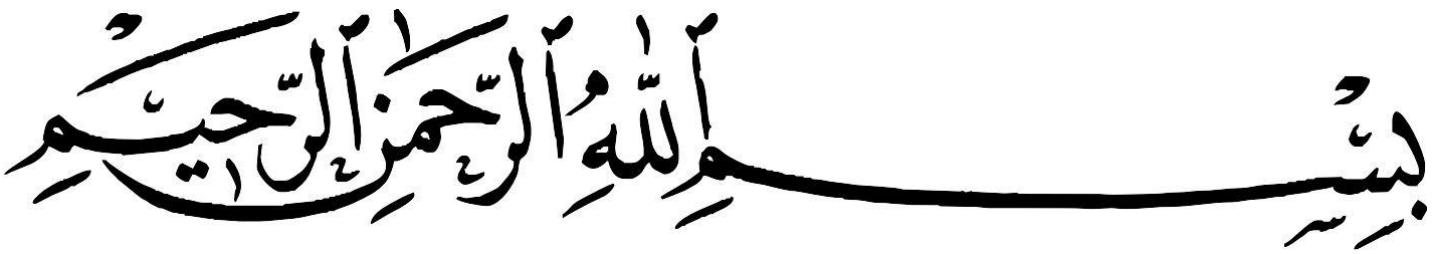

گزارش کار آزمایشگاه FPGA

آزمایش 8

نام استاد : دکتر ثامنی

ایمان استوار9332366

علیرضا رحمان ستایش9332679

صورت مسئله :

FSK Modulation

در این آزمایش قصد داریم FSk reciver , Transmitter پیاده سازی کنیم .

در این آزمایش پک های 8 بیتی را از طریق fskارسال کرده و دریافت و دیکود کنیم .

طرح (پیاده سازی)

ورودی و خروجی های های ماژول گیرنده :

```
module Reciver #(parameter dataWidth =16))
```

ریست

```
   input reset,
```

کلاک

```
input clk,
```

کلاک سمبل ریت

```
input rsclk,
```

سیگنال ورودی

```
input [9:0]signalIn,
```

دیتا دیکود شده خروجی

```
output reg [dataWidth-1:0]out
);
```

رجیستر های داخلی ماژول:

رجیستر دیتا خروجی

```
reg [dataWidth-1:0] outdata = 0;
```

رجیستر شمارنده پکیج دیتا

```
reg [dataWidth/2:0]count=0;
```

تک بیت تشخیص داده شده لحظه ای

```
reg inBit;
```

سایر wireها

Periodلحظه ای پالس دریافتی

```
wire [60:0]T;
```

amplituidلحظه ای پالس دریافتی

```
wire [9:0]A;
```

پروسه های : initial

```
initial out =0;
```

پروسه های : always

تشخیص و مقایسه فرکانس و در نتیجه بیت دریافتی

```
always@(posedge clk)begin
        if(T>='d1350)
                inBit<=0;
        else
                inBit<=1;
end
```

Fsmکنترلی پکیج دریافتی و تغییر خروجی

```
always @(posedge rsclk or negedge reset)begin
        if(!reset)
                count<=0;
        else if (count>=dataWidth)begin
                count <=0;
                out<=outdata;
        end
```

```
        else count<=count+1;
end
```

استفاده از بیت های دریافتی برای تکمیل پکیج دیتا

```
always @(posedge clk)begin
                outdata[count]<=inBit;
end
```

ماژولPuls detector (آزمایش 2)

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    10:28:19 05/26/2017
// Design Name:
// Module Name:    PulsDetector
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module PulsDetector(
   input clk,
         input [9:0]signal,
         output [63:0]period,
         output [9:0]amp
```

```
 );

reg [1:0]state;
wire [63:0]pulsenum;
reg [20:0]TOA;
reg [20:0]PW ;
reg [9:0]PA;
//reg [920:0]count ;
reg [9:0]threshold1 ;
reg [9:0]threshold2 ;

assign period=PW;
assign amp=PA;

initial begin
PA <=10'b00_0000_0000;
TOA <= 0;
state <= 2'b00;
//count=10'b0;
threshold1=100;
threshold2=200;
PW=0;
end

always @ (posedge clk) begin
        if (PA) begin
                threshold2<=(3*PA)>>2;
```

```
                threshold1<=PA>>1+2;
        end
        if ({signal[9:5],5'b00000} < threshold1)begin
                if (state==2'd1)begin
                        state<=2'd0;
                end
                if (state==2'd2)begin
                        state<=2'd3;
                end
                if (state==2'd3)begin//pulse ended
                        state<=2'd0;
                        PW <= pulsenum-TOA;
                end
        end

        if ({signal[9:5],5'b00000}>=threshold1 && {signal[9:5],5'b00000} <
threshold2)begin
                if (state==2'd0)begin
                        state<=2'd1;
                end
                if (state==2'd2)begin
                        state<=2'd3;
                end
        end

        if ({signal[9:5],5'b00000} >= threshold2)begin
                if (state==2'd0)begin
                        state<=2'd1;
```

```
                end
                if (state==2'd1)begin//pulse detected
                        state<=2'd2;
                        TOA <= pulsenum;
                        PA <= 0;
                        //count=count+1;
                end
                if (state==2'd3)begin
                        state<=2'd2;
                end
        end


        if (state==2'd2)begin
                if (signal > PA )
                        PA <= signal;
        end
    end

Binary_Counter mybinary (
   .clk(clk),
   .resetbar(1'b1),
   .enable(1'b1),
   .counter(pulsenum)
   );
```

endmodule

نحوه instance گرفتن :

PulsDetector pd)

. clk(clk) ,

. signal(signalIn) ,

. period(T) ,

. amp(A)

;(

ورودی و خروجی های های ماژول فرستنده :

```
module Transmitter#(parameter dataWidth =16)(
```

ریست

```
   input reset,
```

کلاک

```
input clk,
```

کلاک سمبل ریت

```
input rsclk,
```

کلاک ممووری برای ماژول مولد سیگنال

```
input memclk,
```

دیتا ورودی

```
input [dataWidth-1:0]in,
```

سیگنال خروجی

```
output [9:0]toSend,
```

فلگ ready for data

```
output reg rfd
);
```

رجیستر های داخلی ماژول:

رجیستر دیتا ورودی

```
reg [dataWidth-1:0]indata=0;
```

رجیستر شمارنده پکیج دیتا

```
reg [dataWidth/2:0]count=0;
```

جهت ساخت سیگنال با فرکانس های مختلف

```
reg [7:0]inc=0 ;
```

پروسه های : initial

```
initial rfd =0;
```

پروسه های : always

پیاده سازی fsmجهت ارسال پکیج دیتا

```
always @(posedge rsclk or negedge reset)begin
        if(!reset)
                count<=0;
        else if (count>=dataWidth)begin
                count <=0;
                rfd<=1'b1;
                indata<=in;
        end
        else begin
                count<=count+1;
                rfd<=0;
        end
end
```

تنظیم کلاک خروجی توسط inc با استفاده ازبیت های لحظه ای

```
always @(posedge clk)begin
        if(indata[count]) inc<=11;//110Khz
        else                    inc<=9;//90k
end
```

(3آزمایش) SIN waveماژول

```
module SIN(
        input clk,
        input [7:0]increment,
        output reg[9:0]out
   );

reg  [9:0] mem_array [0:1023];  // 1024 byte array
reg [9:0]address =0;



initial $readmemh ("sin_samples.txt", mem_array,0,1023);
initial out =0;

always @(posedge clk)
        out <= mem_array[address];//memory _ no interpolation

always @(posedge clk) begin
        if(address>=1024) address<=0;
        else address<=address+increment;
end

endmodule
```

نحوه instance گرفتن :

```
SIN sin)
.   clk(memclk) ,
```

. increment(inc) ,

. out(toSend)

;(

ماژول Debouncer

در پروژه های قبلی توضیح داده شده و دلیل استفاده این ماژول دیبانس کردن کلید ورودی جهت ریست کردن fsm

```
module Debouncer(input in ,output reg out ,input clk);
	reg [15:0] cnt;
	initial begin
	//out<=in;
	out<=0;
	cnt<=0;
	end
always @ (posedge clk)begin
	if(out !=in)
		cnt<=cnt+1;
	else
		cnt<=0;
if(cnt==16'hffff)
	out<=in;
end


endmodule
```

به صورت زیر instanceمیگیریم :

```
	Debouncer decouncer (
```

```
.in(en),
.out(den),
.clk(clk)
);
```

ماژول Topmodule

```
module Topmodule(
```

کلاک ورودی

```
input SysClk,
```

ریست

```
input Reset,
```

پایه های DACA

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

پایه های کلاک DACA

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

پایه های کلاک ADC

```
output [9:0]ADC_A,
output ADC_CLKIN,
output ADC_OE,
output SCLK,
output SDI,
```

ورودی dip switch

```
input [7:0]DipSw,
```

خروجی های 7segment

```
output [6:0]segment1pin,
output [6:0]segment2pin
);
```

Inctanceماژول فرستنده جهت ارسال دیتا

```
Transmitter #(.dataWidth(8) )Tr (
  .reset(dReset),
        .clk(clk),
  .rsclk(rsclock),
  .memclk(clock_3),
  .in(DipSw),
  .toSend(sig), //90Khz or 100Khz
  .rfd(rfd)
  );
```

Inctance ماژول فرستنده جهت دریافت دیتا

```
        Reciver #(.dataWidth(8) )RE (
  .reset(dReset),
        .clk(clk),
        .signalIn(ADC_A),
  .rsclk(rsclock),
  .out(data)
  );
```

Inctance ماژول مقسم کلاک برای DAC , ADC

```
FrequencyDivider  #(.DIVIDENUMLENGTH(2)) half(
   .clock(clk),
   .divNum('d2),//12Mhz
   .clkout(clock_2)
   );
```

Inctance ماژول مقسم کلاک برای memory clk ماژول sin wave generator در فرستنده

```
FrequencyDivider  #(.DIVIDENUMLENGTH(5)) inc(
   .clock(clk),
   .divNum('d24),//1Mhz
   .clkout(clock_3)
   );
```

Inctance ماژول مقسم کلاک برای فرکانس symbolrate ارال داده

```
FrequencyDivider #(.DIVIDENUMLENGTH(16)) rs  (
   .clock(clk),
   .divNum('d4800), //5Khz
   .clkout(rsclock)
   );
```

ماژول ADC

```
ADC adcA (
   .clk(clock_2),
   .in(ADC_A),
   .ADC_CLKIN(ADC_CLKIN),
   .SCLK(SCLK),
   .ADC_OE(ADC_OE),
   .SDI(SDI)
```

```
);
```

ماژول تبدل باینری به BCD(آزمایش1)

```
BCD_Counter toBCD (
    .in(data),
    .clk(clk),
    .out1(out1),
    .out0(out0)
    );
```

دیکودر 7segment

```
        Decoder_7 d1 (
    .in(out1),
    .out(segment1pin)
    );
```

دیکودر 7segment

```
        Decoder_7 d2 (
    .in(out0),
    .out(segment2pin)
    );

Debouncer debouncer (
    .in(Reset),
    .out(dReset),
    .clk(clk)
    );
```

ماژول 7segment decoder

```
module Decoder_7 )
                                                    input wire[ 3 : 0 ] in,
                                                    output reg [ 7 : 0 ] out
                                                    ; (

 //abcde fg = fd [ 6 ] , d [ 5 ] , d [ 4 ] , d [ 3 ] , d [ 2 ] , d [ 1 ] , d [ 0 ] g
 //a c t i v e low
always @( * ) begin
out <= 7'b0000_001;
case ( in )
'4b0000 : out <= 7'b0000_001;
'4b0001 : out <= 7'b1001_111;
'4b0010 : out <= 7'b0010_010;
'4b0011 : out <= 7'b0000_110;
'4b0100 : out <= 7'b1001_100;
'4b0101 : out <= 7'b0100_100;
'4b0110 : out <= 7'b0100_000;
'4b0111 : out <= 7'b0001_111;
'4b1000 : out <= 7'b0000_000;
'4b1001 : out <= 7'b0000_100;
endcase
end
endmodule
```

و نحوه inctanceگرفتن :

Decoder_7 dec)

. in(seg1) ,

. out(segment1pin)

;(

Decoder_7 dec2)

. in(seg2) ,

. out(segment2pin)

;(

به پیوست شبیه سازی با استفاده از GTKW در فولدر پروژه قرار داده شده است .