超声波测距综合实验

实验报告

袁晨贺 520021910997　冯绍庭 520021911362

2021 年 12 月

1 实验目的

设计并制作超声波应用电路，实现超声波测距，并将测得的距离显示在数码管上。综合了课程所学的数电与模电知识，加深项目参与者对 FPGA 和运放的认识，学习并使用 FPGA 编程语言参与电路搭建和调试。

2 实验原理

2.1 超声波测距

激励信号通过超声波发射器发射超声波，声波在空气中传播，经障碍物反射或者直接传播至超声波接收器，获得收发信号间的时间差t，结合声波在空气中的传播速度v，即可得出发射器与障碍物及接收器的距离d，如图 1 所示为超声波收发器，T 为发射器，R 为接收器，且管脚中更靠近外壳的为负。

图1 超声波收发器

2.2 接收电路对信号处理

超声波信号通过第一级运放高通滤波电路，滤去低频信号；第二级带通滤波电路选出我们需要的 40kHz 信号；经过第三级整形电路输出 echo 信号到 FPGA，得到反馈。

3 实验设计

3.1 总体思路

如图 2 所示为实验设计总体思路示意图，核心控制器产生激励信号，为能更好得振荡超声波发射器，将激励信号经过电平转换后加至发射器，接收器接收的声波信号，声波信号经过信号调理模块进行滤波，比较和电平转换，送至核心控制器，核心控制器记录收发时间差，计算距离。

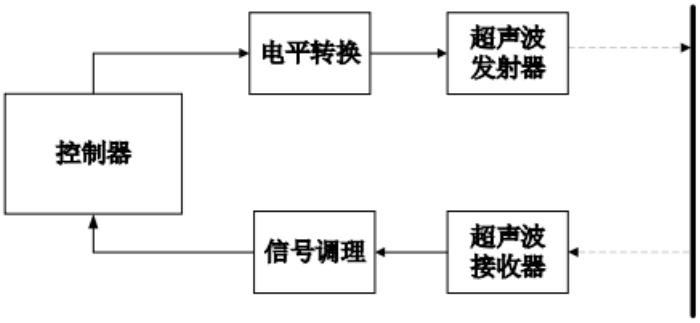

图 2 总体思路示意图

3.2 FPGA 部分

完整代码及管脚约束代码见附录。

3.2.1 整体思路

伪代码如下:

```
loop:
    if press button
        send out 40kHz square wave (#40)
        start receive detection
    if receive
        calculate distance
        display with transistor
```

```
//各端口及中间变量定义如下
input   clk,
input   start,
input   receive,
output  reg signal_1,
output  reg signal_2,
output  [6:0] a_to_g,
output  dot,
output  [3:0] led_bits ,
output  reg receive_detect,        // LED for display
output  reg start_detect,          // ~
output  reg flag_square_detect,    // ~
output  reg flag_button_detect     // ~
);

reg [3:0]   t_led_bits     ;
reg [31:0]  clk_cnt        ;
reg         flag_button    ; // for debounce
reg [9:0]   delay          ; // ~
reg         flag_square    ; // for #waves control
reg [6:0]   waves          ;
reg [6:0]   waves_cnt      ;
reg         flag_receive   ;
integer     ultrasound_speed;
integer     t              ; // timekeeping
reg         t_rst          ;
integer     distance       ;
reg [3:0]   num            ;
reg [31:0]  d              ; // for specified freguency
```

3.2.2 发射模块

如果想要发送 40kHz 的两列反向方波，可以将两个信道依次在高低电平之间转换，每 12.5us 转换一次。FPGA 内置时钟采取 10ns 计数一次，所以设置变量 d，每当 d 达到 1250 便对 signal1 和 signal2 取反。

关键变量：

d	发波计时变量
signal1	发射信道 1
signal2	发射信道 2

发射模块代码如下：

```
always@(posedge clk) begin
    clk_cnt <= clk_cnt + 1;  // 10ns
    if (t_rst) begin
        t <= 0;
        t_rst <= 0;
    end else
        t <= t + 1;

    if (start && (~flag_button)) begin  // press the button
        flag_button <= 1;
        flag_square <= 1;
        flag_receive <= 0;
        t_rst <= 1;
        d <= 0;
    end

    if (flag_square) // generate square wave module
        if (d == 1250) begin // 40kHz
            d <= 0;
            signal_1 <= signal_2;
            signal_2 <= ~signal_2;
            waves_cnt <= waves_cnt + 1;
        end else begin
            d <= d+1;
            signal_1 <= signal_1;
            signal_2 <= signal_2;
            waves_cnt <= waves_cnt + 1;
        end
    if (waves_cnt >= waves) begin
        flag_square <= 0;
        waves_cnt <= 0;
    end

    if (flag_button && t >= delay * 1000) begin // debounce
        flag_button <= 0;
    end
```

3.2.3 晶体管显示模块

每一位依次显示距离的相应数位，利用视觉暂留效应，可以读出距离。

```
//输出显示
always@(*) begin // display
   case (clk_cnt[15:14])
      0: begin
         num <= distance % 10;
         t_led_bits <= 4'b0001;
      end
      1: begin
         num <= distance / 10 % 10;
         t_led_bits <= 4'b0010;
      end
      2: begin
         num <= distance / 100 % 10;
         t_led_bits <= 4'b0100;
      end
      3: begin
         num <= distance / 1000 % 10;
         t_led_bits <= 4'b1000;
      end
   endcase
end

always@(*)  begin   // LED for display
   receive_detect <= flag_receive;
   start_detect <= start;
   flag_square_detect <= flag_square;
end

D0_display myD0_display(.D0_bits(t_led_bits),.D0_NUM(num),.D0_a_to_g(a_to_g),.D0_dot(dot),.D0_led_bits(led_bits)) ;
```

3.3 模电部分

3.3.1 发射端

发射激励信号选择 40 个 40kHz 的脉冲激励信号，如图 3 所示为通过 Max232 芯片进行电平转换的参考电路，控制器输出两路反相的脉冲激励信号，经 Max232 转换为电压更高的差分信号激励信号激励超声波发射器。电路采用单电源低电压供电。

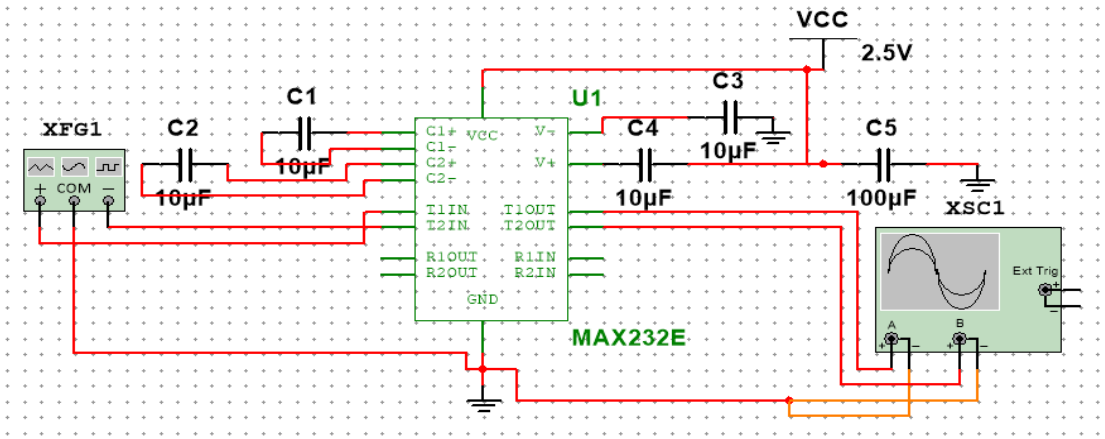

图 3 电平转换参考电路

3.3.2 滤波电路

滤波电路由高通滤波级联带通滤波电路构成。

3.3.2.1 高通滤波电路

高通滤波电路先将输入信号滤去部分低信号杂音，具体参数选择如图 4。

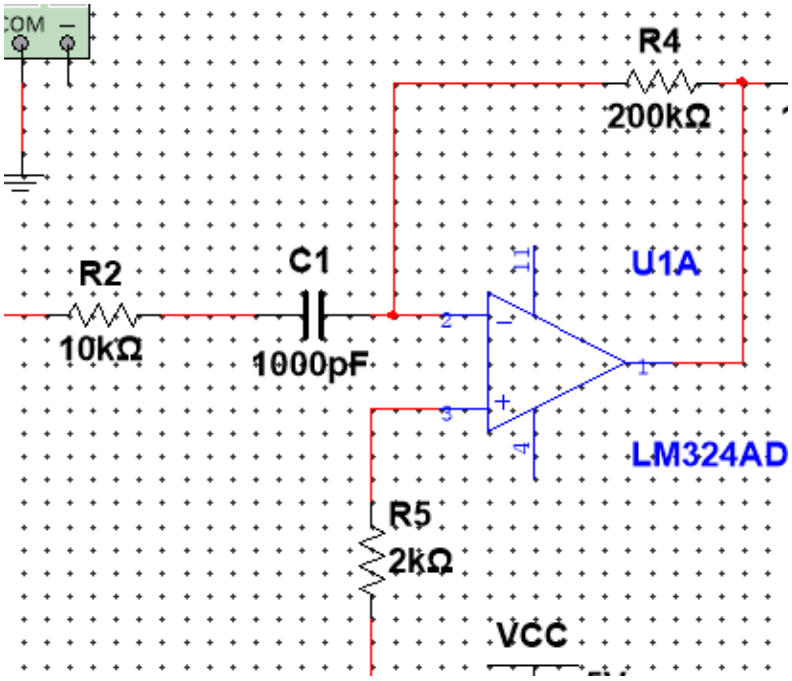

图 4 高通滤波电路

按照如下公式记录元件参数:

$$\frac{V_0}{R_4} = \frac{-V_i}{\frac{1}{sC_1} + R_2}$$

$$A_u(s) = \frac{R_4}{R_2 + \frac{1}{sC_1}} = \frac{R_4}{R_2} \cdot \frac{1}{1 + j\frac{f_0}{f}}$$

$$f_0 = \frac{1}{2\pi C_1 R_2}$$

3.3.2.2 带通滤波电路

带通滤波电路进一步滤去不满足控制频率的信号，具体参数选择如图 5。
可以列出如下六个方程:

$$U_0 = R_{12} i_1$$

$$U_t = U_0 - \left(R_{12} + \frac{1}{sC_6}\right) i_1$$

$$U_t = U_0 - \frac{1}{sC_7} i_2$$

$$U_t = U_i + R_{10} i_3$$

$$U_t = R_{13} i_4$$

$$i_1 + i_2 = i_3 + i_4$$

解得:

$$A_u = \frac{U_i}{U_0} = -\frac{R_{12} C_6}{R_{10}(C_6 + C_7)} \cdot \frac{1}{1 + j\left(\frac{f}{f_1} - \frac{f_2}{f}\right)}$$

$$f_1 = \frac{C_6 + C_7}{2\pi R_{12} C_6 C_7}$$

$$f_2 = \frac{R_{13} + R_{10}}{2\pi R_{13} R_{10}(C_6 + C_7)}$$

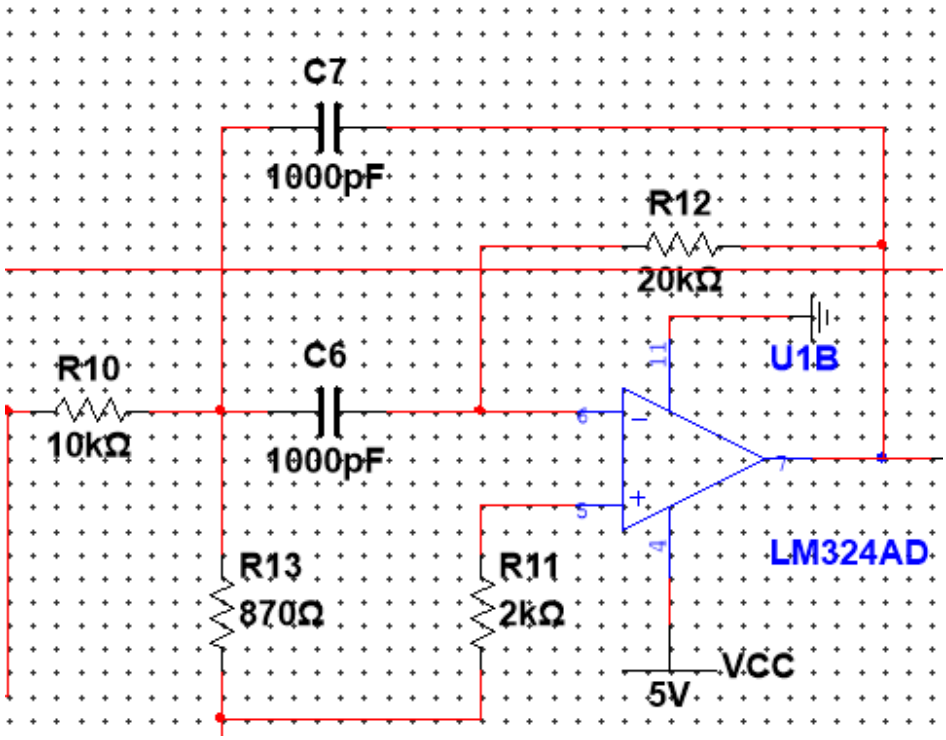

图 5 带通滤波电路

3.3.2.3 参数选择

由公式可以影响通频带和放大倍数的电阻电容值，大刀阔斧地选参。通过 Multisim 的波特仪微调参数改进效果。

3.3.3 整形电路

整形电路由比较器及 NPN 晶体管的电平转换电路组成，将滤波输出正弦信号整形为方波信号。U_-约为 2.5V，当U_+为正弦波中心线以上部分时（>2.5V），U_O'输出高电平，三极管导通，i_C很大，U_O输出为低电平。当U_-为正弦波中心线以下部分时（<2.5V），U_O'输出低电平，三极管处于截止区，$i_C \approx 0$，U_O输出为高电平。

整形电路具体参数见图 6 。

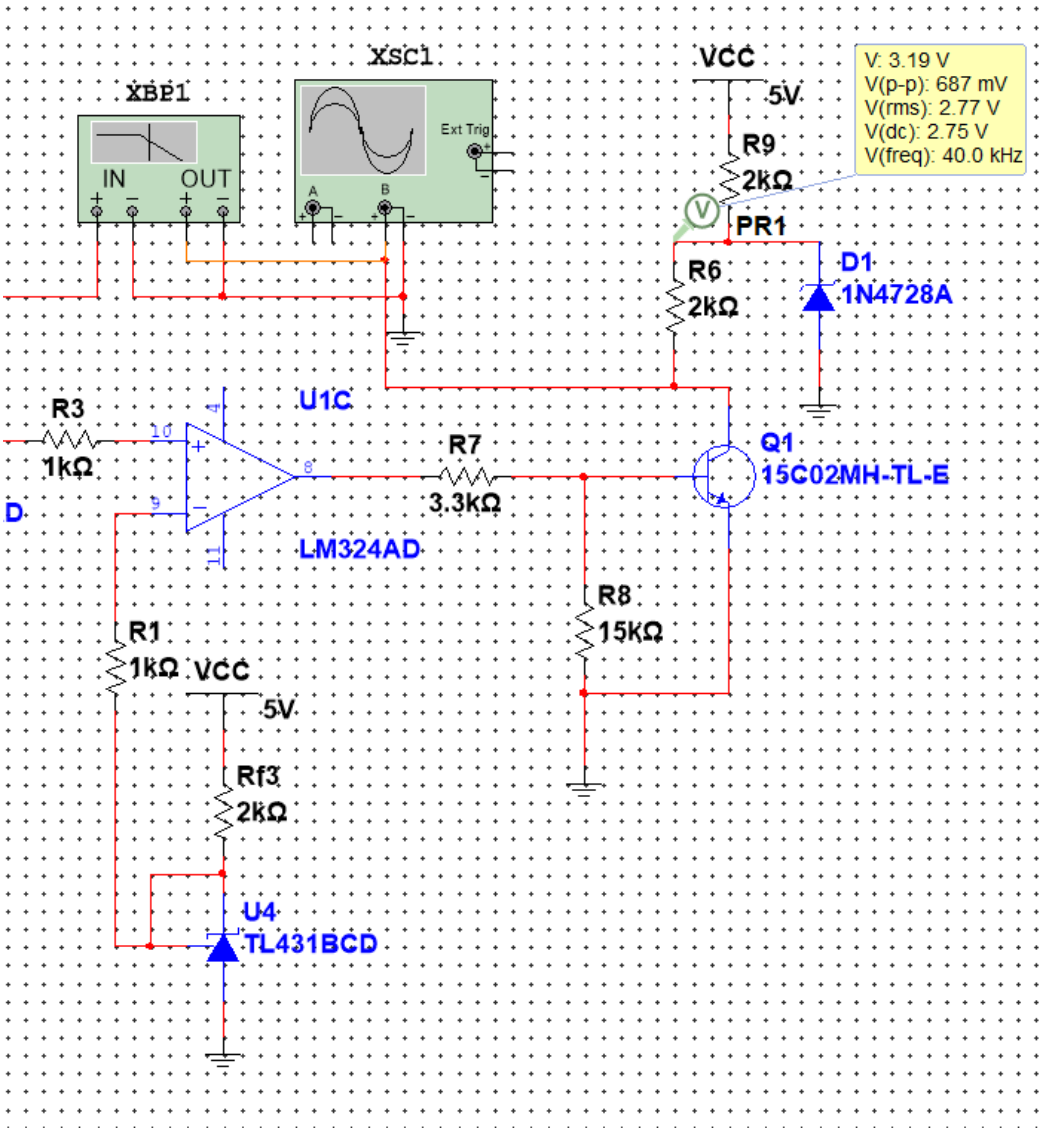

图 6 整形电路

4 实验结果

4.1 仿真测试

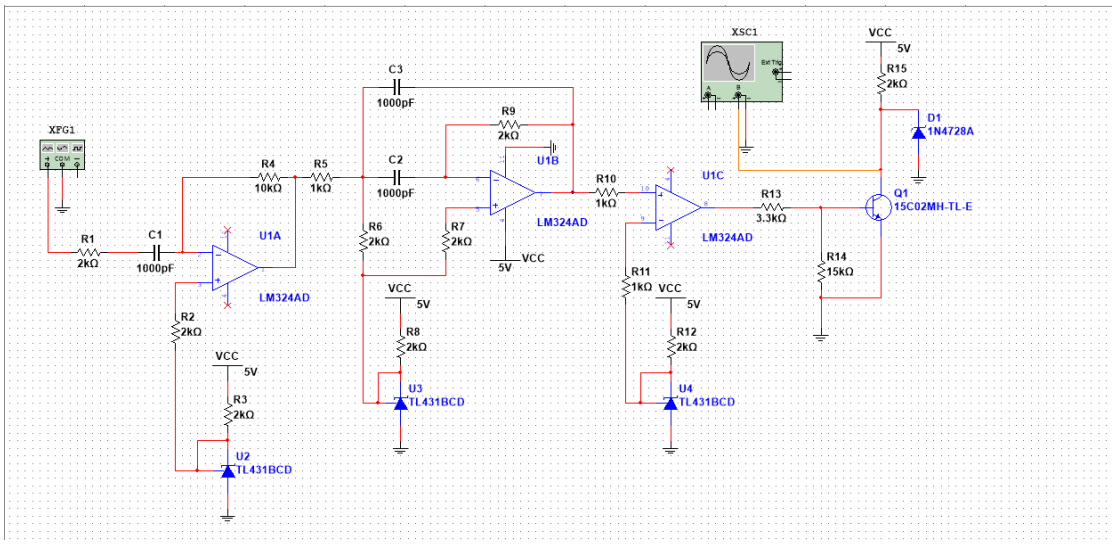

图 7 仿真电路电路图

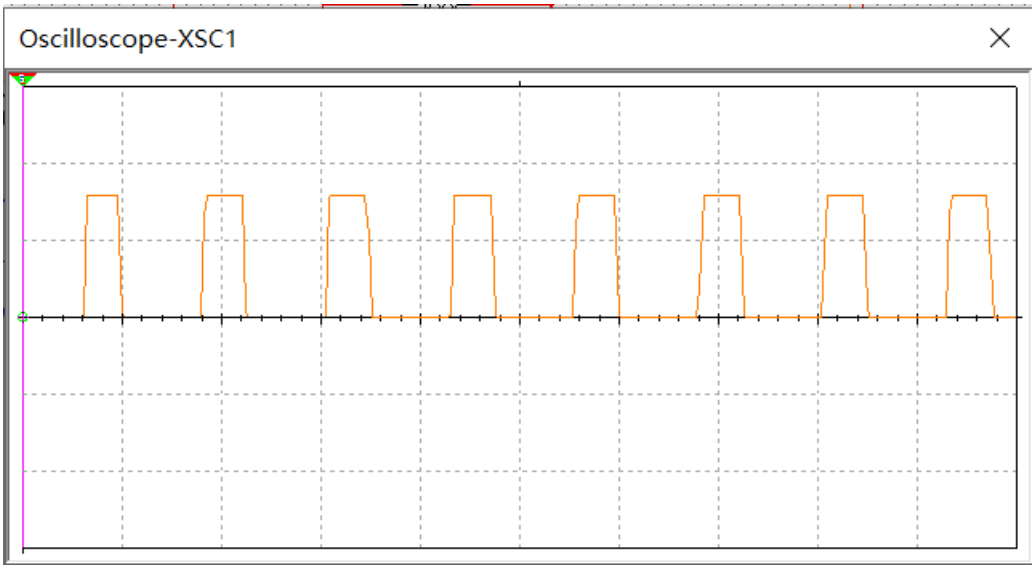

图 8 输出端示波器结果显示

仿真电路与最终输出分别如图 7、图 8 所示，实验电路在仿真软件中是可行的，符合预期。

4.2 电路实物

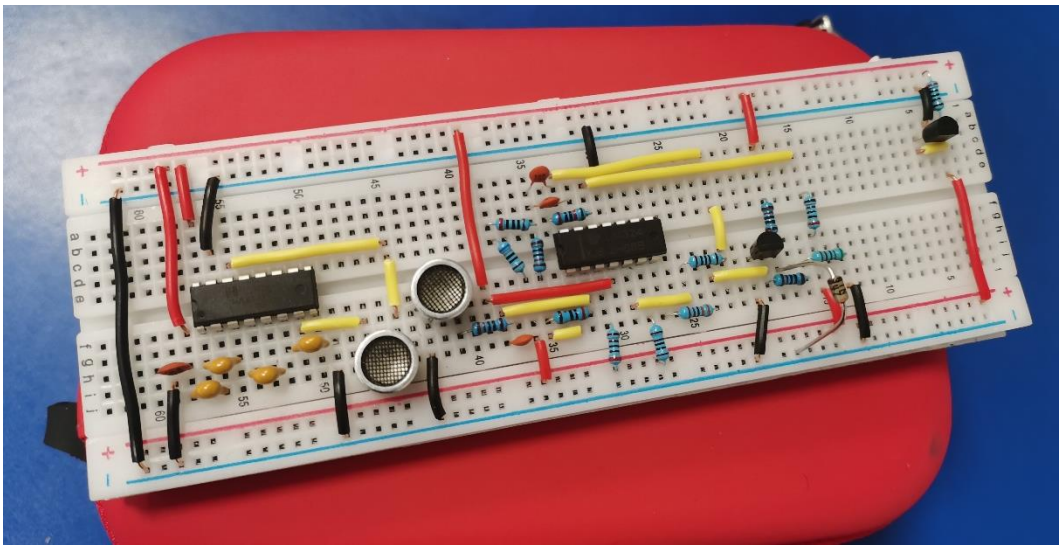

图 9 最终电路搭建结果

4.3 性能测试

电路性能测试 （检测前填写完成，教师抽检）预留测试端接入示波器，测试器输出波形。	FPGA 显示距离值 Cm	示波器同时显示 FPGA 发送端及接收端波形，用时间标尺测量发送及接收时间间隔	
		时间/ms	计算距离(cm)= 时间 X 声速/2
障碍物距离： 10cm	9.7	0.75	10.5
障碍物距离： 20cm	19.5	1.40	19.6

障碍物距离（自定）： 15 cm	14.5	1.05	14.7
障碍物距离最小： 1.8 cm	1.8	0.16	2.24
障碍物距离最大： 24 cm	23.5	1.70	23.8

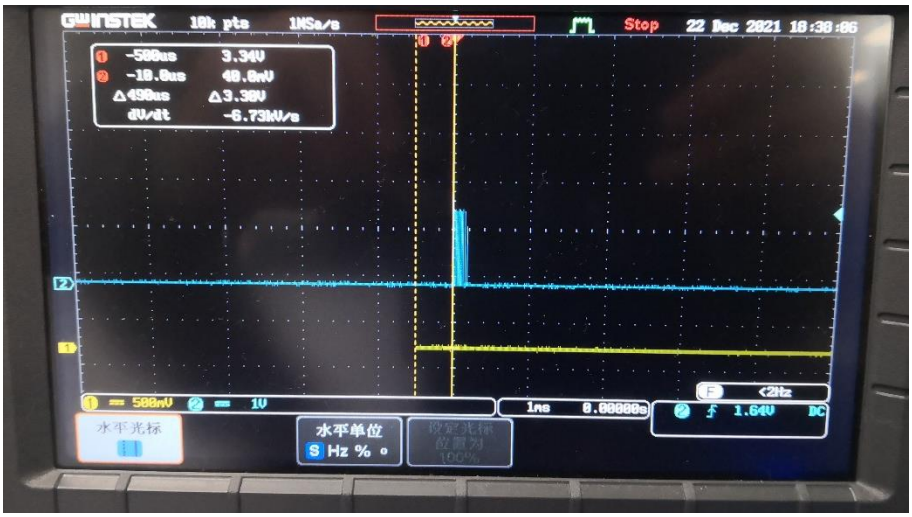

图 10 测试时示波器显示结果

4.4 结果分析

性能测试表格显示，测量误差在接受范围内，不过当距离过小时受最短距离影响，测量结果变得比较不精确。此外由于电路参数选择原因，接收器最远测量距离只有约 25cm，这一点比较遗憾。同时，由于布线时未考虑两个超声波探头间的距离问题，布线时二者过近，导致测量时接收探头容易直接接收到发射器发出的超声波，导致测量结果显示为 3cm，受器材限制我们在测量时被迫用洞洞板进行遮挡。这也导致当我们想要调整接收电路增大测量距离时，继续增大距离会导致接收到穿过洞洞板的波，造成错误，25cm 已是使用洞洞板遮挡的前提下能测到的最远距离，这一点是比较遗憾的，如果能使用别的遮挡材料，则可以得到更大的最大测量距离。

5 感想与致谢

首先感谢上海交通大学电子信息与电气工程学院为本次实验提供的 FPGA 板、各种电子元器件、实验工具和试验场地，其次也感谢所有在项目过程中提供帮助的老师和同学，没有你们就没有本次项目的完成，谢谢你们。

本次项目对于我们而言难度颇高，尤其是在基本没有接触过 Verilog 语言和模电课程进度较慢的前提下，能完成本次项目实为不易，这两周里三四次从下午两点一直做到晚上，确实很辛苦。不过这样的付出是值得的，最终的成品也比较符合我们的预期，在项目的过程中我们也学习到了很多知识，同时也锻炼了我们的能力，总的来说收获颇丰。

最后，再次感谢学校与所有帮助过我们的人，也感谢参与评测过程的老师，谢谢你们。

附录

主程序代码

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2021/12/10 14:00:58
// Design Name: ultrasound measuring distance
// Module Name: UltrasoundDME
// Project Name: ultrasound measuring distance
// Target Devices: FPGA
// Tool Versions: 3
// Description: ultrasound measuring distance
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module D0_display( // transistor module
    input [3:0] D0_bits,
    input [3:0] D0_NUM,
    output reg [6:0] D0_a_to_g,
    output D0_dot,
    output [3:0] D0_led_bits
);
    assign D0_dot = D0_led_bits[1];
    assign D0_led_bits = D0_bits;
    always @(*) begin
        case(D0_NUM)
            0:D0_a_to_g=7'b1111110;
            1:D0_a_to_g=7'b0110000;
            2:D0_a_to_g=7'b1101101;
            3:D0_a_to_g=7'b1111001;
            4:D0_a_to_g=7'b0110011;
            5:D0_a_to_g=7'b1011011;
            6:D0_a_to_g=7'b1011111;
```

```
                7:D0_a_to_g=7'b1110000;
                8:D0_a_to_g=7'b1111111;
                9:D0_a_to_g=7'b1111011;
                default: D0_a_to_g=7'b1111110;
            endcase
        end
endmodule

module UltrasoundDME(
        input     clk,
        input     start,
        input     receive,
        output    reg signal_1,
        output    reg signal_2,
        output    [6:0] a_to_g,
        output    dot,
        output    [3:0] led_bits ,
        output    reg receive_detect,              // LED for display
        output    reg start_detect,                // ~
        output    reg flag_square_detect,          // ~
        output    reg flag_button_detect           // ~
        );

        reg [3:0]     t_led_bits          ;
        reg [31:0]    clk_cnt             ;
        reg           flag_button         ; // for debounce
        reg [9:0]     delay               ; // ~
        reg           flag_square         ; // for #waves control
        reg [6:0]     waves               ;
        reg [6:0]     waves_cnt           ;
        reg           flag_receive        ;
        integer       ultrasound_speed;
        integer       t                   ; // timekeeping
        reg           t_rst               ;
        integer       distance            ;
        reg [3:0]     num                 ;
        reg [31:0]    d                   ; // for specified freguency

        initial begin
            clk_cnt             <=      0     ;
            flag_button         <=      0     ;
            flag_square         <=      0     ;
            d                   <=      0     ;
            delay               <=      1000;
```

```
        waves                   <=      40  ;
        waves_cnt               <=      0   ;
        flag_receive            <=      0   ;
        receive_detect          <=      0   ;
        ultrasound_speed        <=      280 ;
        distance                <=      0   ;
        num                     <=      0   ;
        t                       <=      0   ;
        t_rst                   <=      0   ;
    end

    always@(posedge clk) begin
        clk_cnt <= clk_cnt + 1;   // 10ns
        if (t_rst) begin
            t <= 0;
            t_rst <= 0;
        end else
            t <= t + 1;

        if (start && (~flag_button)) begin   // press the button
            flag_button <= 1;
            flag_square <= 1;
            flag_receive <= 0;
            t_rst <= 1;
            d <= 0;
        end

        if (flag_square) // generate square wave module
            if (d == 1250) begin // 40kHz
                d <= 0;
                signal_1 <= signal_2;
                signal_2 <= ~signal_2;
                waves_cnt <= waves_cnt + 1;
            end else begin
                d <= d+1;
                signal_1 <= signal_1;
                signal_2 <= signal_2;
                waves_cnt <= waves_cnt + 1;
            end
        if (waves_cnt >= waves) begin
            flag_square <= 0;
            waves_cnt <= 0;
        end
```

```
        if (flag_button && t >= delay * 1000) begin // debounce
            flag_button <= 0;
        end

        if (receive) begin
            if (~flag_receive) begin
                flag_receive <= 1;
                distance <= ultrasound_speed * t / 200000;
            end
        end
    end

    always@(*) begin // display
        case (clk_cnt[15:14])
            0: begin
                num <= distance % 10;
                t_led_bits <= 4'b0001;
            end
            1: begin
                num <= distance / 10 % 10;
                t_led_bits <= 4'b0010;
            end
            2: begin
                num <= distance / 100 % 10;
                t_led_bits <= 4'b0100;
            end
            3: begin
                num <= distance / 1000 % 10;
                t_led_bits <= 4'b1000;
            end
        endcase
    end

    always@(*)   begin     // LED for display
        receive_detect <= flag_receive;
        start_detect <= start;
        flag_square_detect <= flag_square;
    end

    D0_display
myD0_display(.D0_bits(t_led_bits),.D0_NUM(num),.D0_a_to_g(a_to_g),.D0_dot(dot),.D0_led_bi
ts(led_bits)) ;
endmodule
```

管脚约束代码

```
set_property PACKAGE_PIN B4 [get_ports {a_to_g[6]}]
set_property PACKAGE_PIN A4 [get_ports {a_to_g[5]}]
set_property PACKAGE_PIN A3 [get_ports {a_to_g[4]}]
set_property PACKAGE_PIN B1 [get_ports {a_to_g[3]}]
set_property PACKAGE_PIN A1 [get_ports {a_to_g[2]}]
set_property PACKAGE_PIN B3 [get_ports {a_to_g[1]}]
set_property PACKAGE_PIN B2 [get_ports {a_to_g[0]}]
set_property PACKAGE_PIN D5 [get_ports dot]
set_property PACKAGE_PIN D15 [get_ports signal_1]
set_property PACKAGE_PIN E15 [get_ports signal_2]
set_property PACKAGE_PIN R15 [get_ports start]
set_property IOSTANDARD LVCMOS33 [get_ports {a_to_g[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a_to_g[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a_to_g[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a_to_g[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a_to_g[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a_to_g[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a_to_g[0]}]
set_property PACKAGE_PIN G2 [get_ports {led_bits[3]}]
set_property PACKAGE_PIN C2 [get_ports {led_bits[2]}]
set_property PACKAGE_PIN C1 [get_ports {led_bits[1]}]
set_property PACKAGE_PIN H1 [get_ports {led_bits[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led_bits[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led_bits[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led_bits[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led_bits[0]}]
set_property PACKAGE_PIN P17 [get_ports clk]
set_property PACKAGE_PIN E17 [get_ports receive]
set_property PACKAGE_PIN K2 [get_ports start_detect]
set_property PACKAGE_PIN J2 [get_ports receive_detect]
set_property PACKAGE_PIN J3 [get_ports flag_square_detect]
set_property PACKAGE_PIN H4 [get_ports flag_button_detect]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports dot]
set_property IOSTANDARD LVCMOS33 [get_ports flag_button_detect]
set_property IOSTANDARD LVCMOS33 [get_ports flag_square_detect]
set_property IOSTANDARD LVCMOS33 [get_ports receive]
set_property IOSTANDARD LVCMOS33 [get_ports receive_detect]
set_property IOSTANDARD LVCMOS33 [get_ports signal_1]
set_property IOSTANDARD LVCMOS33 [get_ports signal_2]
set_property IOSTANDARD LVCMOS33 [get_ports start]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports start_detect]
create_clock -period 10.000 -name clk -waveform {0.000 5.000}
```