Anita Ghandehari
810195533

Soheil Shirvani
810195416

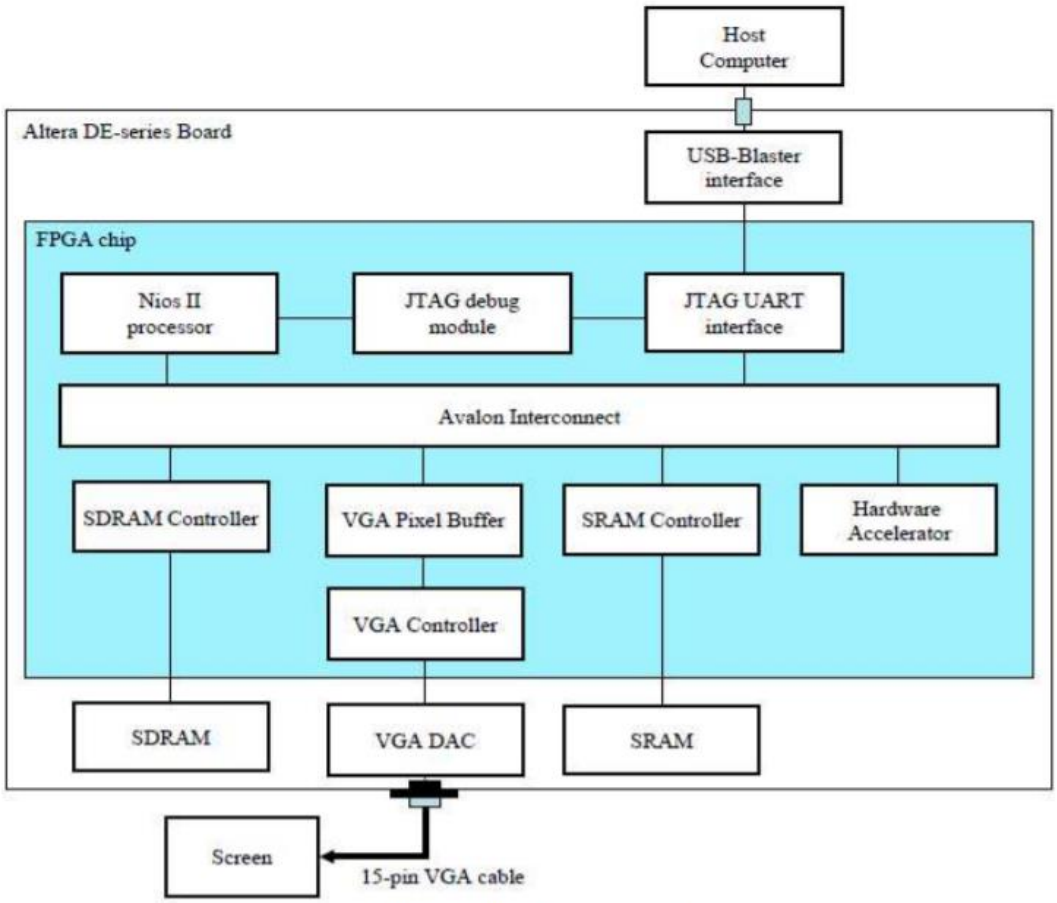
Lab4 _ FPGA Reports

مقدمه:

در این آزمایش می خواهیم صوت را گرفته در بازه های مساوی تقسیم کرده میانگین مجموع مقادیر ان را در ان بازه زمانی محاسبه کرده و نمودار مقادیر محاسبه شده را روی صفحه نمایش نشان دهیم
سپس در زمان پخش ان به ازای مقدار زمان گذشته روی ان بازه از تصویر کشیده شده, تصویر دیگری زیر آن نمایش داده می شود که نشانگر زمان گذشته شده است

این کار ها را یک بار به صورت نرم افزاری سپس به صورت سخت افزاری انجام می دهیم

آزمایش:



شکل ۱ سیستمی با شتابدهنده‌ی سخت‌افزاری

Anita Ghandehari
810195533

Soheil Shirvani
810195416

Lab4 _ FPGA Reports

قسمت نرم افزاری:

ابتدا 3 باکس که در آزمایش های قبل طراحی شده را در بالای صفحه کشیده و یک خط برای ماکسیموم مقدار نمودار و یک خط برای مینیموم مقدار آن می کشیم و بخش تصویر جدید را مشخص می کنیم

کد های این بخش به صورت زیر است:

```
//Green Box For Record //Record ******************
blue_x1 = 15; blue_x2 = 25; blue_y1 = 4; blue_y2 = 14;
// character coords * 4 since characters are 4 x 4 pixel buffer coords (8 x 8 VGA coords)
color = 0x100F;     // a medium blue color
alt_up_pixel_buffer_dma_draw_box (pixel_buffer_dev, blue_x1 * 4, blue_y1 * 4, blue_x2 * 4,
    blue_y2 * 4, color, 0);
alt_up_char_buffer_string (char_buffer_dev, text_Record, blue_x1 + 2, blue_y1 + 4);

//Green Box For Play //PLAY**********************
blue_x1 = 35; blue_x2 = 45; blue_y1 = 4; blue_y2 = 14;
// character coords * 4 since characters are 4 x 4 pixel buffer coords (8 x 8 VGA coords)
color = 0x100F;     // a medium blue color
alt_up_pixel_buffer_dma_draw_box (pixel_buffer_dev, blue_x1 * 4, blue_y1 * 4, blue_x2 * 4,
    blue_y2 * 4, color, 0);
alt_up_char_buffer_string (char_buffer_dev, text_Play, blue_x1 + 4, blue_y1 + 4);

//Green Box For Echo //Echo**********************
blue_x1 = 55; blue_x2 = 65; blue_y1 = 4; blue_y2 = 14;
// character coords * 4 since characters are 4 x 4 pixel buffer coords (8 x 8 VGA coords)
color = 0x100F;     // a medium blue color
alt_up_pixel_buffer_dma_draw_box (pixel_buffer_dev, blue_x1 * 4, blue_y1 * 4, blue_x2 * 4,
    blue_y2 * 4, color, 0);
alt_up_char_buffer_string (char_buffer_dev, text_Echo, blue_x1 + 4, blue_y1 + 4);
```

```
// Draw Audio Display
blue_x1 = 0; blue_x2 = 100; blue_y1 = 20; blue_y2 = 60;
// character coords * 4 since characters are 4 x 4 pixel buffer coords (8 x 8 VGA coords)
color = 0x0328;     // a medium blue color
alt_up_pixel_buffer_dma_draw_box (pixel_buffer_dev, blue_x1 * 4, blue_y1 * 4, blue_x2 * 4,
    blue_y2 * 4, color, 0);

//Write Audio
blue_x1 = 39; blue_y1 = 22;
color = 0xABCD;
alt_up_char_buffer_string (char_buffer_dev, test_Audio, blue_x1 , blue_y1 );
//Audio Line
blue_x1 = 10; blue_x2 = 70; blue_y1 = 21;
color = 0xABCD;
alt_up_pixel_buffer_dma_draw_hline(pixel_buffer_dev,blue_x1 * 4, blue_x2 * 4, blue_y1 * 4, color, 0);
blue_x1 = 10; blue_x2 = 70; blue_y1 = 23;
alt_up_pixel_buffer_dma_draw_hline(pixel_buffer_dev,blue_x1 * 4, blue_x2 * 4, blue_y1 * 4, color, 0);
//Audio OverFlow
alt_up_pixel_buffer_dma_draw_hline(pixel_buffer_dev,10, 300, 105, 0xABCD, 0);
```

Anita Ghandehari
810195533

Soheil Shirvani
810195416

Lab4 _ FPGA Reports

سپس برای بدست آوردن مجموع مقادیر در این بازه ها متغییر های مورد استفاده را تعریف می کنیم:

```
extern int Play_Index;
extern int Play_Flag;
```

و برای این که مقادیر 32 بیت هستند ما متغییر های 64 بیت تعریف می کنیم تا این مجموع از تعداد بیت ما بیشتر نشود

```
long long int Average[N];
long long int Avrage_Hardware[N];
```

سپس کد مربوط به محاسبه مجموع این مقادیر را می نویسیم که داریم:

```
/************************************************************************************************/
void Audio_Average(){
    int i=0;
    int j=0;
    long long int Sum=0;

    int Duration = BUF_SIZE / N;
    for(j=0;j<N;j++){
        Average[j] = 0;
        Sum =0;
        for(i=0;i<Duration;i++){
            Sum = ((alt_u64)abs(((alt_u32)l_buf[i + j * Duration]))) + Sum;
            Sum = ((alt_u64)abs(((alt_u32)r_buf[i + j * Duration]))) + Sum;
        }

        Average[j] = (((Sum * 400) / (Duration) ) >> 32);
        if((200-Average[j]) < 110 ){
            Average[j] = 95;
        }
    }
}
/************************************************************************************************/
```

سپس برای کشیدن مقدار مقدار میانگین ان روی صفحه داریم:

Anita Ghandehari
810195533

Soheil Shirvani
810195416


```
/*****************************************************************************************/
void Plot_AudioRecord(alt_up_pixel_buffer_dma_dev *pixel_buffer){

    // Draw Audio Display
    // character coords * 4 since characters are 4 x 4 pixel buffer coords (8 x 8 VGA coords)
    alt_up_pixel_buffer_dma_draw_box (pixel_buffer, 0 , 20 * 4, 100 * 4,
        60 * 4, 0x0328, 0);

    //Audio Line
    alt_up_pixel_buffer_dma_draw_hline(pixel_buffer,10 * 4, 70 * 4, 21 * 4, 0xABCD, 0);
    alt_up_pixel_buffer_dma_draw_hline(pixel_buffer,10 * 4, 70 * 4, 23 * 4, 0xABCD, 0);
    //Audio OverFlow
    alt_up_pixel_buffer_dma_draw_hline(pixel_buffer,10, 300, 105, 0xABCD, 0);

    int i=0;
    short color = 0x8053;
    int x1 = 10;
    int x2 = 310;
    int gap = 3;
    int Range = ((x2 - x1)/N)-3;

    int x_Start=0,x_End=10;

    for(i=0;i<N;i++){
        x_Start = x_End + gap;
        x_End = x_Start + Range;
        //Draw Audio Range
        alt_up_pixel_buffer_dma_draw_box (pixel_buffer, x_Start, 200-Average[i], x_End,
            200, color, 0);
    }
}
/*****************************************************************************************/
```

که این تابع بعد از Record کردن صدا, صدا می شود تا میانگین این مجموع ها را گرفته به صورت نمودار روی صفحه نشان دهد و مقدار ماکسیموم ان را نیز مشخص می کنیم تا نمودار روی بقیه صفحه نرود

برای نشان دادن اندازه زمان گذشته بعد از پخش و زدن دکمه Play تابع زیر را هر سری در While کلی کد صدا می کنیم:

```
/*****************************************************************************************/
void Plot_Play_Audio(alt_up_pixel_buffer_dma_dev *pixel_buffer){
    int x_Start=0,x_End=10;
    int gap = 3;
    int x1 = 10;
    int x2 = 310;
    int Range = ((x2 - x1)/N)-3;
    short color = 0x1111;


    if(Play_Flag == 1){
        x_Start = Play_Index * Range + gap * Play_Index + 13;
        x_End = x_Start + Range;
        Play_Flag = 0;
        Play_Index = Play_Index + 1;
        alt_up_pixel_buffer_dma_draw_box (pixel_buffer, x_Start, 205, x_End,
            205 + Range , color, 0);
    }
}
/*****************************************************************************************/
```

Anita Ghandehari
810195533

Soheil Shirvani
810195416

Lab4 _ FPGA Reports

که فلگ های هد از فایل اینتراپت مربوط به Audio درست می شود
مانند کد زیر:

```
// output data until the buffer is empty //Play
else if (buf_index_play < BUF_SIZE && Click_Echo == 0)
{

    alt_up_parallel_port_write_data (up_dev->green_LEDs_dev, 0x2); // set LEDG[2] on
    num_written = alt_up_audio_play_r (up_dev->audio_dev, &(r_buf[buf_index_play]),
        BUF_SIZE - buf_index_play);
    /* assume that we can write the same # words to the left and right */
    (void) alt_up_audio_play_l (up_dev->audio_dev, &(l_buf[buf_index_play]),
        num_written);
    buf_index_play += num_written;

    if(buf_index_play >= ((BUF_SIZE / N)*Play_Index)){
        Play_Flag = 1;
    }

    if (buf_index_play >= BUF_SIZE)
    {
        // done playback
        Play_Index = 0;
        alt_up_parallel_port_write_data (up_dev->green_LEDs_dev, 0); // turn off LEDG
        alt_up_audio_disable_write_interrupt(up_dev->audio_dev);
    }
}
else {
    alt_up_parallel_port_write_data (up_dev->green_LEDs_dev, 0x80); //  LEDG
    alt_up_audio_disable_write_interrupt(up_dev->audio_dev);
}
```

بعد از اجرای کد می بینیم که کد کار می کند
برای بدست آوردن زمان از کتابخانه های زیر استفاده می کنیم:

```
#include "sys/alt_timestamp.h"
```

و در کد While آن داریم:

```
printf("[timestamp]timestamp 1:%.3f second\n", (float)alt_timestamp()/(float)alt_timestamp_freq());
Audio_Average(); // Software Calculate
printf("[timestamp]timestamp 1:%.3f second\n", (float)alt_timestamp()/(float)alt_timestamp_freq());
```

در ان با استفاده از تابع های بالا می توانیم زمان مربوط به اجرای کد را داشته باشیم:
در خروجی داریم: 1:85.854 و 1:85.703

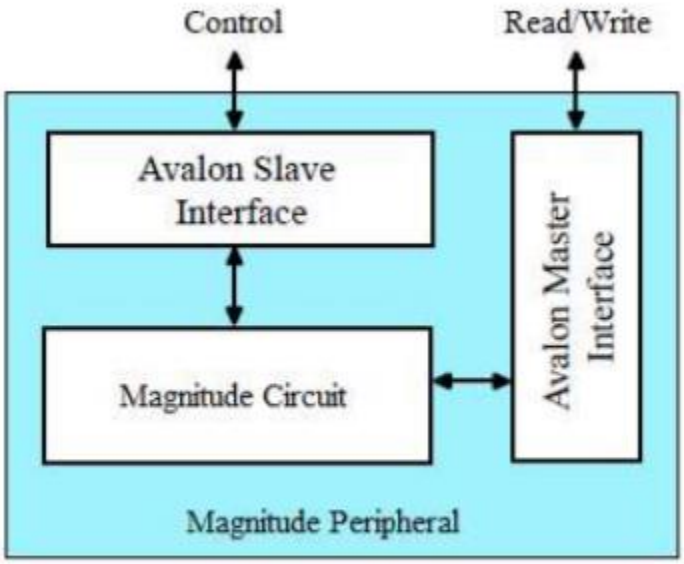
که ان ها مقدار کلاک گذشته تقسیم بر فرکانس ها که همان زمان گذشته را می دهد است
این زمان تقریبا برابر با 4 ثانیه است.

قسمت سخت افزار:

در این قسمت می خواهیم همان کار های قسمت قبل را در سخت افزار انجام دهیم

برای این کار از معماری Slave-master استفاده می کنیم که در ان Slave ما با Nios در ارتباط است و رجیستر های پر شده را می خواند و در صورت نوشتن, می تواند روی ان ها بنویسد.

برای این معماری داریم:



شکل ۳ شمای کلی مدار جانبی محاسبه‌ی متوسط دامنه

ابتدا رابط Slave مربوط به سخت افزار را درست می کنیم که در ان داریم:

قسمت نوشتن از رجیستر ها به شکل زیر است:

```
  else if(AVS_AVALONSLAVE_WRITE)
  begin
    // address is always bytewise so must devide it by 4 for 32bit word
    case(AVS_AVALONSLAVE_ADDRESS >> 2)
    0: slv_reg0 <= AVS_AVALONSLAVE_WRITEDATA;
    1: slv_reg1 <= AVS_AVALONSLAVE_WRITEDATA;
    2: slv_reg2 <= AVS_AVALONSLAVE_WRITEDATA;
    3: slv_reg3 <= AVS_AVALONSLAVE_WRITEDATA;
    default:
    begin
      slv_reg0 <= slv_reg0;
      slv_reg1 <= slv_reg1;
      slv_reg2 <= slv_reg2;
      slv_reg3 <= slv_reg3;
    end
  endcase
  end
```

قسمت خواندن به صورت زیر است:

```
always@(*)begin
  if(AVS_AVALONSLAVE_READ)
  begin
    // address is always bytewise so must devide it by 4 for 32bit word
    case(AVS_AVALONSLAVE_ADDRESS >> 2)
    0: read_data = slv_reg0;
    1: read_data = slv_reg1;
    2: read_data = slv_reg2;
    3: read_data = slv_reg3;
    default:
    begin
      read_data = 0;
    end
    endcase
  end
  else begin
      read_data = 0;
  end
end
```

که در ان خواندن بدون کلاک است و سیگنال Wait_Request 0 است

قسمت رجیستر های ان به شکل زیر است:

```
// do the other jobs yourself like last codes
assign Go = slv_reg0[0];
assign Number = slv_reg0[11:1];
assign Size = slv_reg0[30:12];
assign Slave_Done = slv_reg0[31];
```

و بعد از تمام شدن به شکل زیر می شود:

```
  else if(DONE)
  begin
    slv_reg0[31] <= 1'b1;
    slv_reg0[0] <= 1'b0;
  end
end
```

سپس برای رابط Master آن داریم:

State های قسمت مستر به شکل زیر است:

```
//parameters
localparam  Wait_For_Go = 1;
localparam  Read_Left_Register = 2;
localparam  Read_Right_Register = 3;
localparam  Do_Sum = 4;
localparam  Send_32MSB = 5;
localparam  Send_32LSB = 6;
localparam  Done_State = 7;
```

و رابط بین خروجی های اصلی به شکل:

```
assign DONE = done;
assign AVM_AVALONMASTER_ADDRESS = address;
assign AVM_AVALONMASTER_READ = read;
assign AVM_AVALONMASTER_WRITE = write;
assign AVM_AVALONMASTER_WRITEDATA = writedata;

//My Assigns
assign Wait_Request = AVM_AVALONMASTER_WAITREQUEST;
assign Read_Data = AVM_AVALONMASTER_READDATA;
```

و برای محاسبه مجموع مقادیر ابتدا از رجیستری که ادرس مقدار های راست را دارد حوانده مقادیر را خوانده و جمع می کنیم با مقادیر خوانده شده از چپ.

این کار را به تعداد Size انجام می دهیم تا هر یک از بازه های مشخص شده بدست اید سپس برای تعداد Number بار ان را تکرار می کنیم تا تمام بازه های صفحه نمایش بدست اید

رجیستر ها به شکل زیر است:

Slave Address	31	30..12	11..1	0	
00	Done	Size	Num	Go	Config. Reg.
01					Right Addr.
10					Left Addr.
11					Out Addr.

شکل ۴ رجیسترهای مورد استفاده در رابط Slave

و کد مربوط به هر State به شکل زیر به ترتیب امده است:

```
Wait_For_Go:begin
    if(Wait_Request) begin
        Now_State <= Wait_For_Go;
    end
    else begin
        if(Go) begin
            Now_State <= Read_Left_Register;
            done <= 1'b0;
            read <= 1'b1;
            write <= 1'b0;
            address <= slv_reg2 + Size_Count << 2;
        end
    end
end
```

Anita Ghandehari
810195533

Soheil Shirvani
810195416

Lab4 _ FPGA Reports

```
        Read_Left_Register:begin
            if(Wait_Request) begin
                Now_State <= Read_Left_Register;
            end
            else begin
                Now_State <= Read_Right_Register;
                Left_Reg_Data <= Read_Data;
                read <= 1'b1;
                write <= 1'b0;
                done <= 1'b0;
                address <= slv_reg1 + Size_Count << 2;
            end
        end

        Read_Right_Register:begin
            if(Wait_Request) begin
                Now_State <= Read_Right_Register;
            end
            else begin
                Now_State <= Do_Sum;
                read <= 1'b0;
                write <= 1'b0;
                done <= 1'b0;
                address <= 1'b0;
                Right_Reg_Data <= Read_Data;
            end
        end
```

```
        Do_Sum:begin
            if(Wait_Request) begin
                Now_State <= Do_Sum;
            end
            else if(Size_Count <= Size) begin
                Size_Count <= Size_Count + 1;
                Sum_Reg <= Left_Reg_Data + Right_Reg_Data + Sum_Reg;
                read <= 1'b1;
                write <= 1'b0;
                done <= 1'b0;
                address <= slv_reg2 + Size_Count << 2;
                Now_State <= Read_Left_Register;
            end
            else begin
                Now_State <= Send_32MSB;
                Size_Count <= 0;
                read <= 1'b0;
                write <= 1'b1;
                done <= 1'b0;
                address <= slv_reg3 + Number_Count << 2;
                writedata <= Sum_Reg[2 * AVM_AVALONMASTER_DATA_WIDTH - 1:AVM_AVALONMASTER_DATA_WIDTH];
            end
        end
```

سپس هر یک از این مجموع های بدست آمده را در ادرسی که رجیستر سوم نشان می دهد می نویسد تا تمام بازه ها (Number) تمام شوند و در این حالت به State آخر می رود و سیگنال Done را یک می کند و ان را در رجیستر یک خود می نویسد تا Nios ان را بخواند و از تمام شدن کار خبر می دهد.

```
Send_32MSB:begin
    if(Wait_Request) begin
        Now_State <= Send_32MSB;
    end
    else begin
        Now_State <= Send_32LSB;
        read <= 1'b0;
        write <= 1'b1;
        done <= 1'b0;
        address <= slv_reg3 + 4 + Number_Count << 2;
        writedata <= Sum_Reg[AVM_AVALONMASTER_DATA_WIDTH - 1:0];
    end
end

Send_32LSB:begin
    if(Wait_Request) begin
        Now_State <= Send_32LSB;
    end
    else if(Number_Count <= Number) begin
        Now_State <= Read_Left_Register;
        Number_Count <= Number_Count + 1;
        write <= 1'b0;
        read <= 1'b1;
        done <= 1'b0;
        address <= slv_reg2 + Size_Count << 2;
    end
    else begin
        Now_State <= Done_State;
    end
end
```

```
        Done_State:begin
            done <= 1'b1;
            write <= 1'b0;
            read <= 1'b0;
            address <= 1'b0;
            Number_Count <= 0;
            if(Master_Done)begin
                Now_State <= Wait_For_Go;
            end
            else begin
                Now_State <= Done_State;
            end
        end
    endcase
```

و در اینجا کار قسمت Master تمام می شود

برای ماژول اصلی ما یک فایل درست کرده و از هر دو ی Master و Slave یک Instance گرفته و ان ها را به هم وصل می کنیم تا ماژول کای ما را تشکیل دهد

```
// clock and reset
input wire csi_clock_clk,
input wire csi_clock_reset,

// control interface ports
input wire [avs_avalonslave_address_width - 1:0] avs_avalonslave_address,
output wire avs_avalonslave_waitrequest,
input wire avs_avalonslave_read,
input wire avs_avalonslave_write,
output wire [avs_avalonslave_data_width - 1:0] avs_avalonslave_readdata,
input wire [avs_avalonslave_data_width - 1:0] avs_avalonslave_writedata,

// magnitude interface ports
output wire [avm_avalonmaster_address_width - 1:0] avm_avalonmaster_address,
input wire avm_avalonmaster_waitrequest,
output wire avm_avalonmaster_read,
output wire avm_avalonmaster_write,
input wire [avm_avalonmaster_data_width - 1:0] avm_avalonmaster_readdata,
output wire [avm_avalonmaster_data_width - 1:0] avm_avalonmaster_writedata
);
```

Anita Ghandehari
810195533

Soheil Shirvani
810195416

سپس سخت افزار تولید شده را به Qsys برده و سخت افزار ان را تولید می کنیم و به بقیه مدار وصل می کنیم.

در این حالت Nios یک آدرس به ما می دهد که همان آدرس رجیستر های مربوط به Slave است

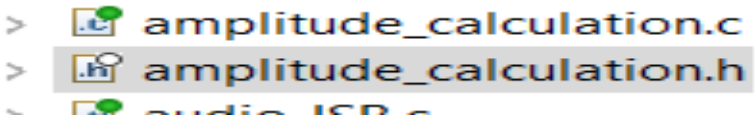
مانند زیر:

```
/*
 * Avrage_Hardware_0 configuration
 *
 */

#define ALT_MODULE_CLASS_Avrage_Hardware_0 Avrage_Hardware
#define AVRAGE_HARDWARE_0_BASE 0x10004000
#define AVRAGE_HARDWARE_0_IRQ -1
#define AVRAGE_HARDWARE_0_IRQ_INTERRUPT_CONTROLLER_ID -1
#define AVRAGE_HARDWARE_0_NAME "/dev/Avrage_Hardware_0"
#define AVRAGE_HARDWARE_0_SPAN 16
#define AVRAGE_HARDWARE_0_TYPE "Avrage_Hardware"
```

سپس از این آدرس استفاده کرده تا کتابخانه ای بنویسیم که مقادیر را در رجیستر ها نوشته و منتظر بماند تا سخت افزار سیگنال پایان را بدهد

برای این کتابخانه دو فیل جدید مانن زیر می سازیم:




```
#include "globals.h"
#include "system.h"

void amplitude_operation(int size, int num, int rbuff_addr, int lbuff_addr, int dest_addr);

void amplitude_circute_stop();

void amplitude_circute_set_size(int size);

void amplitude_circute_set_num(int num);

void amplitude_circute_set_rbuff_addr(int Right_Address);

void amplitude_circute_set_lbuff_addr(int Left_Address);

void amplitude_circute_set_dest_addr(int Dest_Address);

void amplitude_circute_start();

int amplitude_circute_get_status();

int amplitude_circute_get_size();
```

و تعاریف این تابع ها به صورت زیر است:

```
void amplitude_circute_stop(){
    Read_Reg0 = IORD_32DIRECT(0x10004000,0);
    Done_Bit = Read_Reg0 >> 31;
    if(Done_Bit == 1){
        IOWR_32DIRECT(0x10004000,0,Base_Data | 1 << 31);
    }
    else if (Done_Bit == 0){
        IOWR_32DIRECT(0x10004000,0,Base_Data);
    }
}

void amplitude_circute_set_size(int size){
    Base_Data = size;
    Read_Reg0 = IORD_32DIRECT(0x10004000,0);
    Read_Reg0 = Read_Reg0 & 0x80000FFF;
    int Data = Read_Reg0 | (Base_Data << 12);
    IOWR_32DIRECT(0x10004000,0,Data);
}

void amplitude_circute_set_num(int num){
    Base_Data = num;
    Read_Reg0 = IORD_32DIRECT(0x10004000,0);
    IOWR_32DIRECT(0x10004000,0,Read_Reg0 | (Base_Data << 1));
}

void amplitude_circute_set_rbuff_addr(int Right_Address){
    IOWR_32DIRECT(0x10004000,4, Right_Address);
}

void amplitude_circute_set_lbuff_addr(int Left_Address){
    IOWR_32DIRECT(0x10004000,8, Left_Address);
}

void amplitude_circute_set_dest_addr(int Dest_Address){
    IOWR_32DIRECT(0x10004000,12, Dest_Address);
}

void amplitude_circute_start(){
    Read_Reg0 = IORD_32DIRECT(0x10004000,0);
    Base_Data = 1;
    Read_Reg0 = Read_Reg0 & 0xFFFFFFFE;
    IOWR_32DIRECT(0x10004000,0,Read_Reg0 | Base_Data);
}

int amplitude_circute_get_status(){
    Read_Reg0 = IORD_32DIRECT(0x10004000,0);
    Done_Bit = Read_Reg0 >> 31;
    if(Done_Bit == 1){
        return 0;
    }
    if(Done_Bit == 0){
        return 1;
    }
    return 2;
}
```

Anita Ghandehari
810195533

Soheil Shirvani
810195416

Lab4 _ FPGA Reports

و در اخر تابع زیر در While کلی کد صدا می شود تا سخت افزار کار خود را شروع کند و منتظر می مانیم تا کار تمام شود سپس همانند نرم افزاری مقادیر را روی صفحه می کشم:

```
void amplitude_operation(int size, int num, int rbuff_addr, int lbuff_addr, int dest_addr)
{
    amplitude_circute_stop();
    amplitude_circute_set_size(size);
    amplitude_circute_get_size();
    amplitude_circute_set_num(num);
    // also for your debugging make int amplitude_circute_get_num(); (optional)
    amplitude_circute_set_rbuff_addr(rbuff_addr);
    // also for your debugging make int amplitude_circute_get_lbuff_addr(); (optional)
    amplitude_circute_set_lbuff_addr(lbuff_addr);
    // also for your debugging make int amplitude_circute_get_rbuff_addr(); (optional)
    amplitude_circute_set_dest_addr(dest_addr);
    // also for your debugging make int amplitude_circute_get_dest_addr(); (optional)
    amplitude_circute_start();
    while(amplitude_circute_get_status() == 0);
    return;
}
```

سپس در کد While ابتدا تابع بالا را صدا می کنیم تا سخت افزار شروع به کار کند سپس بعد از تمام شدن کار در تابعی دیگر مقادیر میانگین ان را گرفته و برای کشیدن صفحه ان را درست می کنیم.

تابع های زیر برای محاسبه این کار ها است:

```
/*****************************************************************************************/
void Do_Hardware_Calculate(){
    amplitude_operation(Size,Number,(int)r_buf,(int)l_buf,(int)Out_Ram);
//  IOWR_32DIRECT()
}
/*****************************************************************************************/
void Calculate_Avrage(){
    int i;
    for(i=0;i<=Number;i++){
        Average[i] = 0;
        int Anita = Out_Ram[i];
        Average[i] = Out_Ram[i]/Size;
        int Anita2 = Average[i];
        if((200-Average[i]) < 110 ){
            Average[i] = 95;
        }
    }
}
/*****************************************************************************************/
```

و در While اصلی داریم:

```
Do_Hardware_Calculate();//Hardware Calculate
Calculate_Avrage();//For Hardware
```

سپس مانند قبل زمان اجرای ان را بدست می آوریم:

اعداد بدست آماده از قسمت سخت افزار مانند زیر است:

1: 85.350 و 1: 85.322

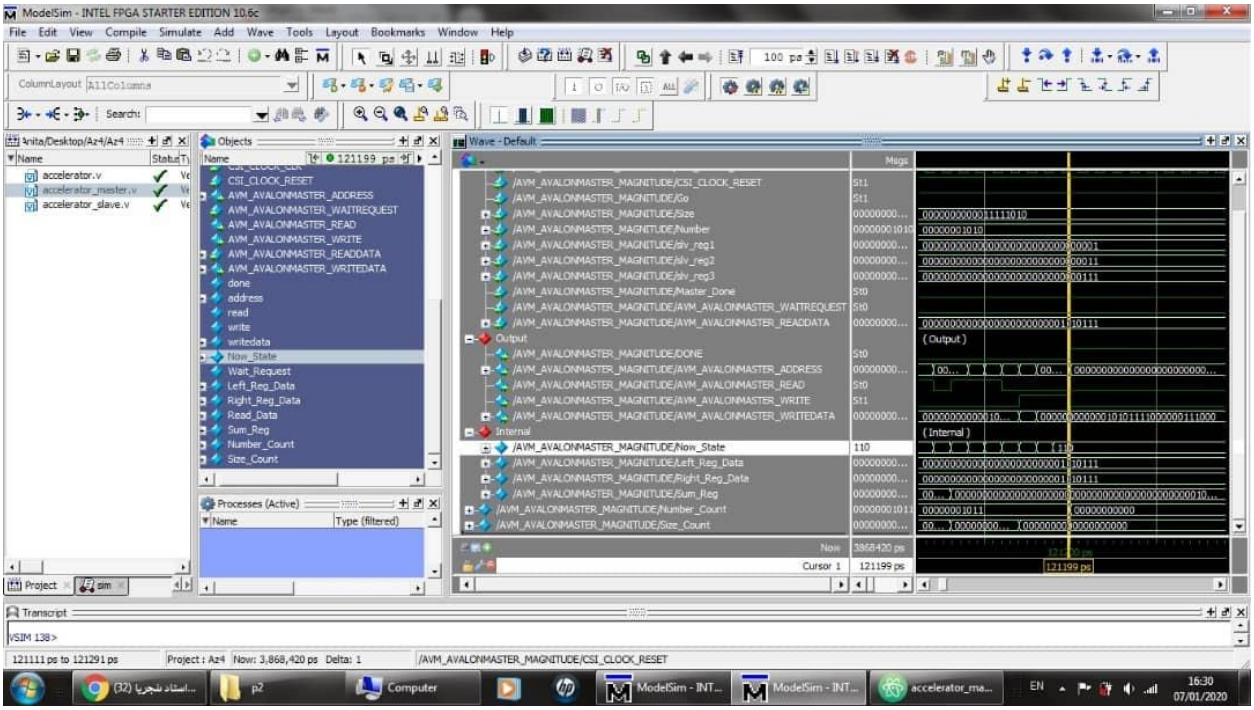
که تقریبا معادل 0.5 ثانیه است.

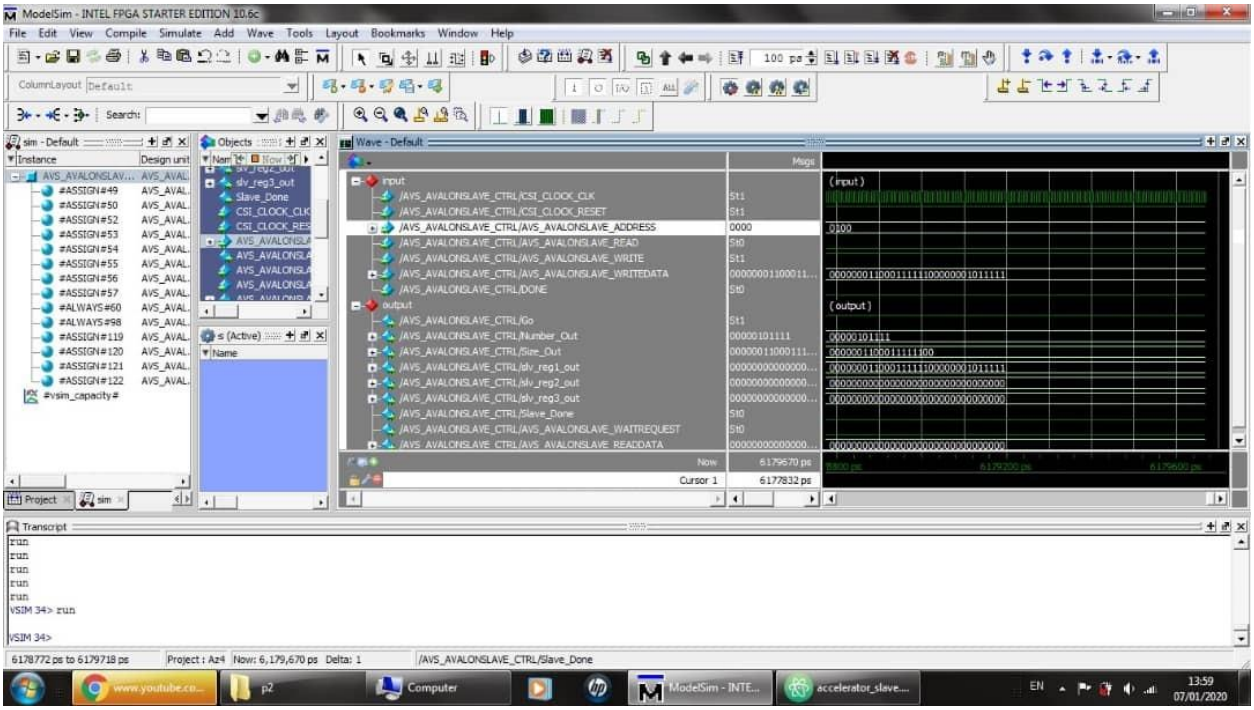
Anita Ghandehari
810195533

Soheil Shirvani
810195416

Lab4 _ FPGA Reports

برای تایید قسمت سخت افزاری ان را در Modelsim تست کرده ایم که شکل های مربوط به آن مانند شکل زیر است:

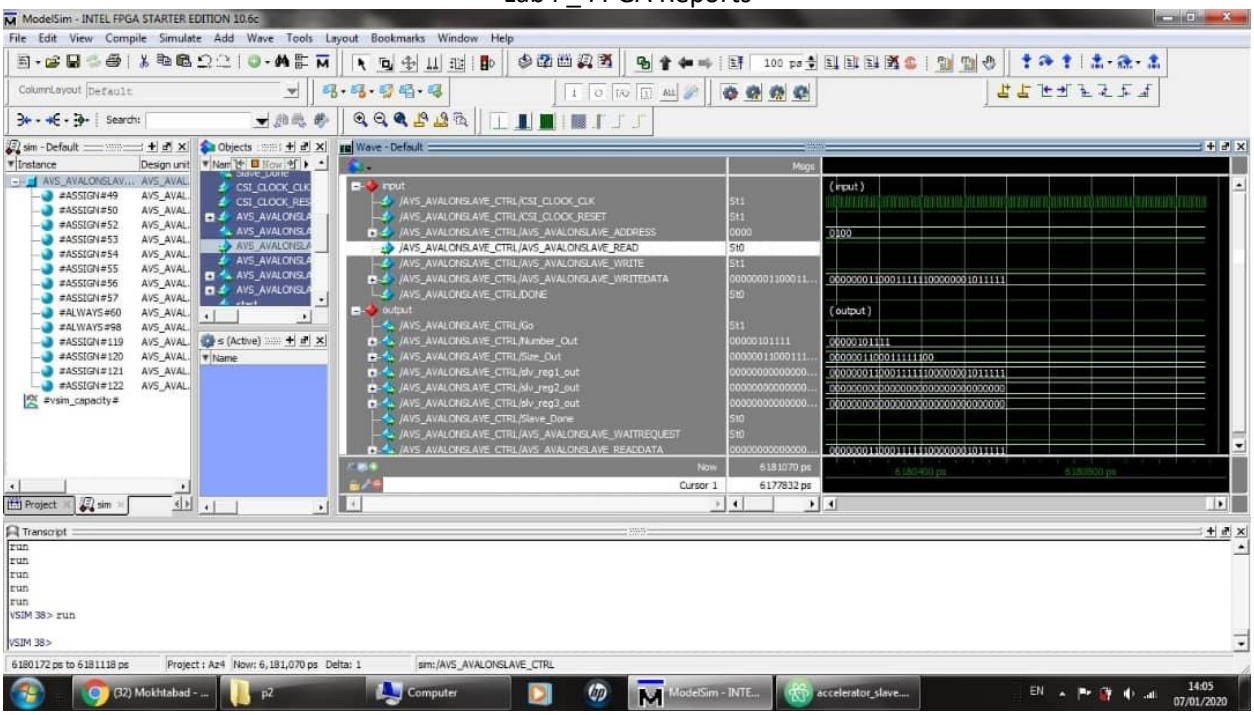




Anita Ghandehari
810195533

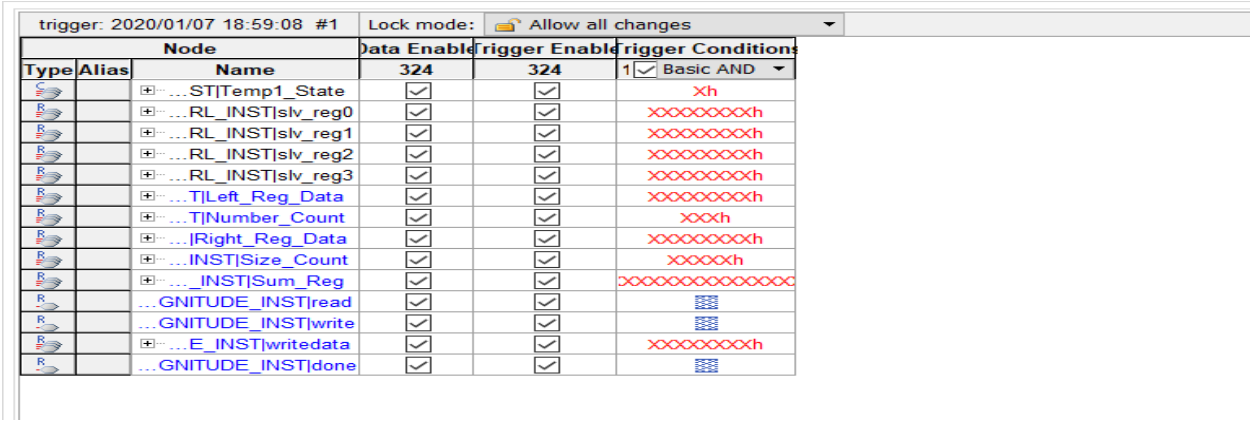
Soheil Shirvani
810195416

Lab4 _ FPGA Reports



که ان ها نشان می دهند قسمت Slave و قسمت Master هر دو به صورت فانکشنال درست هستند

و در سیگنال تب قسمت Slave ان درست کار می کرده است که داریم:



همه ی کد های سخت افزاری و نرم افزاری به پیوست آمده اند.