EE314 Digital Electronics Laboratory Term Project Proposal Report
FPGA Based Oscilloscope

Ekin Ylmaz - 2094738 Nail Tosun - 2094563
Electric and Electronic Engineering Departmant, METU

2018
May

Introduction

In this project a simple FPGA based oscilloscope which can visualize a given single frequency input signal from a signal generator with different waveforms on VGA screen and which can also measure the given parameters in Table 1 is aimed to be designed. Iteration limit here is what determines the resolution. Since the limit is given as 250, using 20mV steps a maximum of 0 to 5 Volts range can be scanned via these steps.

Table 1: The required measurements that the designed oscilloscope can achieve

Parameter	Max-Min Values	Resolution
Frequency	DC 20kHz	10 Hz
Vpp	0 to 5 Volts	20 mV
Vrms	0 to 5 Volts	20 mV
Voffset	0 to 5 Volts	20 mV

Working Prinsiple of Design

For this purpose a system which consists of a ADC measuring module to quantize the signal, a data storage module where the transmitted quantized data is stored, a computation module that performs the required measurements and calculations to determine frequency, peak-to-peak voltage, waveform, rms voltage, and offset voltage and finally a VGA display module where the visualization of the results is realized, is designed. Related block diagram of the overall system can be seen in Figure 1. In addition to these modules, there will be also a mode selection input which decides whether the oscilloscope will measure in AC coupling mode or DC coupling mode. According to the decision made, entire signal will be visualized for DC mode or DC value will be subtracted first then the signal will demonstrated in AC mode. This mode selection part will be realised via switches on the FPGA board. Finally, there must be also adjustable time/div, volt/div and autoscale option too. Those arrangements will be done by using push buttons on the board.

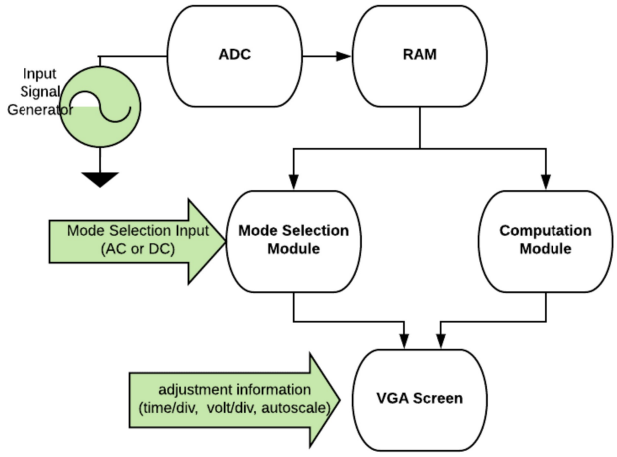

Figure 1: Block diagram of the overall system

Equipment

VGA connector, VGA display screen, De1-SoC FPGA development board. Mainly the VGA outputs and embedded ADC of the board will be used in this project. Input signals will be applied from the ADC header and the output will be taken from the VGA output of the development board for visualization.

Rms voltage measurement unit

Starting the design, an RMS voltage measurement module is designed first since it will also be a crucial part of AC DC selection module too.

Discrete RMS formula is following;

$$V_{rms} = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1} V_i^2}$$

Then we firstly took square every serial in values. This module HDL code is following;

Listing 1: Square module

```
module square(clk,serial_in,square_out);
input clk;
input [11:0] serial_in;
output reg [31:0] square_out;

always @(posedge clk)
begin
square_out<=serial_in*serial_in;
end
endmodule
```

Then we sum squares of serial in values after to one period and divide the period N for mean value of squares. (Frequency of the signal is an *input* for this module). HDL code is following;

Listing 2: Averaging module

```
module rms_finder(clk, freq, average, serial_in);
input clk;
input [11:0] serial_in;
input [31:0] freq;
output reg [31:0] average;
```

```
integer index,N,dum_index,sum;
initial
begin
period = 0;
sum <= 0;
N <= 0;
index <= 0;
average <= 0;
dum_index <= 1;
end
always @(posedge clk)
begin
        //N=f_sampling/freq //input and sampling frequency determines the N
        N = 8; //for testing
        if (index<N)
                begin
                sum <= sum + serial_in;
                index <= index+1;
                end
        else if(index==N)
        begin
        dum_index <= dum_index + 1;
        average <= sum/(N*dum_index);
        index <= index +1;
        dum_index <= dum_index + 1;
        end
        else index <= 0;
end
endmodule
```

Then we write another module that simple take square root of the input. HDL code of this module is following;

Listing 3: Square-root module

```
module m(clk, mean_input, rms);
input clk;
input [11:0] mean_input;
output reg [11:0] rms;
reg [11:0]temp=1;

always@(posedge clk)
begin
        if((temp*temp)>=mean_input)
                rms<=temp;
    else
       temp<=temp+1;
  end
endmodule
```

Listing 4: Top level design

```
module top(clk,serial_in,freq,rmsfindresult);
input clk;
input [11:0]serial_in;
input [31:0] freq;
output  reg [11:0] rmsfindresult;

wire[11:0] square_out;
wire[11:0] average;
wire[11:0] rms;
```

```
reg [11:0] serial_in_sq;
reg[11:0] mean_input;

square(.clk(clk),
.serial_in(serial_in),
.square_out(square_out)
);

Mean(.clk(clk),
.serial_in_sq(square_out),
.freq(freq),
.average(average)
);

root(.clk(clk),
.mean_input(average),
.rms(rms)
);

always begin
serial_in_sq <= square_out;
mean_input <= average;
rmsfindresult <= rms;
end
endmodule
```

Simulation Results

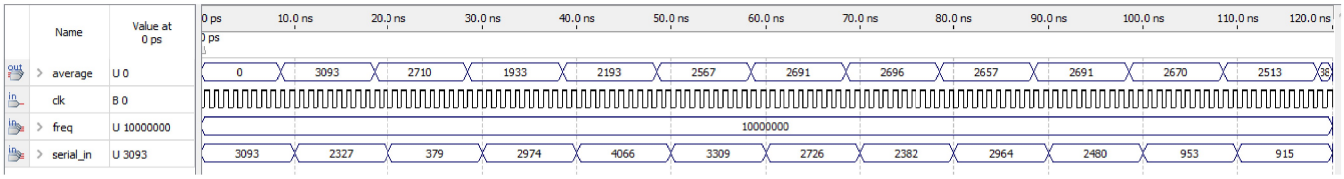

Figure 2: Simulation results of mean-finder

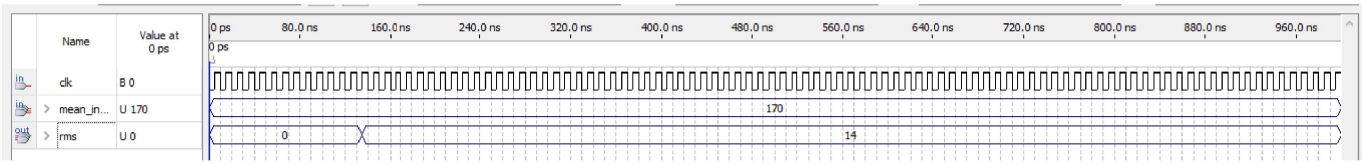

Figure 3: Simulation results of square root module

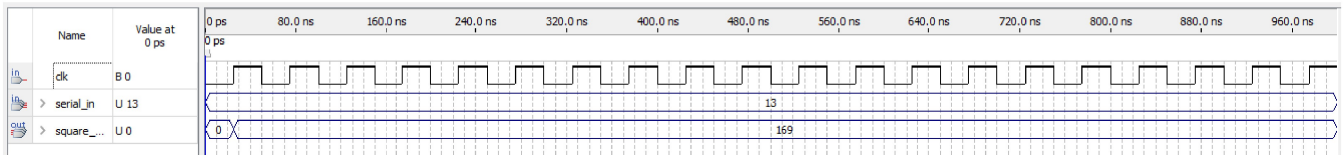

Figure 4: Simulation results of square module