

```

module cross_hairs (
    sys_clk,
    areset_n,
    H_Cont,
    V_Cont,
    MRead_DATA1,
    average_H1,
    average_H2,
    average_V1,
    average_V2,
    slave_WRITE
);

input sys_clk;
input areset_n;
input [12:0] H_Cont;
input [12:0] V_Cont;
input MRead_DATA1;
input slave_WRITE;

reg [12:0] sample_size1;
output reg [12:0] average_H1;
output reg [12:0] average_V1;

reg [12:0] sample_size2;
output reg [12:0] average_H2;
output reg [12:0] average_V2;

reg new_frame;
reg newnew_frame;
reg [5:0] biggest_diameter;
reg [5:0] temp_diameter;

always @(posedge sys_clk, negedge areset_n)
begin
    if (!areset_n)
    begin
        sample_size1<=0;
        average_H1<=0;
        average_V1<=0;
        sample_size2<=0;
        average_H2<=0;
        average_V2<=0;
        new_frame<=0;
        newnew_frame<=0;
        biggest_diameter<=0;
        temp_diameter<=0;
    end
    else
    begin
        if (H_Cont<=1 && V_Cont<=1)
        begin
            biggest_diameter <= 10;
            new_frame <= 1;
        end

        if (MRead_DATA1 && slave_WRITE)
        begin
            if (new_frame==1)
            begin
                sample_size1<=0;
                average_H1<=0;
                average_V1<=0;
                sample_size2<=0;
                average_H2<=0;
                average_V2<=0;
            end
        end
    end
end

```

```

        new_frame<=0;
        newnew_frame<=1;
    end
    temp_diameter <= temp_diameter+1;
    if ((H_Cont<average_H1+(10) && H_Cont>average_H1-(10) && V_Cont<average_V1+(10) && V_Cont>average_V1-(10)) ||
(newnew_frame==1))
        begin
            sample_size1 <= sample_size1 + 1;
            average_H1 <= average_H1 + ((H_Cont)/(sample_size1+1)) - ((average_H1)/(sample_size1+1));
            average_V1 <= average_V1 + ((V_Cont)/(sample_size1+1)) - ((average_V1)/(sample_size1+1));
            newnew_frame <= 0;
        end
    else if ((H_Cont<average_H2+(10) && H_Cont>average_H2-(10) && V_Cont<average_V2+(10) && V_Cont>average_V2-(10)) ||
(average_H2==0))
        begin
            sample_size2 <= sample_size2 + 1;
            average_H2 <= average_H2 + ((H_Cont)/(sample_size2+1)) - ((average_H2)/(sample_size2+1));
            average_V2 <= average_V2 + ((V_Cont)/(sample_size2+1)) - ((average_V2)/(sample_size2+1));
        end
    end
    else
        begin
            if (temp_diameter > biggest_diameter) biggest_diameter <= temp_diameter;
            temp_diameter <= 0;
        end
    end
end
endmodule

```

```

module frame_subtraction (
    sys_clk,
    areset_n,
    SRAM_DQ,
    Read_DATA1,
    Read_DATA2,
    WRead_DATA1,
    WRead_DATA2
);

input sys_clk;
input areset_n;
input [15:0] SRAM_DQ;
input [14:0] Read_DATA1;
input [14:0] Read_DATA2;

output reg [14:0] WRead_DATA1;
output reg [14:0] WRead_DATA2;

//SRAM_DQ[15:11] = red[9:5];
//SRAM_DQ[10:6] = blue[9:5];
//SRAM_DQ[5:0] = green[{14:10},{14:14}];

//Read_DATA2[9:0] = red;
//Read_DATA1[9:0] = blue;
//{NRead_DATA1[14:10],NRead_DATA2[14:10]} = green;

always @(*)
begin
    if (SRAM_DQ[15:11]>Read_DATA2[9:5])
        WRead_DATA2[9:0]={SRAM_DQ[15:11],5'b0}-{Read_DATA2[9:5],5'b0};
    else
        WRead_DATA2[9:0]={Read_DATA2[9:5],5'b0}-{SRAM_DQ[15:11],5'b0};

    if (SRAM_DQ[10:6]>Read_DATA1[9:5])
        WRead_DATA1[9:0]={SRAM_DQ[10:6],5'b0}-{Read_DATA1[9:5],5'b0};
    else
        WRead_DATA1[9:0]={Read_DATA1[9:5],5'b0}-{SRAM_DQ[10:6],5'b0};

    if (SRAM_DQ[5:0]>{Read_DATA1[14:10],Read_DATA2[14]})
        {WRead_DATA1[14:10],WRead_DATA2[14:10]}={SRAM_DQ[5:0],4'b0}-{Read_DATA1[14:10],Read_DATA2[14],4'b0};
    else
        {WRead_DATA1[14:10],WRead_DATA2[14:10]}={Read_DATA1[14:10],Read_DATA2[14],4'b0}-{SRAM_DQ[5:0],4'b0};
end
endmodule

```

```
module gpio_1_19 (sys_clk, areset_n, GPIOO, Cont);

input sys_clk;
input areset_n;
output reg GPIOO;
output reg [31:0] Cont;

always @ (posedge sys_clk, negedge areset_n)
begin
    if (!areset_n) begin Cont<=0; GPIOO<=0;end
    else
    begin
        if(Cont!=32'h02faf080)
        begin
            Cont<=Cont+1;
            if (Cont<32'h0180_0000)
                GPIOO<=0;
            else
                GPIOO<=1;
        end
    end
    else
    begin
        Cont<=0;
        GPIOO<=1;
    end
end
end

endmodule
```

```

module USB_slave (sys_clk, areset_n, OTG_CS_N,
OTG_RD_N, OTG_WR_N, OTG_RST_N,
    USB_slave_DONE_read, USB_slave_DONE_write,
USB_slave_READ,
    USB_slave_WRITE);

```

```

input sys_clk;
input areset_n;

```

```

input USB_slave_READ;
input USB_slave_WRITE;

```

```

output reg OTG_CS_N;
output reg OTG_RD_N;
output reg OTG_WR_N;
output reg OTG_RST_N;
output reg USB_slave_DONE_read;
output reg USB_slave_DONE_write;

```

```

reg [7:0] counter;
reg [4:0] Sstate, Snext;

```

```

parameter S0 = 5'b00000,

```

```

    S1 = 5'b00001,
    S2 = 5'b00010,
    S3 = 5'b00011,
    S4 = 5'b00100,
    S5 = 5'b00101,
    S6 = 5'b00110,
    S7 = 5'b00111,
    S8 = 5'b01000,
    S9 = 5'b01001,
    S10 = 5'b01010,
    S11 = 5'b01011,
    S12 = 5'b01100,
    S13 = 5'b01101,
    S14 = 5'b01110,
    S15 = 5'b01111,
    S16 = 5'b10000,
    S17 = 5'b10001,
    S18 = 5'b10010,
    S19 = 5'b10011,
    S20 = 5'b10100,
    S21 = 5'b10101,
    S23 = 5'b10110,
    S24 = 5'b10111;

```

```

always @(*)

```

```

begin

```

```

    case (Sstate)

```

```

        S0: begin

```

```

            OTG_CS_N = 1;
            OTG_RD_N = 1;
            OTG_WR_N = 1;
            USB_slave_DONE_read = 1;
            USB_slave_DONE_write = 1;
            if (USB_slave_WRITE)

```

```

                Snext = S6;
            else if (USB_slave_READ)
                Snext = S1;
            else
                Snext = S0;
        end

```

```

// READ READ READ READ READ READ READ READ READ READ READ READ READ
// READ READ READ READ READ READ READ READ READ READ READ READ READ
// READ READ READ READ READ READ READ READ READ READ READ READ READ
// READ READ READ READ READ READ READ READ READ READ READ READ READ

```

```

S1: begin
    OTG_CS_N = 0;
    OTG_RD_N = 1;
    OTG_WR_N = 1;
    USB_slave_DONE_read = 0;
    USB_slave_DONE_write = 0;
    Snext = S2;
end

```

```

S2: begin
    OTG_CS_N = 0;
    OTG_RD_N = 0;
    OTG_WR_N = 1;
    USB_slave_DONE_read = 0;
    USB_slave_DONE_write = 0;
    Snext = S3;
end

```

```

S3: begin
    OTG_CS_N = 0;
    OTG_RD_N = 0;
    OTG_WR_N = 1;
    USB_slave_DONE_read = 0;
    USB_slave_DONE_write = 0;
    Snext = S4;
end

```

```

S4: begin
    OTG_CS_N = 0;
    OTG_RD_N = 1;
    OTG_WR_N = 1;
    USB_slave_DONE_read = 0;
    USB_slave_DONE_write = 0;
    Snext = S5;
end

```

```

S5: begin
    OTG_CS_N = 1;
    OTG_RD_N = 1;
    OTG_WR_N = 1;
    USB_slave_DONE_read = 0;
    USB_slave_DONE_write = 0;
    if (counter == 8'b0000_1010)
        Snext = S0;
end

```

```

        else
            Snext = S5;
        end

// WRITE WRITE WRITE WRITE WRITE WRITE WRITE WRITE
// WRITE WRITE WRITE WRITE WRITE WRITE WRITE WRITE
// WRITE WRITE
// WRITE WRITE WRITE WRITE WRITE WRITE WRITE WRITE
// WRITE WRITE

```

```

S6:    begin
    OTG_CS_N = 0;
    OTG_RD_N = 1;
    OTG_WR_N = 1;
    USB_slave_DONE_read = 0;
    USB_slave_DONE_write = 0;
    Snext = S7;
end

```

```

S7:    begin
    OTG_CS_N = 0;
    OTG_RD_N = 1;
    OTG_WR_N = 0;
    USB_slave_DONE_read = 0;
    USB_slave_DONE_write = 0;
    Snext = S8;
end

```

```

S8:    begin
    OTG_CS_N = 0;
    OTG_RD_N = 1;
    OTG_WR_N = 0;
    USB_slave_DONE_read = 0;
    USB_slave_DONE_write = 0;
    Snext = S9;
end

```

```

S9:    begin
    OTG_CS_N = 0;
    OTG_RD_N = 1;
    OTG_WR_N = 1;
    USB_slave_DONE_read = 0;
    USB_slave_DONE_write = 0;
    Snext = S10;
end

```

```

S10:   begin
    OTG_CS_N = 1;
    OTG_RD_N = 1;
    OTG_WR_N = 1;
    USB_slave_DONE_read = 0;
    USB_slave_DONE_write = 0;
    if (counter == 8'b0000_1010)
        Snext = S0;
    else
        Snext = S10;
    end
end

```

```

default: begin
    OTG_CS_N = 1;
    OTG_RD_N = 1;
    OTG_WR_N = 1;
    USB_slave_DONE_read = 0;
    USB_slave_DONE_write = 0;
    Snext = S0;
end
endcase
end

always @ (posedge sys_clk, negedge areset_n)
begin
    if (!areset_n) begin Sstate <= S0; end
    else
        begin
            Sstate <= Snext;
            if(Snext==S5 || Snext==S10) counter<=counter+1;
            else counter<=0;
        end
    end
end

endmodule

```

```

module USB_master (sys_clk, areset_n,
USB_slave_DONE_read, USB_slave_DONE_write,
USB_slave_READ,
USB_slave_WRITE, OTG_DATA, OTG_ADDR, SW);

```

```

// first step is writing the first 4Bytes of the 32Byte AV
header to memory addresses.

```

```

// the first 4Bytes == 1234_001f (just a sample) (not
actually the first 4 bytes of av header))

```

```

input sys_clk;
input areset_n;

```

```

input [17:0] SW;
input USB_slave_DONE_read;
input USB_slave_DONE_write;

```

```

output reg USB_slave_READ;
output reg USB_slave_WRITE;
output reg [15:0] OTG_DATA;
output reg [1:0] OTG_ADDR;

```

```

reg [32:0] counter;
reg [2:0] Mstate, Mnext;

```

```

parameter
M1 = 3'b001,
M2 = 3'b010,
M0 = 3'b000;

```

```

always @(*)

```

```

begin
OTG_DATA = 16'h1234;
OTG_ADDR = 2'b10;
case (Mstate)
M0: begin
USB_slave_READ = 0;
USB_slave_WRITE = 0;
if (SW[10])
Mnext = M1;
else if (SW[9])
Mnext = M2;
else
Mnext = M0;
end

```

```

M1: begin
USB_slave_READ = 1;
USB_slave_WRITE = 0;
if (SW[10])
Mnext = M1;
else if (SW[9])
Mnext = M2;
else
Mnext = M0;
end

```

```

M2: begin
USB_slave_READ = 0;
USB_slave_WRITE = 1;
if (SW[10])
Mnext = M1;
else if (SW[9])
Mnext = M2;
else
Mnext = M0;
end

```

```

default:
begin
USB_slave_READ = 0;
USB_slave_WRITE = 0;
Mnext = M0;
end
endcase
end

```

```

always @ (posedge sys_clk, negedge areset_n)
begin
if (!areset_n) Mstate <= M0;
else
begin
Mstate <= Mnext;
//if (Mstate==M1 || Mstate==M2) OTG_ADDR <=
OTG_ADDR+1;
//else OTG_ADDR <= 0;
end
end

```

```

endmodule

```

```
module SRAM_master (sys_clk, areset_n, slave_READ, slave_WRITE, SRAM_ADDR, SW, pulse_time,
    slave_DONE_read, slave_DONE_write, KEY_2, KEY_1, H_Cont, V_Cont);
```

```
`include "VGA_Param.h"    // from VGA_Controller.v
wire  [12:0]    v_mask;    // from VGA_Controller.v
assign v_mask = 13'd0;    // from VGA_Controller.v
```

```
input [17:0] SW;
input sys_clk;
input areset_n;
input slave_DONE_read;
input slave_DONE_write;
input KEY_1;
input KEY_2;
input [12:0] H_Cont;
input [12:0] V_Cont;
input [32:0] pulse_time;
```

```
output reg slave_READ;
output reg slave_WRITE;
output reg [17:0] SRAM_ADDR;
    reg [32:0] PRE_SRAM_ADDR;
    reg [2:0] Mstate, Mnext;
```

```
parameter
    M1 = 3'b001,
    M2 = 3'b010,
    M0 = 3'b000;
```

```
always @(*)
```

```
begin
```

```
    case (Mstate)
```

```
        M0:    begin
```

```
            slave_READ = 0;
            slave_WRITE = 0;
            if (!KEY_1)
                Mnext = M2;
            else
                Mnext = M1;
        end
```

```
        M1:    begin
```

```
            slave_READ = 1;
            slave_WRITE = 0;
            if (SW[5] && slave_DONE_read)
                Mnext = M2;
            else if (pulse_time<=32'h02faf080 && pulse_time>=32'h002c0f080)
                Mnext = M2;
            else if (!KEY_2)
                Mnext = M1;
            else if (!KEY_1)
                Mnext = M2;
            else
                Mnext = M1;
        end
```

```
        M2:    begin
```

```
            slave_READ = 0;
            slave_WRITE = 1;
            if (pulse_time>32'h002c0f080 && pulse_time<32'h02faf080)
                Mnext = M2;
            else if (SW[5] && slave_DONE_write)
                Mnext = M1;
            else if (!KEY_1)
                Mnext = M2;
            else
```



```

        Mnext = M1;
    end

    default: begin
        slave_READ = 0;
        slave_WRITE = 0;
        Mnext = M0;
    end
endcase
end

always @ (posedge sys_clk, negedge areset_n)
begin
    if (!areset_n) SRAM_ADDR <= 0;
    else
        begin
            if (SW[5] && slave_DONE_write)
                begin
                    if ((H_Cont == 13'b0) && (V_Cont == 13'b0))
                        SRAM_ADDR <= 0;
                    else
                        SRAM_ADDR <= SRAM_ADDR + 1;
                    end
                end
            else if (slave_DONE_read || slave_DONE_write)
                if ((H_Cont == 13'b0) && (V_Cont == 13'b0))
                    begin
                        SRAM_ADDR <= 0;
                        PRE_SRAM_ADDR <= 0;
                    end
                else if (PRE_SRAM_ADDR > 32'h03ffff)
                    begin
                        PRE_SRAM_ADDR <= PRE_SRAM_ADDR + 1;
                        SRAM_ADDR <= 0;
                    end
                else if ( H_Cont>=X_START  && H_Cont<X_START+H_SYNC_ACT &&
                        V_Cont>=Y_START+v_mask  && V_Cont<Y_START+V_SYNC_ACT )
                    begin
                        PRE_SRAM_ADDR <= PRE_SRAM_ADDR + 1;
                        SRAM_ADDR <= SRAM_ADDR + 1;
                    end
                end
            end
        end
    end

always @ (posedge sys_clk, negedge areset_n)

    if (!areset_n) Mstate <= M0;
    else Mstate <= Mnext;

endmodule

```

```

module SRAM_slave (sys_clk, areset_n, CE_N, OE_N, WE_N, data_drive,
    slave_DONE_read, slave_DONE_write, slave_READ,
    slave_WRITE);

input sys_clk;
input areset_n;

input slave_READ;
input slave_WRITE;

output reg CE_N;
output reg OE_N;
output reg WE_N;
output reg data_drive;
output reg slave_DONE_read;
output reg slave_DONE_write;

reg [7:0] wait_here;

reg [4:0] Sstate, Snext;

parameter S0 = 5'b00000,
    S1 = 5'b00001,
    S2 = 5'b00010,
    S3 = 5'b00011,
    S4 = 5'b00100,
    S5 = 5'b00101,
    S6 = 5'b00110,
    S7 = 5'b00111,
    S8 = 5'b01000,
    S9 = 5'b10000;

always @(*)
begin
    case (Sstate)
        S0: begin
            CE_N = 1;
            OE_N = 1;
            WE_N = 1;
            data_drive = 0;
            slave_DONE_read = 0;
            slave_DONE_write = 0;
            if (slave_WRITE)
                Snext = S6;
            else if (slave_READ)
                Snext = S1;
            else
                Snext = S0;
            end

// READ READ READ READ READ READ READ READ READ READ READ READ
// READ READ READ READ READ READ READ READ READ READ READ READ
// READ READ READ READ READ READ READ READ READ READ READ READ

        S1: begin
            CE_N = 0;
            OE_N = 1;
            WE_N = 1;
            data_drive = 0;
            slave_DONE_read = 0;
            slave_DONE_write = 0;
            Snext = S2;
            end

        S2: begin

```

```

    CE_N = 0;
    OE_N = 0;
    WE_N = 1;
    data_drive = 0;
    slave_DONE_read = 1;
    slave_DONE_write = 0;
    Snext = S3;
end

S3:    begin
    CE_N = 0;
    OE_N = 1;
    WE_N = 1;
    data_drive = 0;
    slave_DONE_read = 0;
    slave_DONE_write = 0;
    if (slave_WRITE)
        Snext = S6;
    else if (slave_READ)
        Snext = S1;
    else
        Snext = S0;
    end

// WRITE WRITE WRITE WRITE WRITE WRITE WRITE WRITE WRITE WRITE
// WRITE WRITE WRITE WRITE WRITE WRITE WRITE WRITE WRITE WRITE
// WRITE WRITE WRITE WRITE WRITE WRITE WRITE WRITE WRITE WRITE

S6:    begin
    CE_N = 0;
    OE_N = 1;
    WE_N = 1;
    data_drive = 1;
    slave_DONE_read = 0;
    slave_DONE_write = 0;
    Snext = S7;
end

S7:    begin
    CE_N = 0;
    OE_N = 1;
    WE_N = 0;
    data_drive = 1;
    slave_DONE_read = 0;
    slave_DONE_write = 1;
    Snext = S8;
end

S8:    begin
    CE_N = 0;
    OE_N = 1;
    WE_N = 1;
    data_drive = 1;
    slave_DONE_read = 0;
    slave_DONE_write = 0;
    if (slave_WRITE)
        Snext = S6;
    else if (slave_READ)
        Snext = S1;
    else
        Snext = S0;
    end

default: begin
    CE_N = 1;
    OE_N = 1;

```

```
        WE_N = 1;
        data_drive = 0;
        slave_DONE_read = 0;
        slave_DONE_write = 0;
        Snext = S0;
    end
endcase
end

always @ (posedge sys_clk, negedge areset_n)
begin
    if (!areset_n) begin Sstate <= S0; end
    else Sstate <= Snext;
end

endmodule
```