به نام خدا

گزارشکار ازمایشگاهfpga

ازمایش 7-1

Amplitude Modulation/Demodulation

علیرضا رحمان ستایش

ایمان استوار

صورت مسئله:

Amplitude Modulation/Demodulation

در بسیاری از موارد سیگنال مورد نظر ما قابلیت ارسال از طریق امواج رادیویی را ندارند پس باید مدوله شده و بعد ارسال شوند

یکی از روش های مدوله سازی am میباشد که در ان اطلاعات در دامنه سیگنال قرار میگیرد

در این آزمایش میخواهیم Amplitude Modulation/Demodulation را پیاده سازی کنیم

طرح (پیاده سازی) :

ماژول am_mod :

```
module AM_MOD(

input clock,
output reg [9:0] modulated_signal,
input [9:0] input_signal
   );

      reg signed [15:0]  mem_array [2047:0];
      reg [10:0] counter =0 ;
      reg signed [10:0] input_signal_s;
      initial $readmemh ("sine.txt", mem_array);

       wire signed [26:0] result ;


 assign result = mem_array[counter]*input_signal_s;
```

```
always @(posedge clock ) begin


  input_signal_s<= { 1'b0,input_signal[9],input_signal[8:0] };

counter<=counter+2;
modulated_signal <={~result[26],result[25:17]};
end



endmodule
```

در این ماژول یک موج سینوسی که قبلا توسط متلب تولید شده و فایل آن ذخیره شده توسط دستور initial $readmemh ("sine.txt", mem_array); خوانده شده و درون mem_array ریخته میشود

reg [10:0] counter =0 ; : این رجیستر شمارنده lut موج سینوسی میباشد و lut با ان ادرس دهی میشود و درون پروسه always ;counter<=counter+2 داریم و مقدار کانتر افزوده میشود تا اینکه سرریز کند و به این شکل lut پیموده میشود و موج سینوسی ساخته میشود

رجیستر reg signed [10:0] input_signal_s; درواقع همان موج وردی است که به سورت علامت دار در آمده چون موج ورودی به صورت sob است و باید به صورت مکمل دو در آید درون پروسه always این عمل انجام میشود:

input_signal_s<= { 1'b0,input_signal[9],input_signal[8:0] };

حاصل ضرب سیگنال ورودی و سیگنال سینوسی به صورت کامبینیشنال محاسبه میشود:

assign result = mem_array[counter]*input_signal_s;

ودر هر لبه کلاک ده بیت بالای resualt بر روی خروجی ماژول ریخته میشود

```
modulated_signal <={~result[26],result[25:17]};
```

ماژول : AM_DEMOD

```
module AM_DEMOD(

input clock,
input [9:0] input_signal,
output reg [9:0] output_signal
   );

      reg signed [9:0] input_signal_b_abs;
      always @(posedge clock) begin


      if(input_signal>=512)
      input_signal_b_abs<=(input_signal-512);
      else
      input_signal_b_abs<=(512-input_signal);

      end
```

```
wire [32:0] dout;

reg [9:0] din;



filter YourInstanceName (
        .clk(clock), // input clk
        //.nd(nd), // input nd
        .rfd(rfd), // output rfd
        .rdy(rdy), // output rdy
        .din(din), // input [9 : 0] din
        .dout(dout)); // output [32 : 0] dout

        reg perivious_rdy=0;
        reg perivious_rfd=0;
        always @ (posedge clock) begin

                perivious_rdy<=rdy;
                if((rdy==1) && perivious_rdy==0)
```

```
        output_signal<={~dout[29],dout[28:20]};
        perivious_rfd<=rfd;
        if((rfd==0) && perivious_rfd==1)
        din<=input_signal_b_abs;
end

                                                    endmodule
```

در این ماژول ابتدا در این قسمت از سیگنال ورودی قدر مطلق گرفته میشود :

```
 always @(posedge clock) begin
 if(input_signal>=512)
 input_signal_b_abs<=(input_signal-512);
 else
 input_signal_b_abs<=(512-input_signal);

 end
```

سپس قدر مطلق موج ورودی با hand shaking به صورت زیر به فیلتر پایین گذر داده میشود

```
always @ (posedge clock) begin
        perivious_rdy<=rdy;
        if((rdy==1) && perivious_rdy==0)
        output_signal<={~dout[29],dout[28:20]};
```

```
        perivious_rfd<=rfd;
        if((rfd==0) && perivious_rfd==1)
        din<=input_signal_b_abs;
    end
```

در واقع هر وقت rdy از صفر به یک تغییر کرد یا خروجی فیلتر روی خروجی ماژول قرار میگیرید و هر وقت rfd از یک به صفر تغییر کرد قدر مطلق سیگنال ورودی به فیلتر داده میشود

درون تاپ ماژول ارتباط ماژول های am_demod و am_mod و adc و dac فراهم شده

توسط ماژول my_ClockDivider کلاک 12 مگاهرتز به adc و dac داده شده

```
module Top_Module(

    input SysClk,

    input [ 9:0 ] ADC_A,
    input [ 9:0 ] ADC_B,


    output ADC_CLKIN,
    output ADC_OE,
    output SCLK,
```

```
    output SDI,

    output  DAC_CLKA,
    output DAC_WRTA,
    output  [ 9:0 ] DAC_A,

    output DAC_CLKB,
    output DAC_WRTB,
    output  [ 9:0 ] DAC_B


);

     my_ClockDivider instance_name1 (
.clock(SysClk),
.divNum('d2),
.clkout(adc_clk)
);

     ADC myADC (
```

```
   .clk(adc_clk),
   .in(ADC_A),
   .ADC_CLKIN(ADC_CLKIN),
   .SCLK(SCLK),
   .ADC_OE(ADC_OE),
   .SDI(SDI)

   );
assign DAC_CLKA=adc_clk;
assign DAC_WRTA=adc_clk;

assign DAC_CLKB=adc_clk;
assign DAC_WRTB=adc_clk;

       AM_MOD AM_MOD (
   .clock(SysClk),
   .modulated_signal(DAC_B),
   .input_signal(ADC_B)
   );

        AM_DEMOD AM_DEMOD (
```

```
.clock(SysClk),
.input_signal(ADC_A),
.output_signal(DAC_A)
);


endmodule
```

نتیجه گیری: بعد از انجام شبیه سازی برنامه مربوطه روی برد امتحان شد و موج های مدوله شده و دی مدوله شده بر روی اسیلوسکپ نمایش داده شد