EC 551
Advanced Digital Design with Verilog and FPGAs

Lab 3 – Design Description Document

Image Convolution Using a Systolic Array

By
Aastha Anand
Runal Nair

OBJECTIVES

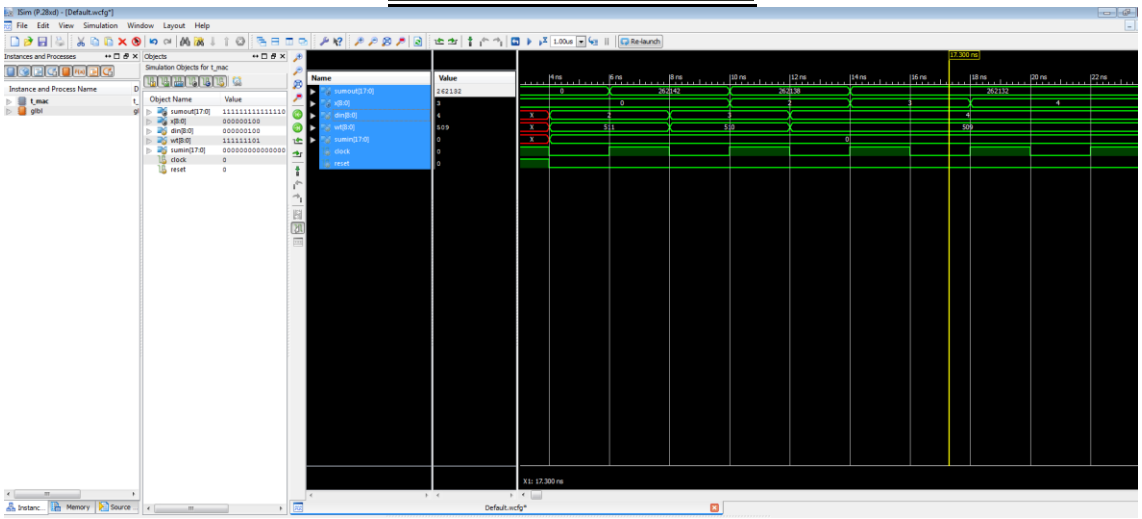
- Practice and learn about systolic arrays.
- Using image processing to work on systolic arrays and convolution.
- Use of a kernel/convolution matrix or a mask which will be used to blur, sharpen, embossing or edge detection and more.
- Implementing convolution between a kernel and an image.

DESIGN DESCRIPTION:

- The design includes the top module which is topmod, this top module is responsible for instantiating all the mac modules which are as mc and the memory module which is memory_ram.
- Each of the mac modules instantiated perform the multiply and accumulation as shown in the code and systolic arrays are thus implemented using them
- The Verilog module, test bench and the waveforms for the mac module is as shown below
- Referred http://www.eecs.harvard.edu/~htk/publication/1984-vlsi-for-pattern-recognition-and-image-processing.pdf

```
module mc(
input wire signed [8:0] din, //the input data value(pixel)
input wire signed [8:0] wt, //The window or weight value
input wire signed [17:0] sumin,
output reg signed [17:0] sumout,
output reg signed [8:0] x,
input wire clock,
input wire reset
);

wire signed [17:0] sum;
reg signed [8:0] temp1=8'd0;

assign sum=(din*wt)+sumin; //Multiplication and accumulation

always@(posedge clock)
begin
    if(reset)
    begin
        sumout<=18'b0;
        temp1<=9'b0;
        x<=9'b0;
    end
    else
    begin
        temp1<=din;
        x<=temp1;
        sumout<=sum;
    end

end

endmodule
```

```
module t_mac;

    reg [8:0] din;
    reg [8:0] wt;
    reg [17:0] sumin;
    reg clock;
    reg reset;

    wire [17:0] sumout;
    wire [8:0] x;

    mc uut (
        .din(din),
        .wt(wt),
        .sumin(sumin),
        .sumout(sumout),
        .x(x),
        .clock(clock),
        .reset(reset)
    );

        always
    begin
        clock=1'b0;
        #2;
        clock=1'b1;
        #2;
    end

    initial begin
        reset = 1;
        #4;
        reset = 0;
        din = 8'h02;
        wt = {1'b1,8'hFF};
        sumin = 8'b0;
        #4;
        din = 8'h03;
        wt = {1'b1,8'hFE};
        sumin = 8'b0;
        #4;
        din = 8'h04;
        wt = {1'b1,8'hFD};
        sumin = 8'b0;
        #4;

        #40;

    end

endmodule
```



PFA THE ENTIRE IMAGE IN ZIP FOLDER

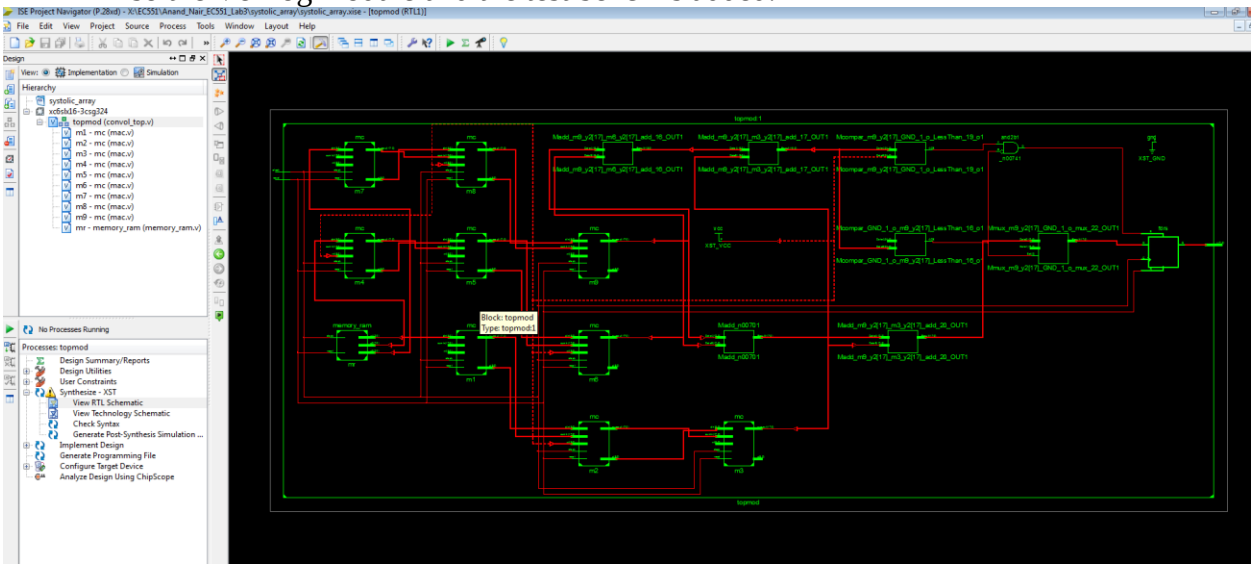
- An integral part of the design makes use of the memory and that module is included as memory_ram
- In the top module topmod, there is requirement of using the memory and thus the memory_ram module is instantiated to get the input of text file

```
mc m1(.din(x1),.wt(w[0]),.sumin(18'b0),.sumout(m1_y0),.x(m2_x2),.clock(clock),.reset(reset));
mc m2(.din(m2_x2),.wt(w[1]),.sumin(m1_y0),.sumout(m2_y1),.x(m3_x3),.clock(clock),.reset(reset));
mc m3(.din(m3_x3),.wt(w[2]),.sumin(m2_y1),.sumout(m3_y2),.x(),.clock(clock),.reset(reset));
mc m4(.din(x2),.wt(w[3]),.sumin(18'b0),.sumout(m4_y0),.x(m5_x2),.clock(clock),.reset(reset));
mc m5(.din(m5_x2),.wt(w[4]),.sumin(m4_y0),.sumout(m5_y1),.x(m6_x3),.clock(clock),.reset(reset));
mc m6(.din(m6_x3),.wt(w[5]),.sumin(m5_y1),.sumout(m6_y2),.x(),.clock(clock),.reset(reset));
mc m7(.din(x3),.wt(w[6]),.sumin(18'b0),.sumout(m7_y0),.x(m8_x2),.clock(clock),.reset(reset));
mc m8(.din(m8_x2),.wt(w[7]),.sumin(m7_y0),.sumout(m8_y1),.x(m8_x3),.clock(clock),.reset(reset));
mc m9(.din(m8_x3),.wt(w[8]),.sumin(m8_y1),.sumout(m9_y2),.x(),.clock(clock),.reset(reset));
memory_ram mr(.clock(clock),.reset(reset),.x1(x1_t),.x2(x2_t),.x3(x3_t));
```

Image shows the instantiating of the memory module after instantiation of the mac modules

```
module memory_ram
(
input wire clock,
input wire reset,
output wire [7:0] x1,
output wire [7:0] x2,
output wire [7:0] x3
);

reg [7:0] memory [9999:0];

initial
begin
    $readmemb("input.txt",memory);
end

reg [31:0] location;
reg [6:0] count;

always@(posedge clock)
begin
    if(reset)
    begin
        location=10'b0;
        count=7'b0;
    end
    else
    begin
        if(location==9799)
        begin
            location=10'b0;
        end
        else
        begin
            if(count>97)
            begin
                location=location+3;
                count=7'b0;
            end
            else
            begin
                location=location+1'b1;
                count=count+1'b1;
            end
        end
    end

end

assign x1=memory[location];
assign x2=memory[location+8'h64];
assign x3=memory[location+8'hC8];


endmodule
```

- One important part to be noticed in the memory_ram part is that which is circled, that is when 97 count is reached it has to go to the next row because of which the location is added with 3
- The top module schematic is as shown below: -
- Also the Verilog module and the test bench is added.

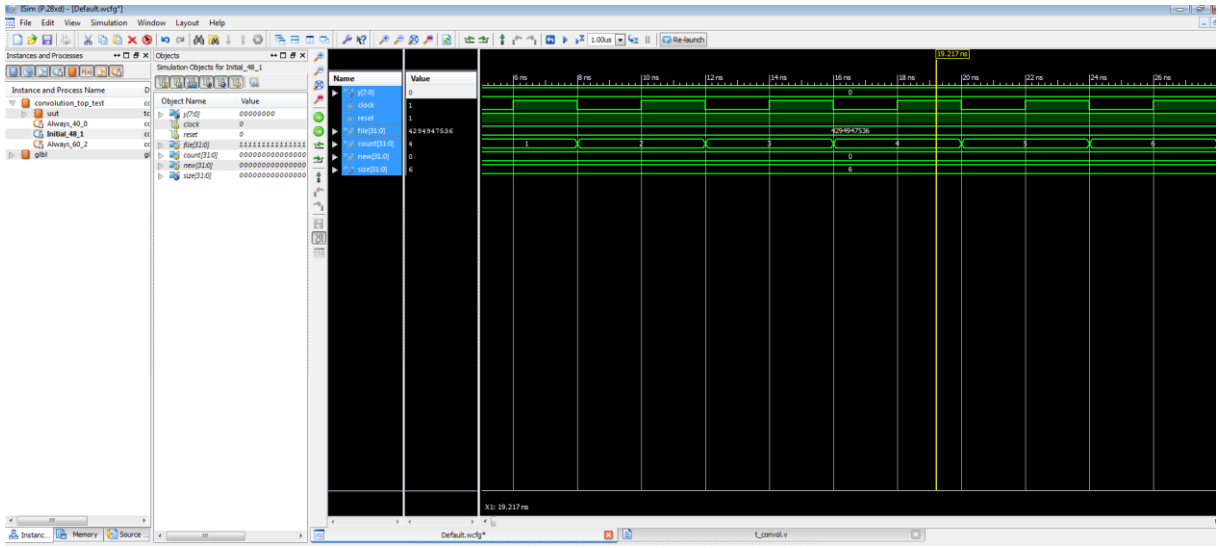


PFA ENTIRE SCHEMATIC IMAGE IN ZIP FOLDER

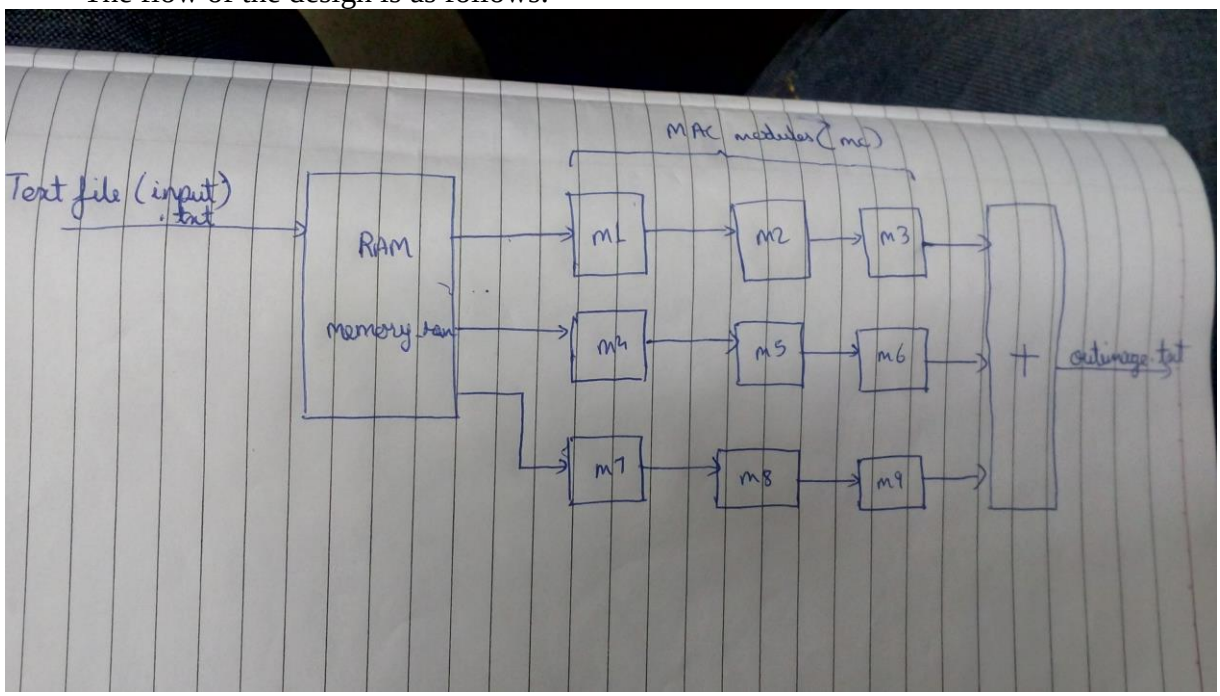
```
module topmod
(
input wire clock,
input wire reset,
output wire [7:0] y
);

wire signed [17:0] m1_y0,m2_y1,m3_y2,m4_y0,m5_y1,m6_y2,m7_y0,m8_y1,m9_y2;
wire signed [8:0] w [8:0];
wire signed [8:0] m2_x2,m3_x3,m5_x2,m6_x3,m8_x2,m8_x3;

wire signed [8:0] x1,x2,x3;
wire [7:0] x1_t,x2_t,x3_t;
reg signed [17:0] y_t;

assign x1={1'b0,x1_t};
assign x2={1'b0,x2_t};
assign x3={1'b0,x3_t};


//window inputs - change for different kernels
assign w[0]={1'b0,8'h00};
assign w[1]={1'b0,8'h01};
assign w[2]={1'b0,8'h00};
assign w[3]={1'b0,8'h01};
assign w[4]={1'b1,8'hFC};
assign w[5]={1'b0,8'h01};
assign w[6]={1'b0,8'h00};
assign w[7]={1'b0,8'h01};
assign w[8]={1'b0,8'h00};

mc m1(.din(x1),.wt(w[0]),.sumin(18'b0),.sumout(m1_y0),.x(m2_x2),.clock(clock),.reset(reset));
mc m2(.din(m2_x2),.wt(w[1]),.sumin(m1_y0),.sumout(m2_y1),.x(m3_x3),.clock(clock),.reset(reset));
mc m3(.din(m3_x3),.wt(w[2]),.sumin(m2_y1),.sumout(m3_y2),.x(),.clock(clock),.reset(reset));
mc m4(.din(x2),.wt(w[3]),.sumin(18'b0),.sumout(m4_y0),.x(m5_x2),.clock(clock),.reset(reset));
mc m5(.din(m5_x2),.wt(w[4]),.sumin(m4_y0),.sumout(m5_y1),.x(m6_x3),.clock(clock),.reset(reset));
mc m6(.din(m6_x3),.wt(w[5]),.sumin(m5_y1),.sumout(m6_y2),.x(),.clock(clock),.reset(reset));
mc m7(.din(x3),.wt(w[6]),.sumin(18'b0),.sumout(m7_y0),.x(m8_x2),.clock(clock),.reset(reset));
mc m8(.din(m8_x2),.wt(w[7]),.sumin(m7_y0),.sumout(m8_y1),.x(m8_x3),.clock(clock),.reset(reset));
mc m9(.din(m8_x3),.wt(w[8]),.sumin(m8_y1),.sumout(m9_y2),.x(),.clock(clock),.reset(reset));
memory_ram mr(.clock(clock),.reset(reset),.x1(x1_t),.x2(x2_t),.x3(x3_t));

always@(posedge clock)
begin
    if(reset)
    begin
        y_t=18'b0;
    end
    else
    begin
        if((m9_y2+m6_y2+m3_y2)<0)
          begin
                y_t=18'b0;
          end
          else if(((m9_y2+m6_y2+m3_y2)>255))
```

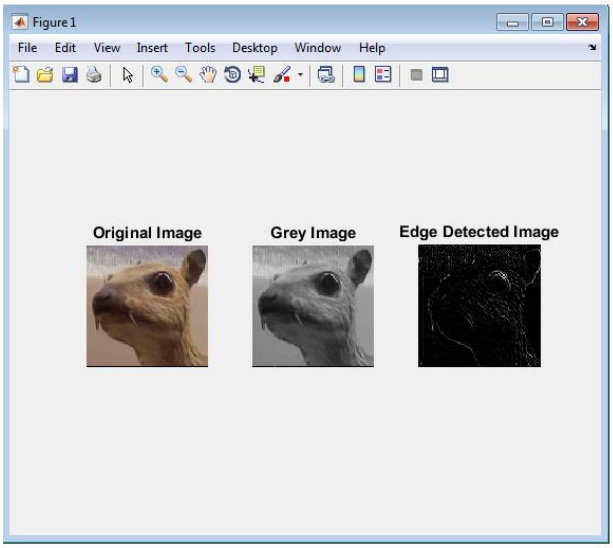
```
always@(posedge clock)
begin
    if(reset)
    begin
        y_t=18'b0;
    end
    else
    begin
        if((m9_y2+m6_y2+m3_y2)<0)
          begin
                y_t=18'b0;
          end
          else if(((m9_y2+m6_y2+m3_y2)>255))
          begin
                y_t=8'b11111111;
          end
          else
          begin
                y_t=m9_y2+m6_y2+m3_y2;
          end
    end
end

assign y=y_t;

endmodule
```

```
module convolution_top_test;

    reg clock;
    reg reset;
    integer file;
    reg [31:0] count;
    reg [31:0] new;
    wire  [7:0] y;
    parameter size=6;

    topmod uut (
        .clock(clock),
        .reset(reset),
        .y(y)
    );

    always
    begin
        clock=1'b0;
        #2;
        clock=1'b1;
        #2;
    end

    initial begin
    reset=1'b1;
    #4;

    count=0;
    new=0;
    file=$fopen("outimage.txt","w");
    #30 reset=1'b0;
    #50000 $finish;

    end

    always @ (negedge clock)
    begin
        count<=count+1;
        if(count>10 && new<9604)
        begin
            file=$fopen("outimage.txt","a");
            $fwrite(file,"%d \n",y);
            new<=new+1;
        end
    $fclose(file);
   end
endmodule
```



The flow of the design is as follows:-





The edge detection output image

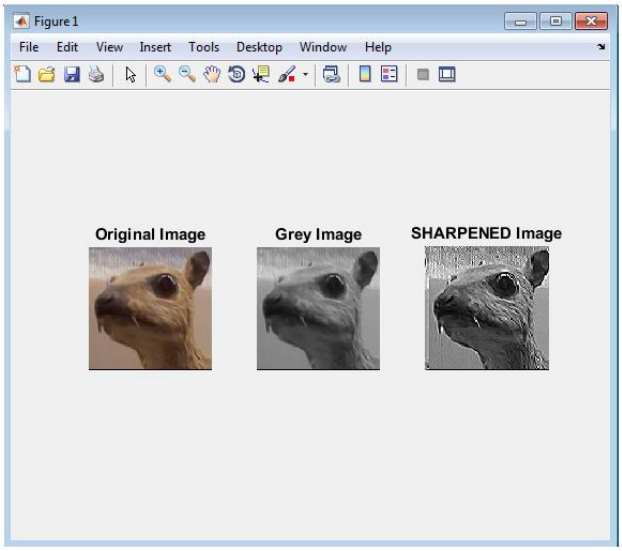
BONUS: -

Aastha Anand & Runal Nair

Generalization of the circuit to do other sized convolutions

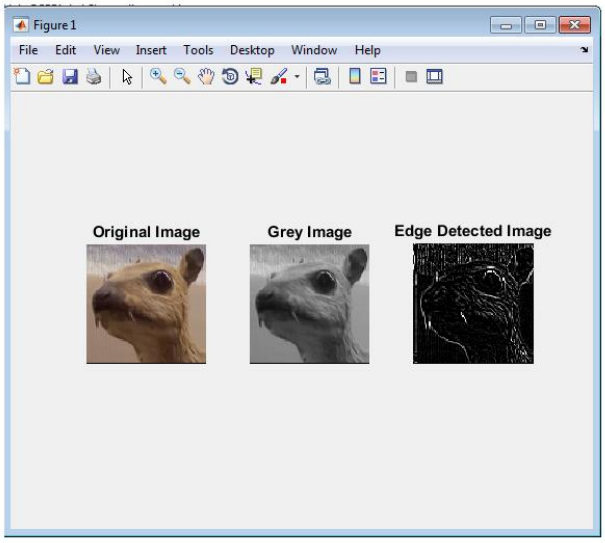
For sharpen:

```
//window inputs - change for different kernels
assign w[0]={1'b0,8'h00};
assign w[1]={1'b1,8'hFF};
assign w[2]={1'b0,8'h00};
assign w[3]={1'b1,8'hFF};
assign w[4]={1'b0,8'h05};
assign w[5]={1'b1,8'hFF};
assign w[6]={1'b0,8'h00};
assign w[7]={1'b1,8'hFF};
assign w[8]={1'b0,8'h00};
```



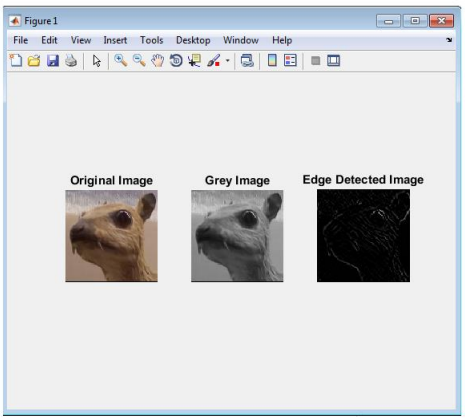
For second type of edge detection

```
//window inputs - change for different kernels
assign w[0]={1'b1,8'hFF};
assign w[1]={1'b1,8'hFF};
assign w[2]={1'b1,8'hFF};
assign w[3]={1'b1,8'hFF};
assign w[4]={1'b0,8'h08};
assign w[5]={1'b1,8'hFF};
assign w[6]={1'b1,8'hFF};
assign w[7]={1'b1,8'hFF};
assign w[8]={1'b1,8'hFF};
```



For the third type: -

```
//window inputs - change for different kernels
assign w[0]={1'b0,8'h01};
assign w[1]={1'b0,8'h00};
assign w[2]={1'b1,8'hFF};
assign w[3]={1'b0,8'h00};
assign w[4]={1'b0,8'h00};
assign w[5]={1'b0,8'h00};
assign w[6]={1'b1,8'hFF};
assign w[7]={1'b0,8'h00};
assign w[8]={1'b0,8'h01};
```



FOR TIMING REPORT PLEASE REFER ATTACHED DOC.