI(Serial Peripheral Interface--串行外设接口)总线系统是一种同步串行外设接口，它可以使 MCU 与各种外围设备以串行方式进行通信以交换信息。SPI 总线可直接与各个厂家生产的多种标准外围器件相连，包括 FLASHRAM、网络控制器、LCD 显示驱动器、A/D 转换器和 MCU 等。该接口一般使用 4 条线：串行时钟线（SCLK）、主机输入/从机输出数据线 MISO、主机输出/从机输入数据线 MOSI 和低电平有效的从机选择线 NSS。

② **接口信号**

MOSI – 主器件数据输出，从器件数据输入

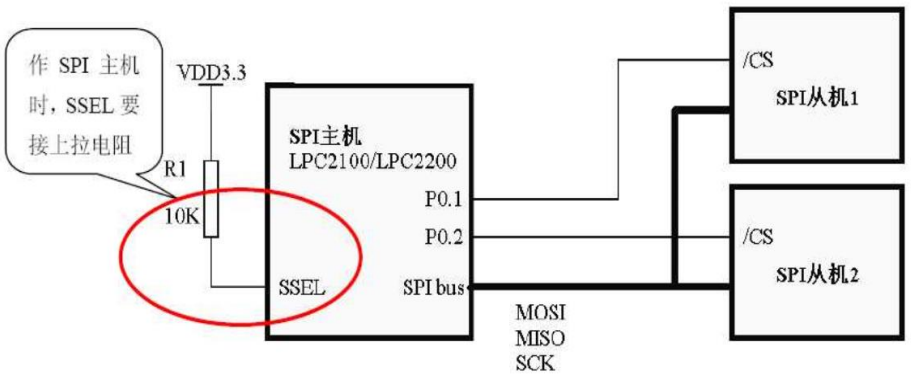
MISO – 主器件数据输入，从器件数据输出

SCLK – 时钟信号，由主器件产生,最大为 fPCLK/2，从模式频率最大为 fCPU/2

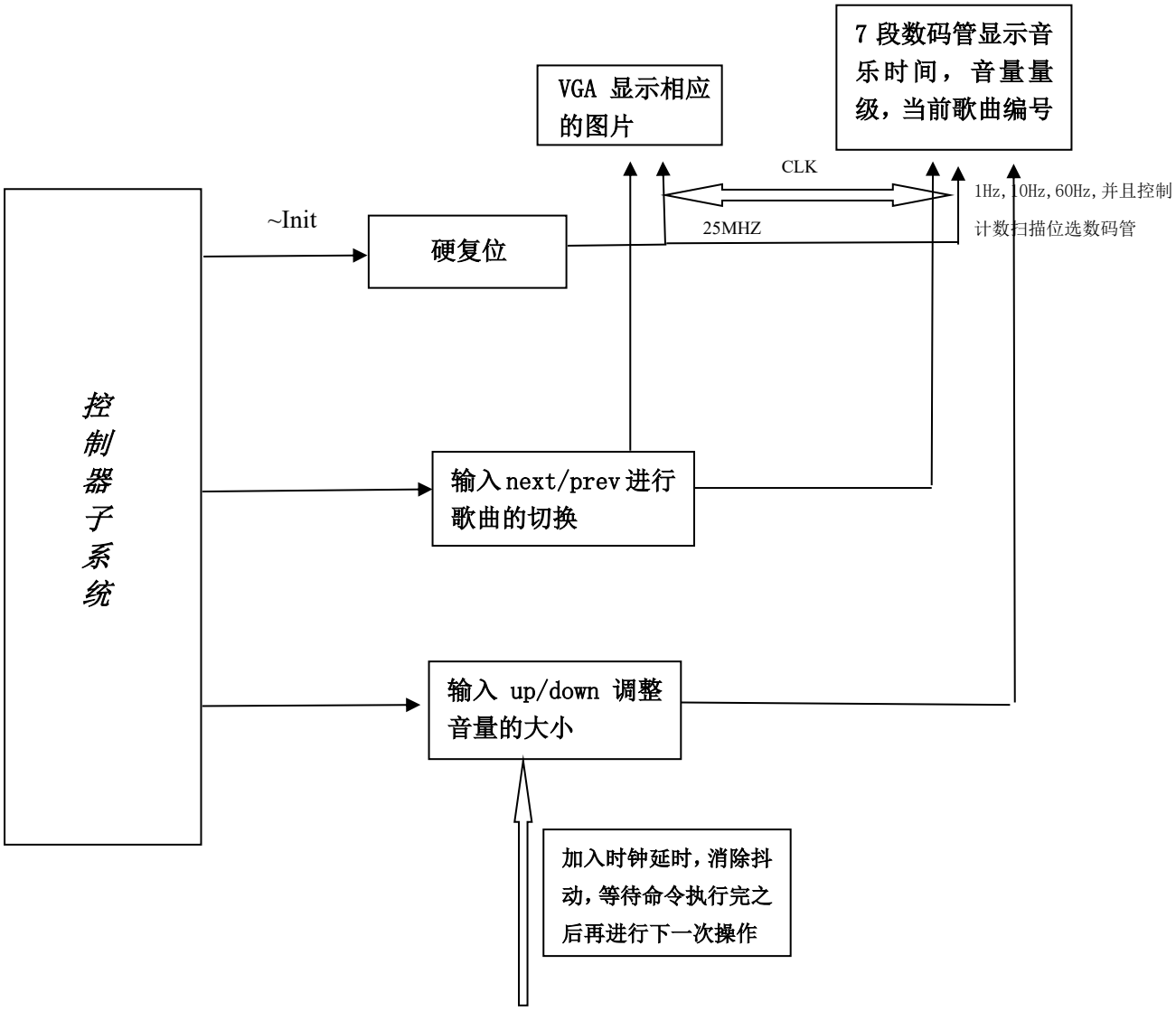
NSS – 从器件使能信号，由主器件控制,有的 IC 会标注为 CS(Chip select)

③ 点对点的通信中，SPI 接口不需要进行寻址操作，且为全双工通信，显得简单高效。在多个从器件的系统中，每个从器件需要独立的使能信号，硬件上比 I2C 系统 要稍微复杂一些。SPI 接口在内部硬件实际上是两个简单的移位寄存器，传输的数据为 8 位,在主器件产生的从器件使能信号和移位脉冲下,按位传输，高位在前，低位在后。如下图所示，在 SCLK 的上升沿上数据改变，同时一位数据被存入移位寄存器。

④ **下图为 SPI 通信原理图：**

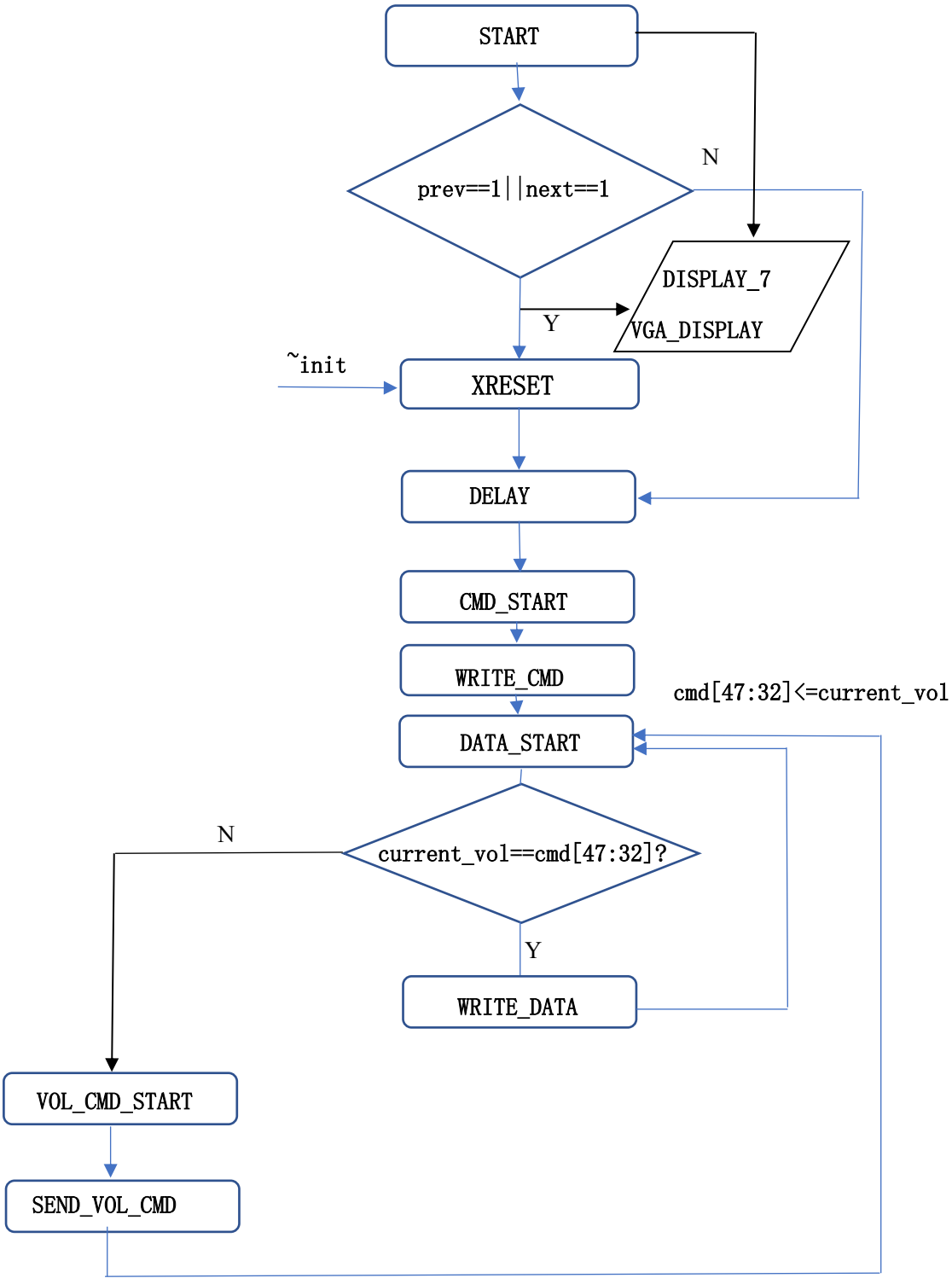


3. 数字系统总框图


7 段数码管显示音乐时间，音量量级，当前歌曲编号
VGA 显示相应的图片
CLK
1Hz, 10Hz, 60Hz, 并且控制计数扫描位选数码管
25MHZ
~Init
硬复位
控制器子系统
输入 next/prev 进行歌曲的切换
输入 up/down 调整音量的大小
加入时钟延时，消除抖动，等待命令执行完之后再进行下一次操作

4. 系统控制器设计

ASM 流程图



计数器法状态编码：

NAME	CODE
START	000
DELAY	001
CMD_START	010
WRITE_CMD	011
DATA_START	100
WRITE_DATA	101
VOL_CMD_START	110
SEND_VOL_CMD	111

状态转移表：

PS	NS	CONDITION
START 000	DELAY 001	NO
DELAY 001	CMD_START 010	NO
CMD_START 010	WRITE_CMD 011	NO
WRITE_CMD 011	DATA_START 100	NO
DATA_START 100	WRITE_DATA 101	Current_vol==cmd[47:32]
DATA_START 100	VOL_CMD_START 110	Current_vol!=cmd[47:32]
SEND_VOL_CMD 110	DATA_START 111	NO
WRITE_DATA 101	DATA_START 100	NO

5. 子系统模块建模

(1) MP3_TOP 模块

```
module MP3_TOP(
input CLK,//系统时钟
input DREQ,//数据请求
output reg XRSET,//硬件复位
output reg XCS,//低电平有效片选输入
output reg XDCS,//数据片选/字节同步
output reg SI,//串行数据输入
output reg SCLK,//SPI 时钟
input init,//初始化
input up_vol,//音量加
input down_vol,//音量减
input prev,//上一曲
input next,//下一曲
output [7:0]seg_sel,//七段数码管段选
output  [6:0]seg_data,//七段数码管位选
output dot,//小数点
```

```
output  [3:0]R,//vga-R
output  [3:0]G,//vga-G
output   [3:0]B,//vga-B
output  H_Sync,//行扫描信号
output  V_Sync//列扫描信号
    );

    parameter  CMD_START=0;//开始写指令
    parameter  WRITE_CMD=1;//将一条指令全部写入
    parameter  DATA_START=2;//开始写数据
    parameter  WRITE_DATA=3;//将一条数据全部写入
    parameter  DELAY=4;//延时
    parameter VOL_CMD_START=5;//音量发送命令开始
    parameter SEND_VOL_CMD=6;//将音量发送命令全部写入 MP3 芯片
    reg [95:0]cmd={32'h02000804,32'h020B0000,32'hffffffff};//32'h02000804 软复位
    integer status=CMD_START;
    integer cnt=0;//位计数
    integer cmd_cnt=0;//命令计数

    wire CLK_1M;//分频 1MHz
    Divider #(.N(100))CLKDIV(CLK,CLK_1M);


    assign dot=1;
    wire [15:0]adjusted_vol;
    reg [31:0]volcmd;
```

(2) 调整音量模块

```
    Adjust_vol vol(
    //input
    .clk(CLK_1M),
    .up(up_vol),
    .down(down_vol),
     //output
    .vol(adjusted_vol)//调整之后的音量
    );


    wire [3:0]current;
    reg  [2:0]pre=0;
```

(3) 切换音乐模块

```
Switch_music
sw_msc(
.clk(CLK_1M),
.prev(prev),
.next(next),
.current(current)
);
```

(4) VGA 显示图片模块

```
VGA_DISPLAY
display_vga(
.clk(CLK),
.rst(init),
.switch(current),
.R(R),
.G(G),
.B(B),
.H_Sync(H_Sync),
.V_Sync(V_Sync)
 );
```

(5) 七段数码管显示模块

```
 Display7
 display7(
 .rst(init),
 .clk(CLK_1M),
 .idata1(vol_class),
 .idata2(current),
 .seg_data(seg_data),
 .seg_sel(seg_sel)
 );
```

(6) ROM 模块

```
reg [14:0]addr;
wire [15:0] Data;
reg [15:0] _Data;
blk_mem_gen_0 your_instance_name(
.clka(CLK),
.ena(1),
.addra({current,addr}),
.douta(Data)
```

);

(MP3 工作原理)

```
always @(posedge CLK_1M) begin
pre<=current;
if(~init || pre!=current) begin
    XCS<=1;
    XDCS<=1;
    XRSET<=0;//硬件复位
    cmd_cnt<=0;
    status<=DELAY;
    SCLK<=0;
    cnt<=0;
end
else begin
    case(status)
    /*开始发送命令*/
    CMD_START:begin
        SCLK<=0;
        if(cmd_cnt>=3)
            status<=DATA_START;
        else if(DREQ) begin
            XCS<=0;
            status<=WRITE_CMD;
            SI<=cmd[95];
            cmd<={cmd[94:0],cmd[95]};
            cnt<=1;
        end
    end
    /*将命令全部写入 MP3 芯片*/
     WRITE_CMD:begin
        if(DREQ) begin
            if(SCLK) begin
                if(cnt>=32)begin
                    XCS<=1;
                     cnt<=0;
                    cmd_cnt<=cmd_cnt+1;
                    status<=CMD_START;
                end
                else begin
                   SI<=cmd[95];
                   cmd<={cmd[94:0],cmd[95]};
                   cnt<=cnt+1;
```

```
            end
        end
        SCLK<=~SCLK;
    end
end
/*开始发送数据*/
DATA_START:begin
if(adjusted_vol[15:0]!=cmd[47:32])begin//音量变了
    cnt<=0;
    volcmd<={16'h020B,adjusted_vol};
    status<=VOL_CMD_START;
end
else if(DREQ) begin
        XDCS<=0;
        SCLK<=0;
        SI<=Data[15];
        _Data<={Data[14:0],Data[15]};
        cnt<=1;
        status<=WRITE_DATA;
    end
   cmd[47:32]<=adjusted_vol;
end
/*将数据全部写入芯片*/
WRITE_DATA:begin
    if(SCLK)begin
        if(cnt>=16)begin
            XDCS<=1;
            addr<=addr+1;
            status<=DATA_START;
        end
        else begin
          cnt<=cnt+1;
          _Data<={_Data[14:0],_Data[15]};
           SI<=_Data[15];
        end
    end
    SCLK<=~SCLK;
end
/*在正式的发送命令前首先进行时钟准备*/
DELAY:begin
           if(cnt<1000)
               cnt<=cnt+1;
           else begin
               cnt<=0;
```

```
                    status<=CMD_START;
                    XRSET<=1;
                end
            end
        /*开始发送调整音量的命令*/
        VOL_CMD_START:begin
        if(DREQ) begin
                XCS<=0;
                status<=SEND_VOL_CMD;
                SI<=volcmd[31];
                volcmd<={volcmd[30:0],volcmd[31]};
                cnt<=1;
            end
        end
        /*将调整音量的命令全部写入 MP3 芯片*/
        SEND_VOL_CMD:begin
          if(DREQ) begin
                 if(SCLK) begin
                    if(cnt<32)begin
                        SI<=volcmd[31];
                       volcmd<={volcmd[30:0],volcmd[31]};
                       cnt<=cnt+1;
                    end
                    else begin
                        XCS<=1;
                         cnt<=0;
                         status<=DATA_START;

                    end
                end
                SCLK<=~SCLK;
            end
        end
        default:;
        endcase
    end
end
endmodule
```

（VGA 显示 TOP 模块）

```
module VGA_DISPLAY(
input clk,rst,
input [3:0]switch,
```

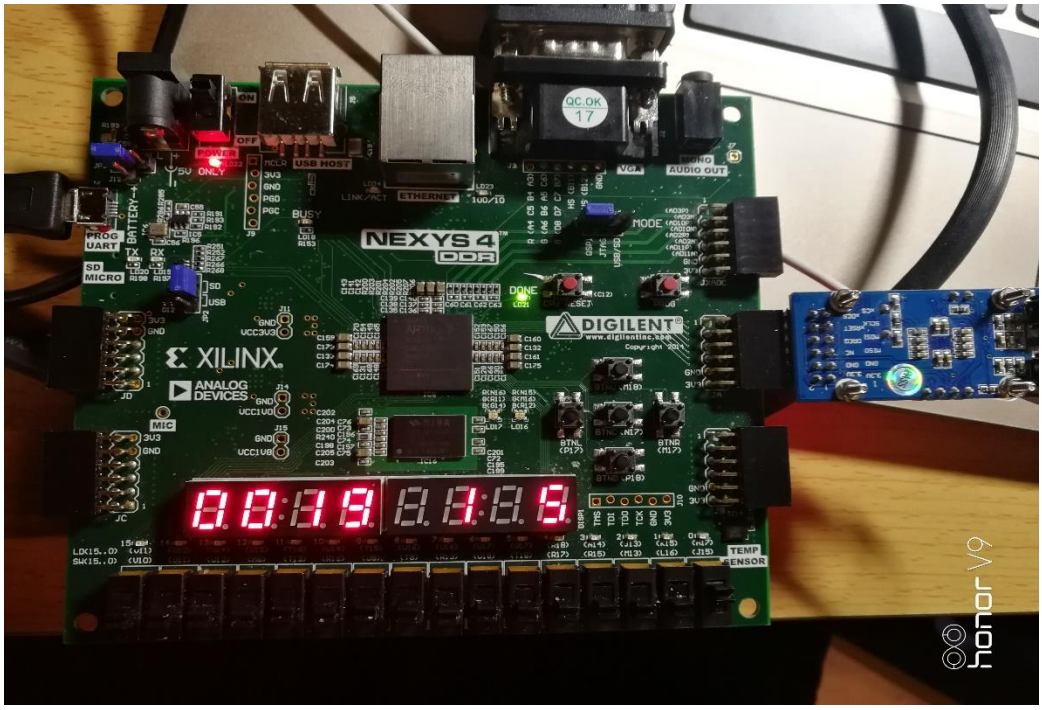
```
output [3:0]R,
output [3:0]G,
output [3:0]B,
output H_Sync,V_Sync);
wire [9:0]x_cnt,y_cnt;
wire clk_25M;
 //分频
 Divider #(.N(4))CLKDIV(clk,clk_25M);
  //生成 VGA 扫描信号
 VGA_Sync
vga_sync(.clk(clk_25M),.rst(rst),.x_cnt(x_cnt),.y_cnt(y_cnt),.H_Sync(H_Sync),.V
_Sync(V_Sync));
  //显示图像
  VGA__RGB
vga_rgb(.clk(clk_25M),.x_cnt(x_cnt),.y_cnt(y_cnt),.R(R),.G(G),.B(B),.switch(swi
tch));
endmodule
```

6. 测试模块建模

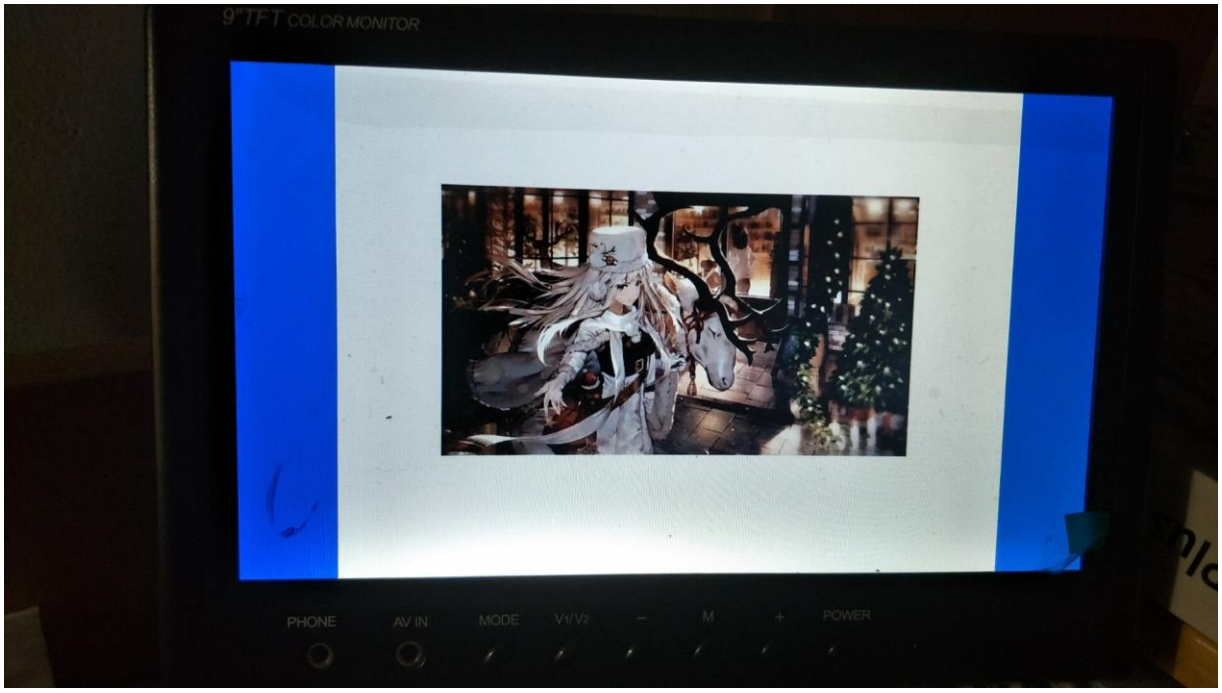
该部分测试主要是进行从 vs1003b 读数再解码，故无法进行数据是否读到的测试。

7. 实验结果

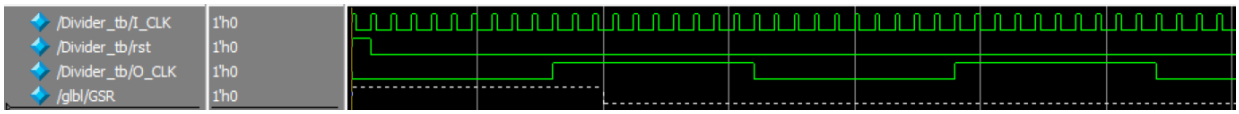
（1）音量调节及歌曲播放顺序界面：（如下图）



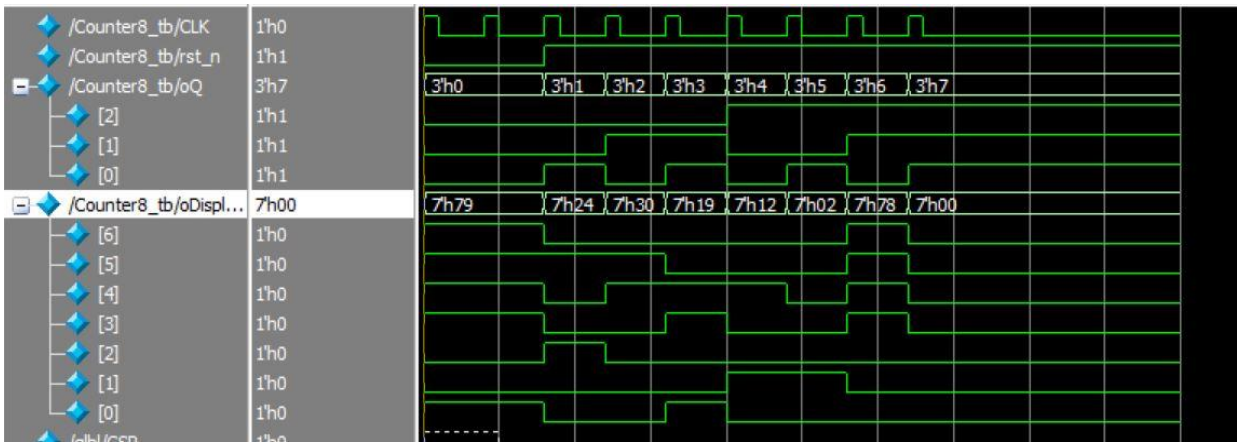
（2） VGA 显示界面：



（3）分频：



（4）七段数码管：



8. 结论

该 MP3 播放器是基于 VS1003B 芯片解码的，它支持多种比特流（MP3，WAV，mid 等格式），在此由于在做 SD 卡的时候出现了困难没能解决，所以只能用 vivado 自带的 IP 核中的 ROM 进行歌曲的存放,存放了几首 mid 格式的歌曲，在播放音乐的同时还可以对音量进行调节，还能左右切换歌曲，并且同时进行音量量级的显示和歌曲顺序的显示，切换歌曲的同时还可以对 vga 显示屏进行控制让其显示相应歌曲对应歌手及相关图片，从而实现音乐的播放。

9. 心得体会及建议

心得体会：通过本次大作业，我完全体会到了完成一件相对较难而且从未接触过的任务是一种怎样的体验，比如该如何去开展一项工程、如何对自己的作品更进行创新性的构思和制作、如何组合各个模块让其发挥出自己内心想要完成的功能、如何充分调动各方面的资源进行参考和学习、如何完全搞懂一个部件内在的工作原理（比如蓝牙的通信协议、MP3 的通信协议、vga 显示屏扫描显示原理、SD 卡读写数据以及初始化响应的原理）。之不得不说是一件艰难的任务，首先在思想上就有一定的抵触性：本来就没学过，还要在这么短的时间做出一件作品那得多难？常人思维就是这样，所以本身就增加了难度，对我而言亦是如此。

但是随着网上资料的阅读量逐渐增大，还有参考例程习得的知识，对自己所做的内容有了一个大体的认识，之后又去细化了各方面细节，充分理解了 SPI 通信的原理及 MP3 内部构造，从根本上消除了之前认为很难的想法，进而对其产生了兴趣，认为自己一定可以搞定。通过不断的研究，尤其是对其时序的控制，几种模块之间不同的时序带来的混乱和繁杂感也慢慢有了条理，并且掌握了方法如何去控制。一次一次 debug，一次一次试错，每一次的小进步都是令人喜悦的——我正在掌握它并且完成它。

经过几周的努力，一件作品也算终于成功了。每天苦思冥想地解决问题，终于得到了回应，这是一种超爽的感觉。也感觉自己正在一点点进步，去掌握各个变量之间的逻辑关系，思维的逻辑性也渐渐建立起来了。

建议：希望老师们多多给我们自由发展的空间，让我们自己去完成一件作品，从中自己探索、培养创新思维。我们更应该将平时学习的东西在实践中去整合运用。