关于声源定位

最近在做一个小项目要用到声源定位，刚好和大三学的DSP，通信原理有关，属于是专业对口，本篇文章需要有一点信号方面的基础知识，看不懂的同学不妨学完《信号与系统》跟《概率论》再看看，比较好玩，还有推荐所有电子部的同学好好学一学matlab，这玩意在工程领域方面做数据处理几乎无敌，还可以做FPGA的上层开发

1.GCC-PHAT的两种实现（离散采样，非连续）

1.1互相关算法

假设我们要计算1024个数据的正弦波的GCC-PHAT

我们首先会计算互相关函数R(SIN1,SIN2)

得到互相关函数应该是1024 X2 -1个值

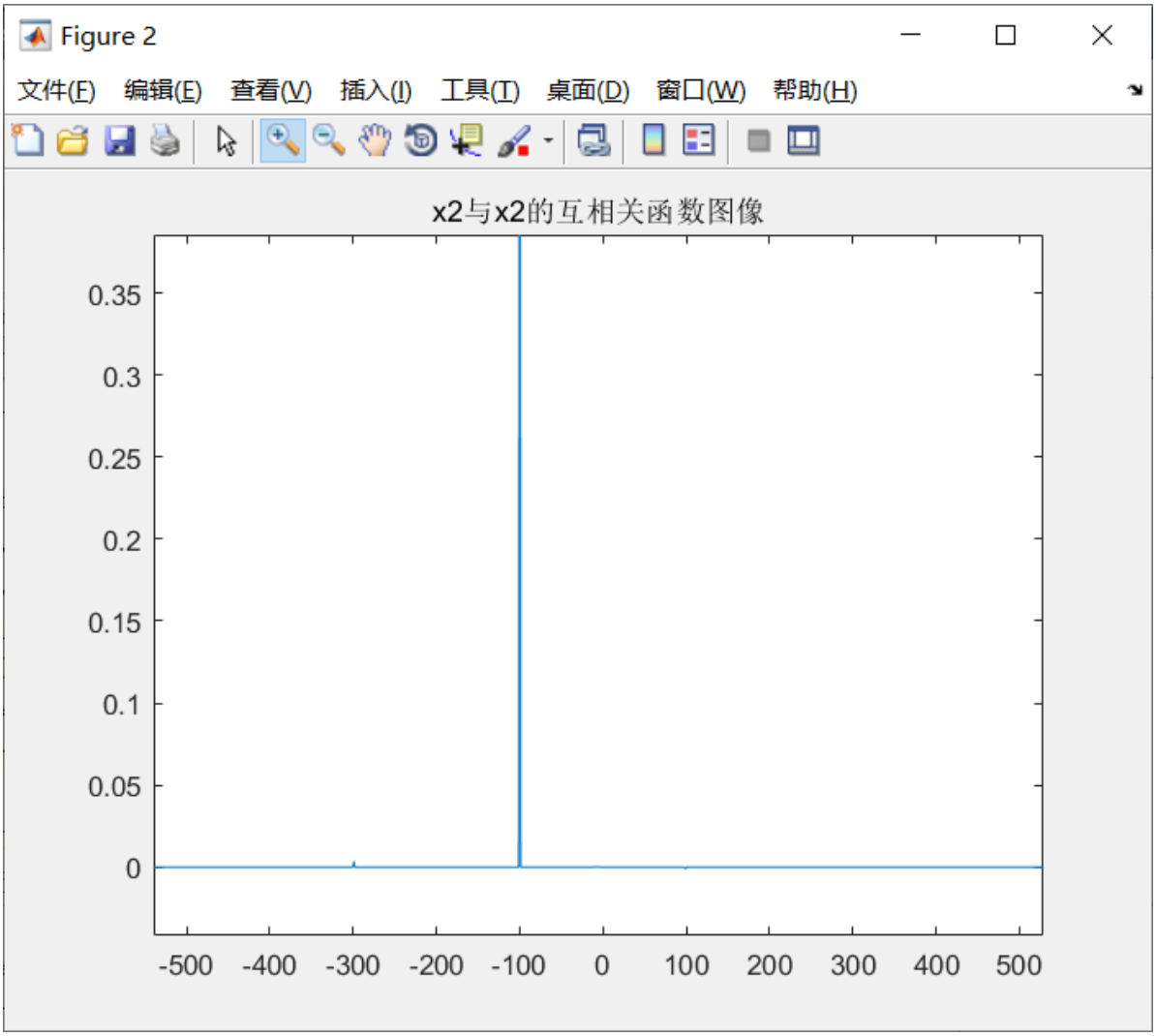
然后再对得到的这个数组进行FFT计算

sin_fft=sin_fft*1./real(sin_fft);

sin_ifft=ifft(sin_fft)

之后便会得到这个数组经过PHAT因子变换之后的广义互相关，对峰值进行检波，得到最大峰值对应的点的序号再进行频率采样模拟化就可以得到对应的延时

```
delay_number=100;
x1=([1024:-1:1,zeros(1,1023)])';
x2=([zeros(1,delay_number),1024:-1:1+delay_number,zeros(1,1023)])';
NFFT=length(x1);
xk1=fft(x1,NFFT);
xk2=fft(x2,NFFT);
P=xk1.*conj(xk2);
A=1./abs(P);
R1=ifft(A.*P);
R2=fftshift(R1);
figure(2)
plot(-(NFFT-1)/2:(NFFT-1)/2,R2);
title('x2与x2的互相关函数图像');
```



1.2直接求取功率谱

首先对两个信号（如果1024位）补0（补到2*1024-1位）进行fft,得到他们的频谱H(SIN1),H(SIN2)

然后对这第二个信号进行共轭变换，之后对这两个信号进行点乘P

之后

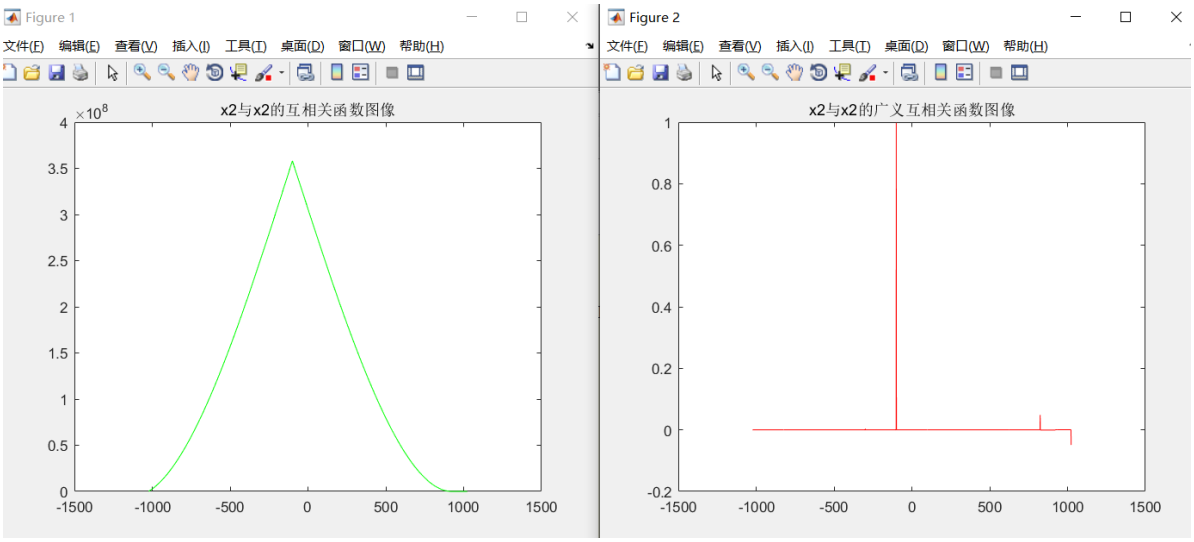
A=1./abs(P);
R1=ifft(A.*P);

但是因为采样的问题，需要对得到的数据做一个类似matlab里面的shiftfft变换，也就是频谱左右交换

之后就会得到广义互相关图像，对图形进行最大峰值检波，得到最大峰值对应的点的序号再进行频率采样模拟化就可以得到对应的延时

给出matlab代码

```
delay_number=100;
x1=([1024:-1:1])';
x2=([zeros(1,delay_number),1024:-1:1+delay_number])';
m=xcorr(x1,x2);
NFFT=length(m);
figure(1);
plot(-(NFFT-1)/2:(NFFT-1)/2,m,'g');
title('x2与x2的互相关函数图像');
n=fft(m);
n=ifft(n*1./abs(n));
figure(2);
plot(-(NFFT-1)/2:(NFFT-1)/2,n,'r');
title('x2与x2的广义互相关函数图像');
```



可以非常好的看到经过phat因子滤波之后，图像的峰值变的极其尖锐，非常容易遍历找到对应的点。

2.verilog算法实现

选用平台：GW2A-LV18PG256C8/I7

首先因为高云的xcorr ip核目前来说是坏的不能用，其次鉴于gcc-phat跟我们直接做互相关对于fpga来说其实相差不大，我们直接采用xcorr算法来做一次就可以峰值检波

其次我们采样得到的1024个mic1和mic2的数据(采样率52.743khz)，但是麦克风的直线距离只有4cm，经过考虑峰值基本只会出现在真负40个数据以内，所以我们只需要做80个互相关就可以

直接贴代码了(核心思想是一个状态机，mic_data_store是mic数据采集的一个.v)

```
module xcorr(input clk,
             input rst_n,
             input start_flag,
             input finish_left,
             input finish_right,
             input signed[17:0]mic_1,
             input signed[17:0]mic_2,
             output reg[5:0]sequence_num,
             output reg finished,
             output reg finished_temp1);

    reg finished_temp;
```

```
    wire data_ok_1_flag;
    wire data_ok_2_flag;
    reg data_ok_1_flag_temp;
    reg data_ok_2_flag_temp;
    reg data_ok_all_flag;
    reg [10:0]mic_1_cnt;
    reg [10:0]mic_2_cnt;
    reg [5:0]shift_index;
    reg signed[35:0] mul_add_result;


    reg [10:0]address_1;
    reg [10:0]address_2;
    wire signed[17:0]data_out_1;
    wire signed[17:0]data_out_2;
    reg signed[35:0] mul_add_max;

    parameter state_0 = 3'b000 ;
    parameter state_1 = 3'b001 ;
    parameter state_2 = 3'b010 ;
    parameter state_3 = 3'b011 ;
    parameter state_4 = 3'b100 ;
    parameter state_5 = 3'b101 ;
    reg [2:0]state;
    initial begin
        // data_ok_1_flag   = 0;
        // data_ok_2_flag   = 0;
        sequence_num        = 0;
        data_ok_1_flag_temp = 0;
        data_ok_2_flag_temp = 0;
        data_ok_all_flag    = 0;
        mic_1_cnt           = 0;
        mic_2_cnt           = 0;
        shift_index         = 0;
        mul_add_result      = 0;
        address_1           = 0;
        address_2           = 0;
        mul_add_max         = 0;
    end
    always @(posedge clk or negedge rst_n) begin
        if (!rst_n)
        begin
            data_ok_1_flag_temp <= 0;
            data_ok_2_flag_temp <= 0;
            finished_temp       <= 0;
            finished_temp1      <= 0;
        end
        else
        begin
            data_ok_1_flag_temp <= data_ok_1_flag;
            data_ok_2_flag_temp <= data_ok_2_flag;
            finished_temp       <= finished;
            finished_temp1      <= finished_temp;

        end
```

```
    end
    always @(posedge clk or negedge rst_n) begin
        if (!rst_n)
        begin
            data_ok_all_flag = 0;
        end
        else if ((data_ok_1_flag_temp&&!data_ok_1_flag))
        begin
            data_ok_all_flag <= 1;
        end

            end



            always @(posedge clk or negedge rst_n) begin
                if (!rst_n)
                begin
                    state = state_0;
                    mic_1_cnt <= 0;
                    mic_2_cnt <= 0;
                    shift_index    = 0;
                    finished       = 0;
                    mul_add_max    = 0;
                    mul_add_result = 0;
                end
                else
                begin
                    case (state)
                        state_0://初始
                        begin
                            if (data_ok_all_flag)
                            begin
                                mic_1_cnt <= 0;
                                state     <= state_1;
                            end
                        end
                        state_1://左移
                        begin

                            if (mic_1_cnt == 1023-shift_index)
                                state                          = state_3;
                                //mul_add_result[shift_index] =
mul_add_result[shift_index]+data_out_1*data_out_2;
                                mul_add_result                 =
mul_add_result+data_out_1*data_out_2;
                                mic_1_cnt                      = mic_1_cnt+1;
                                mic_2_cnt                      = mic_2_cnt+1;
                                end
                                state_2://右移
                                begin

                            if (mic_1_cnt == shift_index-30)
```

```
                                state                           = state_3;
                                mul_add_result                  =
mul_add_result+data_out_1*data_out_2;
                                mic_1_cnt                       = mic_1_cnt-1;
                                mic_2_cnt                       = mic_2_cnt-1;
                                // mul_add_result[shift_index] =
mul_add_result[shift_index]+data_out_1*data_out_2;
                                end
                                state_3://shift_index++
                                begin
                                if (mul_add_result>mul_add_max)
                                begin
                                    mul_add_max  = mul_add_result;
                                    sequence_num = shift_index;
                                end
                                mul_add_result = 0;
                                shift_index    = shift_index+1;
                                if (shift_index <= 30)
                                begin
                                    state     <= state_1;
                                    mic_1_cnt <= 0;
                                    mic_2_cnt <= shift_index;
                                end
                                else if (shift_index<61)
                                begin
                                    state = state_2;
                                    mic_1_cnt <= 1023;
                                    mic_2_cnt <= 1023-(shift_index-30);
                                end
                                else
                                begin
                                    shift_index = 0;
                                    state       = state_4;
                                end

                        end
                        state_4://结束
                        begin
                            finished = 1;

                        end
                        state_5:
                        begin
                            finished = 0;
                        end
                    endcase
                end
            end


            mic_data_store mic_data_store_inst1(
            .clk(clk),
            .rst_n(rst_n),
            .mic_1(mic_1),
            .start(start_flag),
```

```
            .output_flag(data_ok_all_flag),
            .finish_left_or_right(finish_right),
            .adb_cnt(mic_1_cnt[9:0]),
            .out_data(data_out_1),
            .output_start_flag(data_ok_1_flag));

            mic_data_store mic_data_store_inst2(
            .clk(clk),
            .rst_n(rst_n),
            .mic_1(mic_2),
            .start(start_flag),
            .output_flag(data_ok_all_flag),
            .finish_left_or_right(finish_left),
            .adb_cnt(mic_2_cnt[9:0]),
            .out_data(data_out_2),
            .output_start_flag(data_ok_2_flag));




            endmodule
```