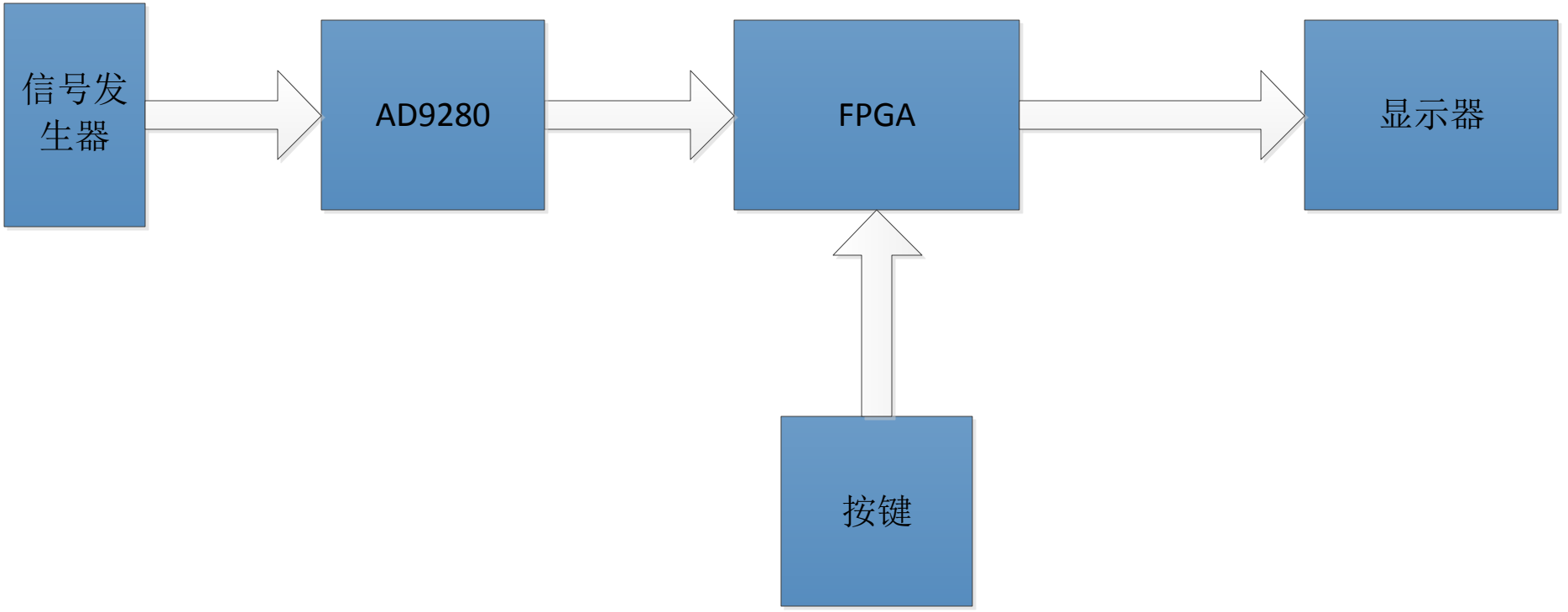
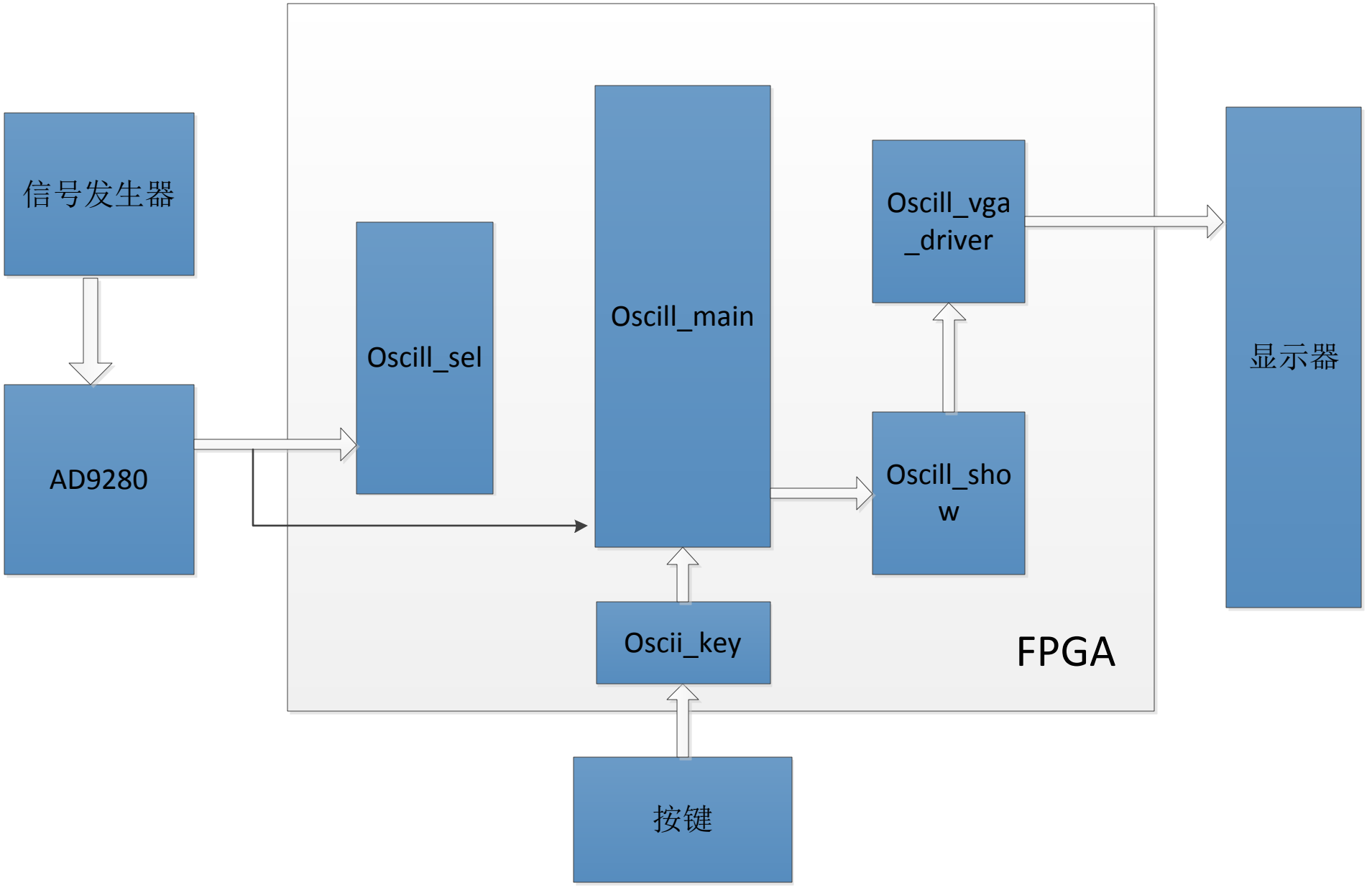
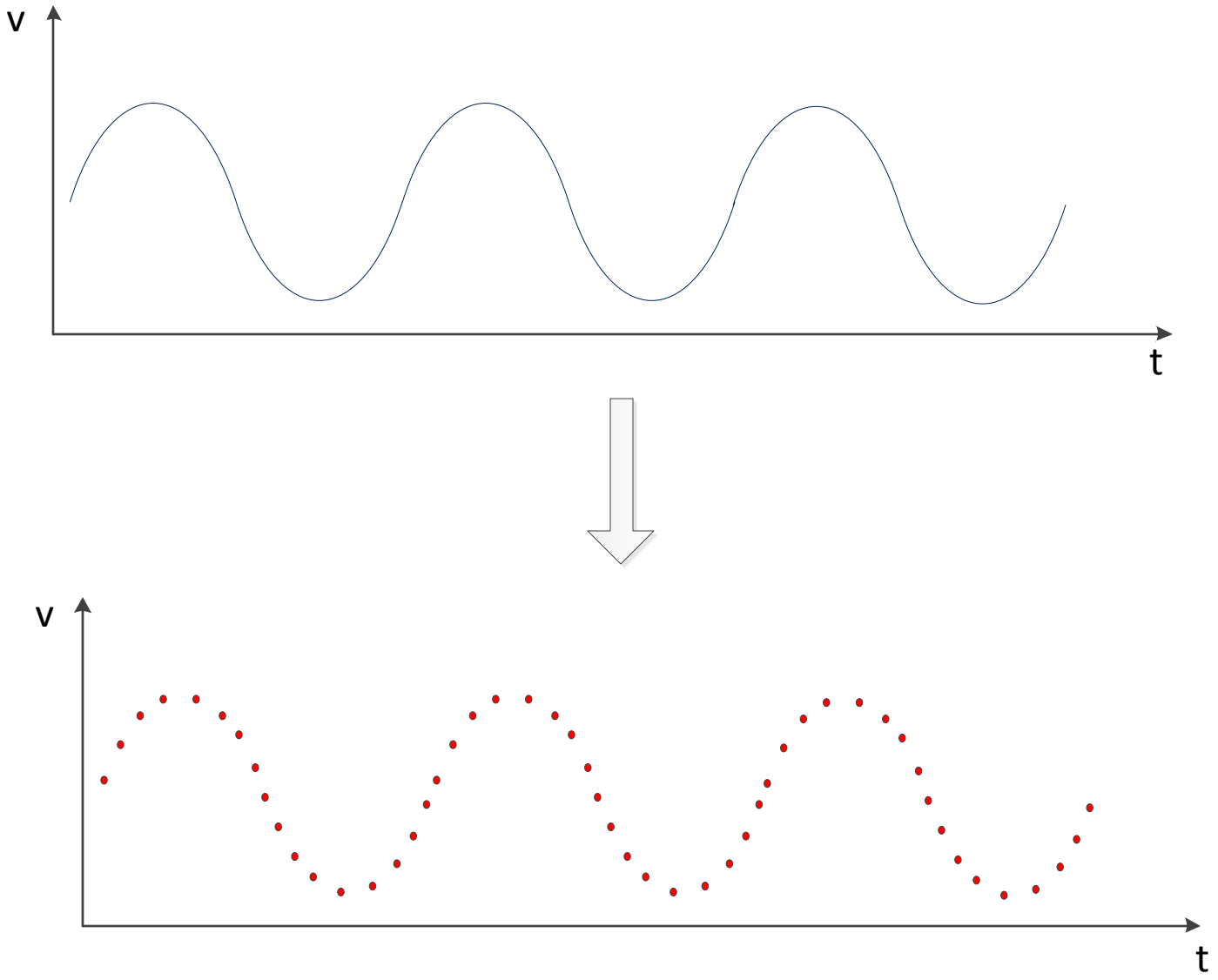
简易示波器项目

本项目功能：将信号发生器输出的波形，通过模数转换器（AD9280）转换为数字信号之后，输入到FPGA进行信号的捕获、存储等操作，然后通过VGA接口在显示器上显示出相应的波形。



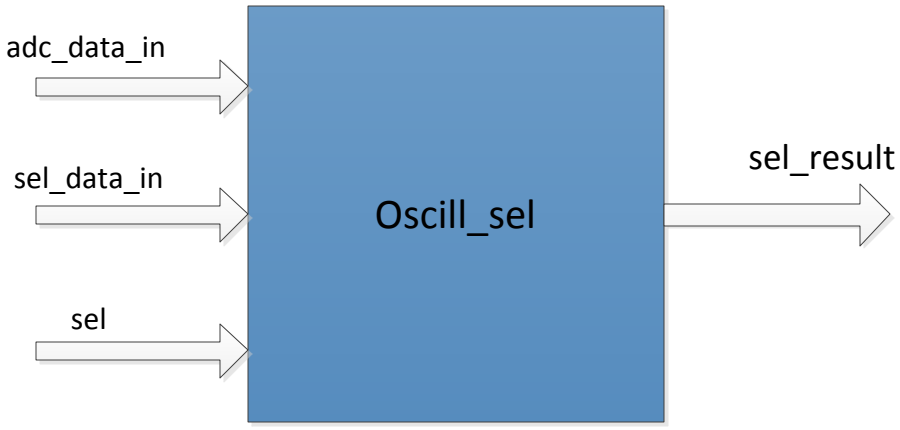
项目模块划分


信号发生器
AD9280
Oscill_sel
Oscill_main
Oscill_vga
_driver
Oscill_sho
w
显示器
Oscii_key
FPGA
按键



由于本工程用到的ADC模数转换器AD9280是8位输出的，所以这里每个采样点的位宽为8bit，最大值为256

Home



模块功能：检测输入信号（由ADC9280将模拟信号转换得到的数字信号）把它与触发的阈值作比较，判断信号上升沿或下降沿并输出一个有效信号，大于阈值为上升沿否则为下降沿

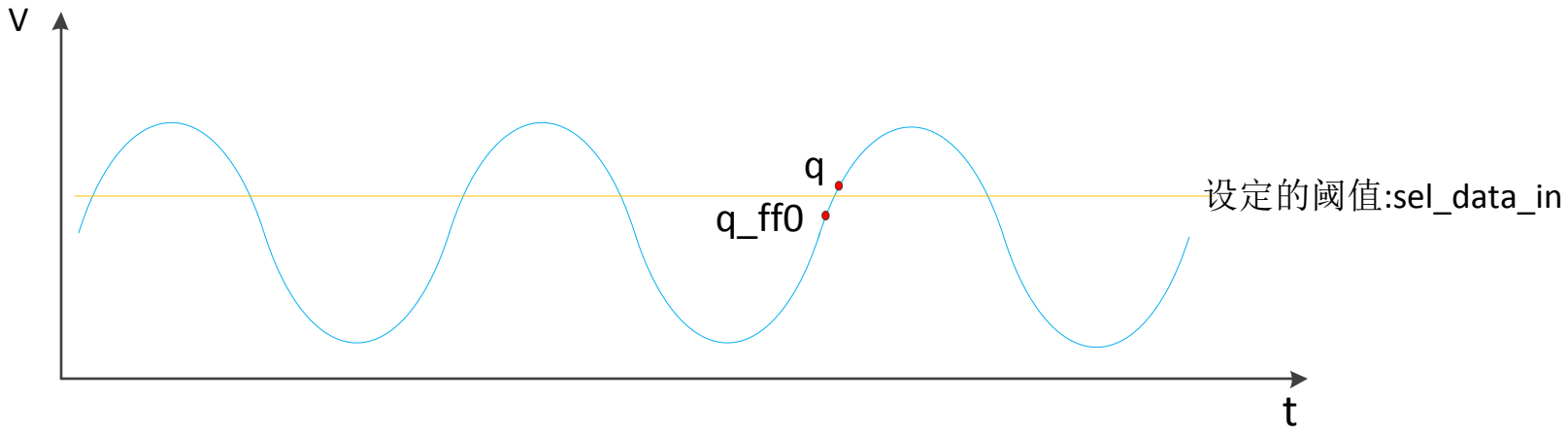
adc_data_in:要检测的输入信号

sel_data_in:触发阈值

Sel触发方式选择:0表示下降沿触发，1表示上升沿触发

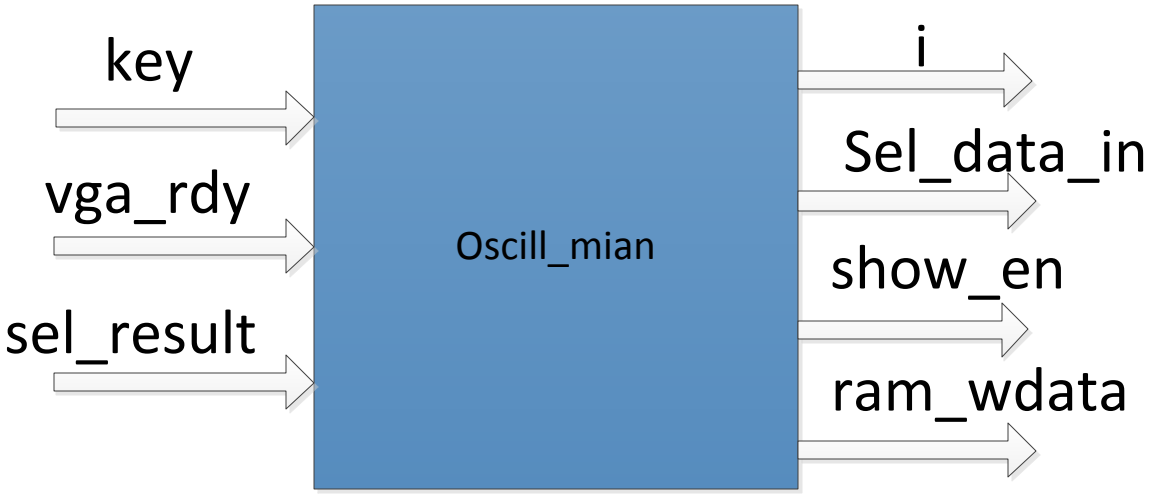
sel_result:上升沿或下降沿触发的有效指示信号

Home
Next

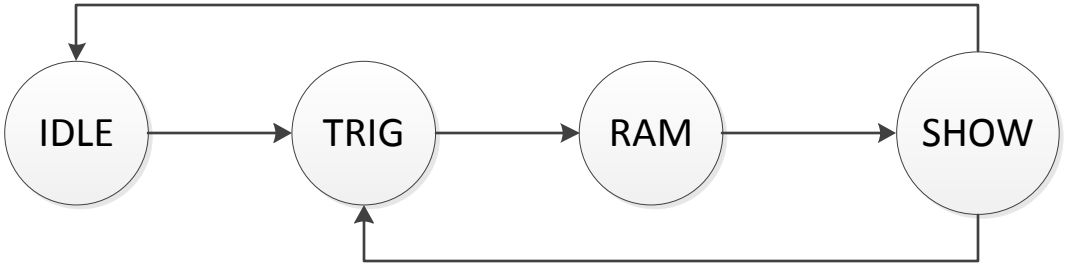



```
always @(posedge clk or negedge rst_n)begin
    if(!rst_n)begin
        sel_result <= 0;
    end
     else if(sel==1 && ( q >= sel_data_in )&&( q_ff0 < sel_data_in ) )begin  //ADC输入值大于设定值  就是上升沿
        sel_result <= 1;
     end
     else if(sel==0 && ( q <= sel_data_in )&&( q_ff0 > sel_data_in ))begin  //ADC输入值小于设定值  就是下升沿
        sel_result <= 1;
     end
     else
        sel_result <= 0;
end
```

Previous



功能：主要划分了IDLE（复位）、TRIG（等待触发状态）、RAM（读取FIFO数据）、SHOW（刷新屏幕）四种状态，几种状态之间切换，实现了对信号的检测触发，并且把数据保存起来，然后把数据送到屏幕上显示。



key:控制状态间的跳转和显示信号波形的大小

vga_rdy：显示屏刷新完一帧图像指示信号

sel_data_in:触发阈值

ram_data:RAM中读取的数据

sel_result：有效触发指示信号

i：调节波形放大、缩小系数

show_en:show状态使能信号

动图

Home

功能：根据我们要的实际显示效果给不同的输出划分不同的显示区域（VGA地址），并且输出数据给下游模块（VGA驱动）进行显示

ram_data:读取ram中的数据

vga_x:在显示屏上显示信号波形区域的横轴坐标

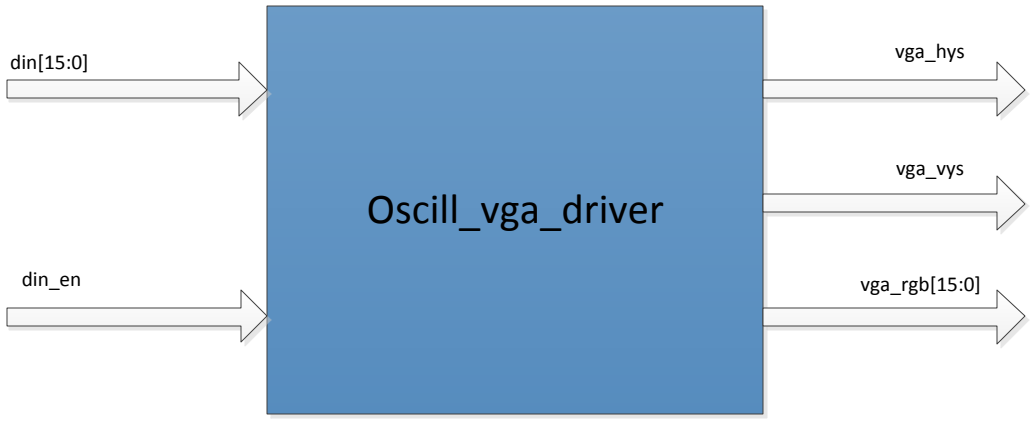
vga_y:在显示屏上显示信号波形区域的横轴坐标

vga_data:送给显示屏显示的数据

ram_raddr:读ram中的那个地址

Home

Next



功能：根据VGA时序驱动显示器，把不同的数据显示其相对应的颜色在不同的区域上。

din:要显示的数据

din_en:使能显示

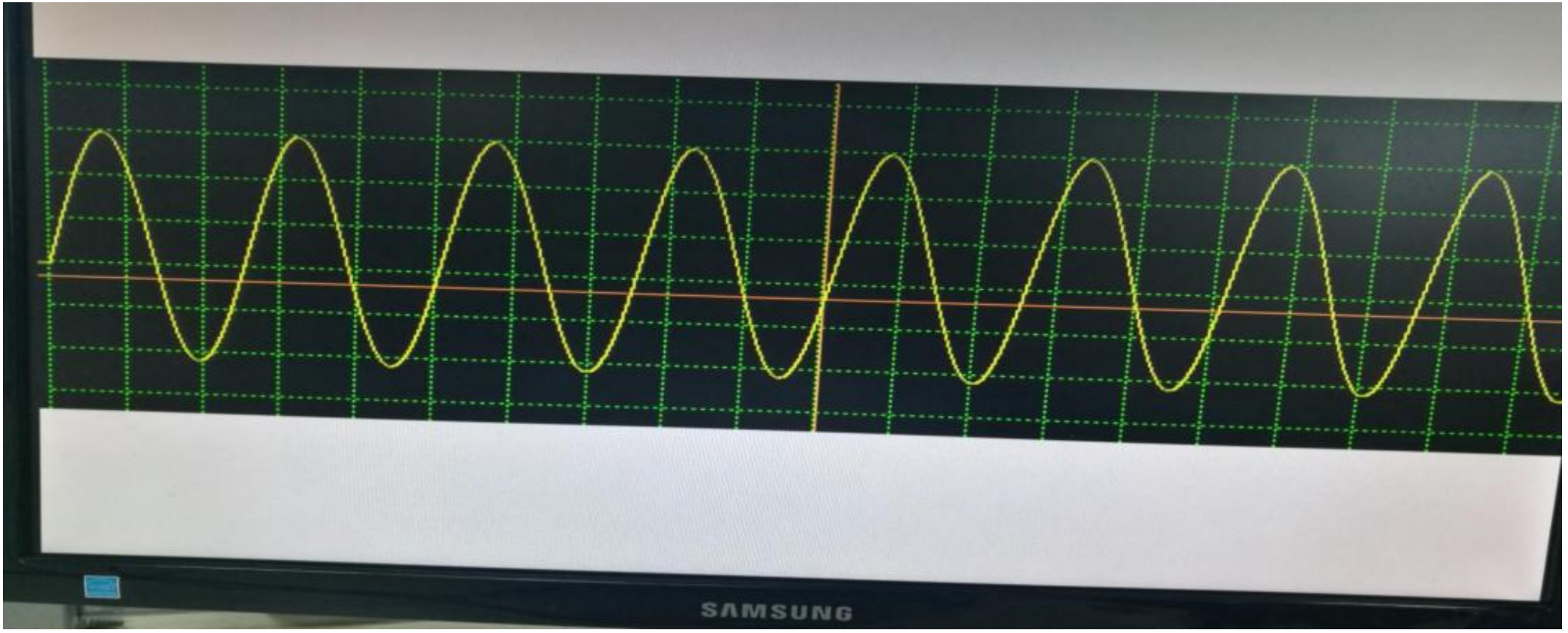
vga_hys:VGA行驱动信号

vga_vys:VGA场驱动信号

vga_rgb:VGA驱动输入信号

Home

Next



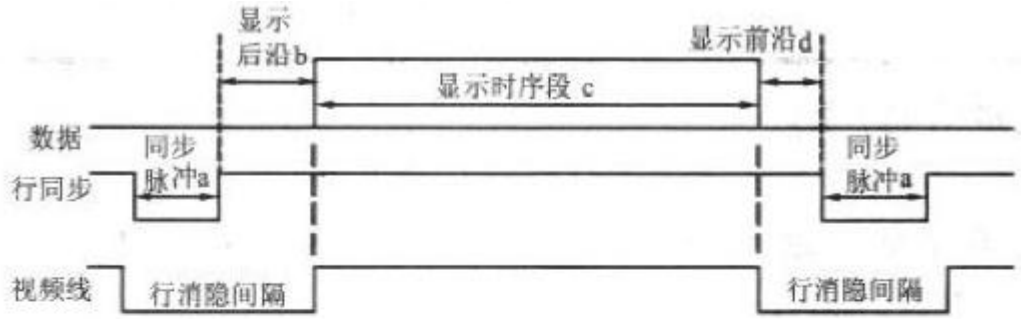
previous

表 1 水平时序

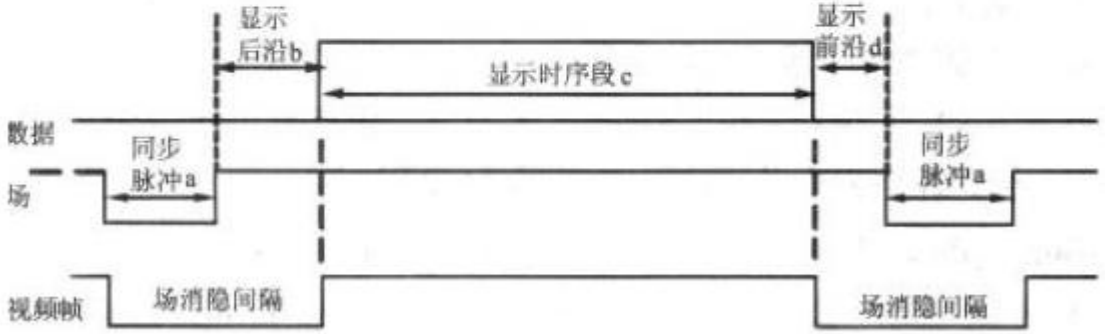
分辨率	刷新速率	像素频率	同步脉冲	后沿	有效时间	前沿	帧长
640/480	60	25	96	45	646	13	800

表 2 垂直时序

分辨率	刷新速率	行宽	同步脉冲	后沿	有效时间	前沿	帧长
640/480	60	31	2	30	484	9	525



VGA 的行时序



VGA 的场时序

Home

previous

Next

怎样根据RAM中的数据，输出到显示器上？

根据Y轴计算出一个对应值(cnt0)，再用这个对应值(cnt0)和RAM中的数据做对比，有效就输出打点(红色数据)，横轴X是取RAM中地址为X的数据

y	cnt0=8-y																								
1																									
2																									
3	5	1	2	3	4	5	4	3	2	1	2	3	4	5	4	3	2	1	2	3	4	5	4	3	2	1
4	4	1	2	3	4	5	4	3	2	1	2	3	4	5	4	3	2	1	2	3	4	5	4	3	2	1
5	3	1	2	3	4	5	4	3	2	1	2	3	4	5	4	3	2	1	2	3	4	5	4	3	2	1
6	2	1	2	3	4	5	4	3	2	1	2	3	4	5	4	3	2	1	2	3	4	5	4	3	2	1
7	1	1	2	3	4	5	4	3	2	1	2	3	4	5	4	3	2	1	2	3	4	5	4	3	2	1
8																									
9																									
	x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
	RAM	1	2	3	4	5	4	3	2	1	2	3	4	5	4	3	2	1	2	3	4	5	4	3	2	1

previous

Next

显示如下，但是点不连续？

y	15-y																								
1																									
2																									
3	12	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
4	11	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
5	10	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
6	9	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
7	8	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
8	7	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
9	6	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
10	5	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
11	4	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
12	3	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
13	2	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
14	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
15																									
16																									
x		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
	RAM	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1

previous

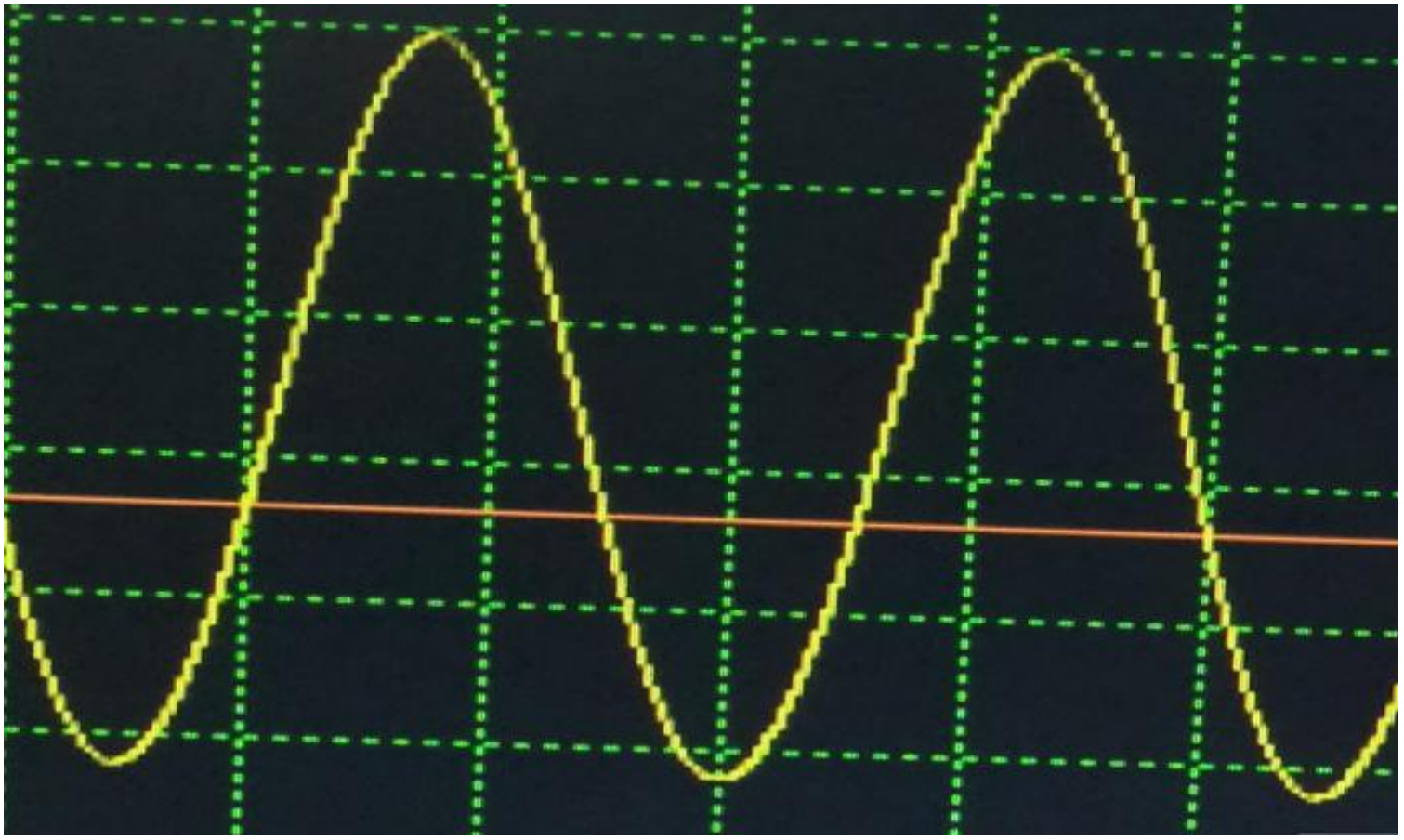
Next

跟上一拍的数据做对比，做点与点之间的连线

y	15-y																								
1																									
2																									
3	12	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
4	11	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
5	10	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
6	9	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
7	8	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
8	7	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
9	6	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
10	5	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
11	4	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
12	3	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
13	2	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
14	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
15																									
16																									
x		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
	RAM	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
	data	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1
	data_ff0		1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1	4	8	12	12	8	4	1	1

Next

处理后的效果图：

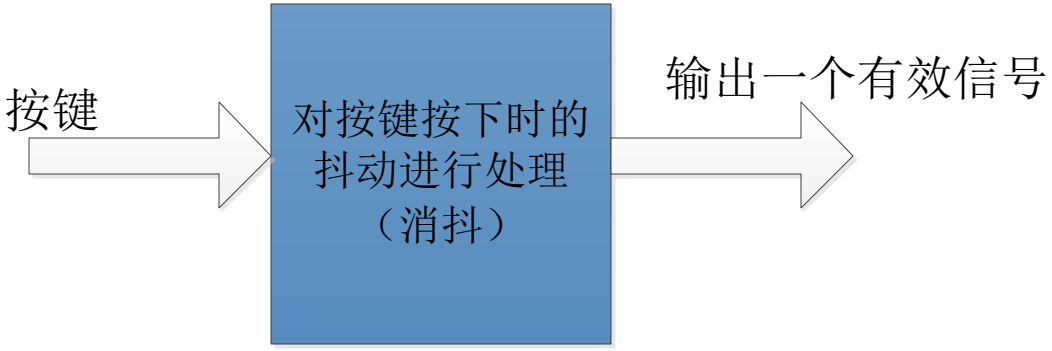


拓展

Home

previous

本模块的主要功能：对按键进行消抖




```
always @(posedge clk or negedge rst_n)begin
    if(!rst_n)begin
        cnt0 <= 0;
    end
    else if(add_cnt0)begin
        if(end_cnt0)
            cnt0 <= 0;
        else
            cnt0 <= cnt0 + 1;
    end
    else
        cnt0 <= 0;
end
assign add_cnt0 =  key_in != key_in_old ;
assign end_cnt0 = add_cnt0 && cnt0 == 250_000-1;
```

Home

经过上述的处理之后，我们能将输入的模拟信号重构在显示器上了，但是显示出的波形大小是固定，除非是改变输入信号的频率，这不便于我们观察信号的更多细节。故以下的拓展是增加能实现波形的放大、缩小功能。

功能实现思路：由于显示器上显示的波形，是根据从RAM IP核中依次取得数据送到相对应的VGA显示位置去的，故可以通过改变取到的数据送出去显示，便可以实现波形的放大、缩小。

实现代码：

```
always @( * )begin
   if(add_flag) begin
        ram_raddr = show_addr + vga_x*i;
    end
   else begin
       ram_raddr =0;
   end
end
```

其中，i是通过按键来设定的，每按下相应的一次按键，i的值就加一，加到某一值之后，每按一下按键，i就减一至到减为一，如此循环。

Home

效果图