EHB436E

DIGITAL SYSTEM DESIGN APPLICATIONS

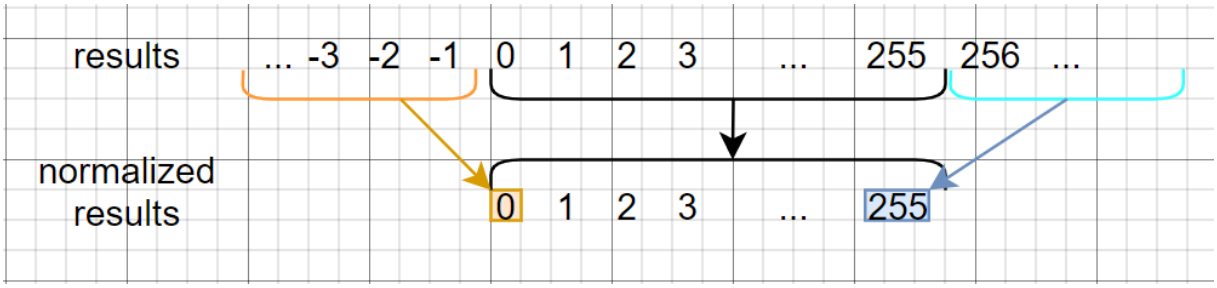
Muhammed Erkmen

040170049

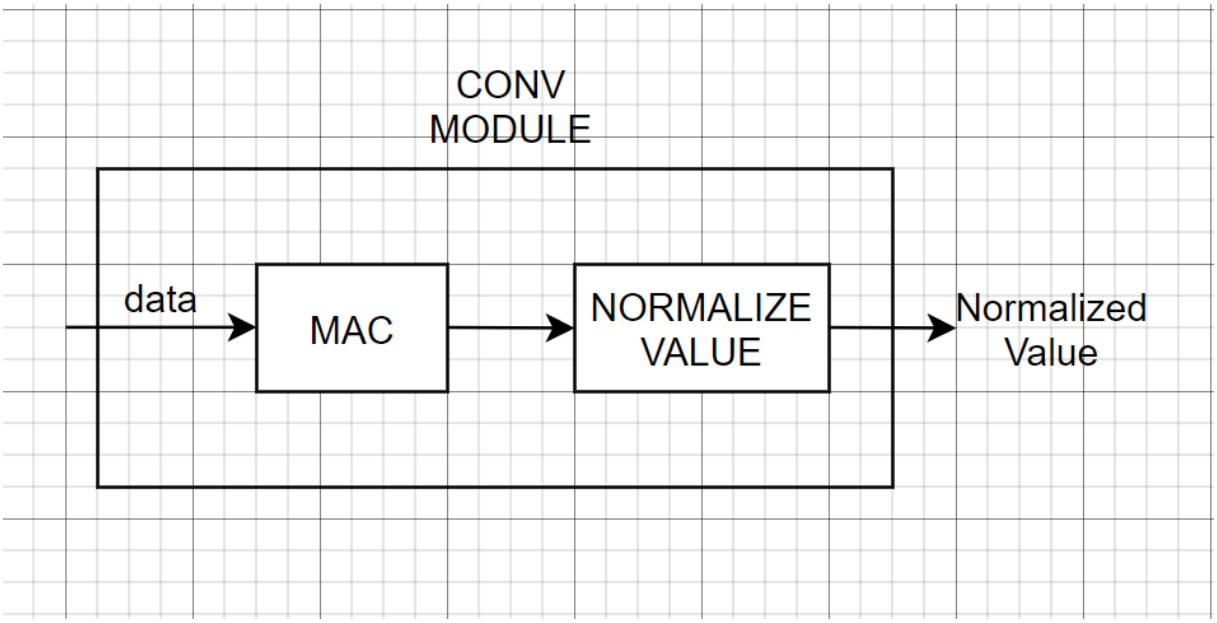
EXPERIMENT-8 REPORT

Convolution Unit

In this stage, i designed my mac again because my last MAC was designed for 8bit-signed inputs, so it was not useful for values more than 127 as an input. I designed it again and mapped output values to [0,255] as if result <0 , newresult =0 & if result>255, newresult=255 & if 0<=result <=255, newresult=result.

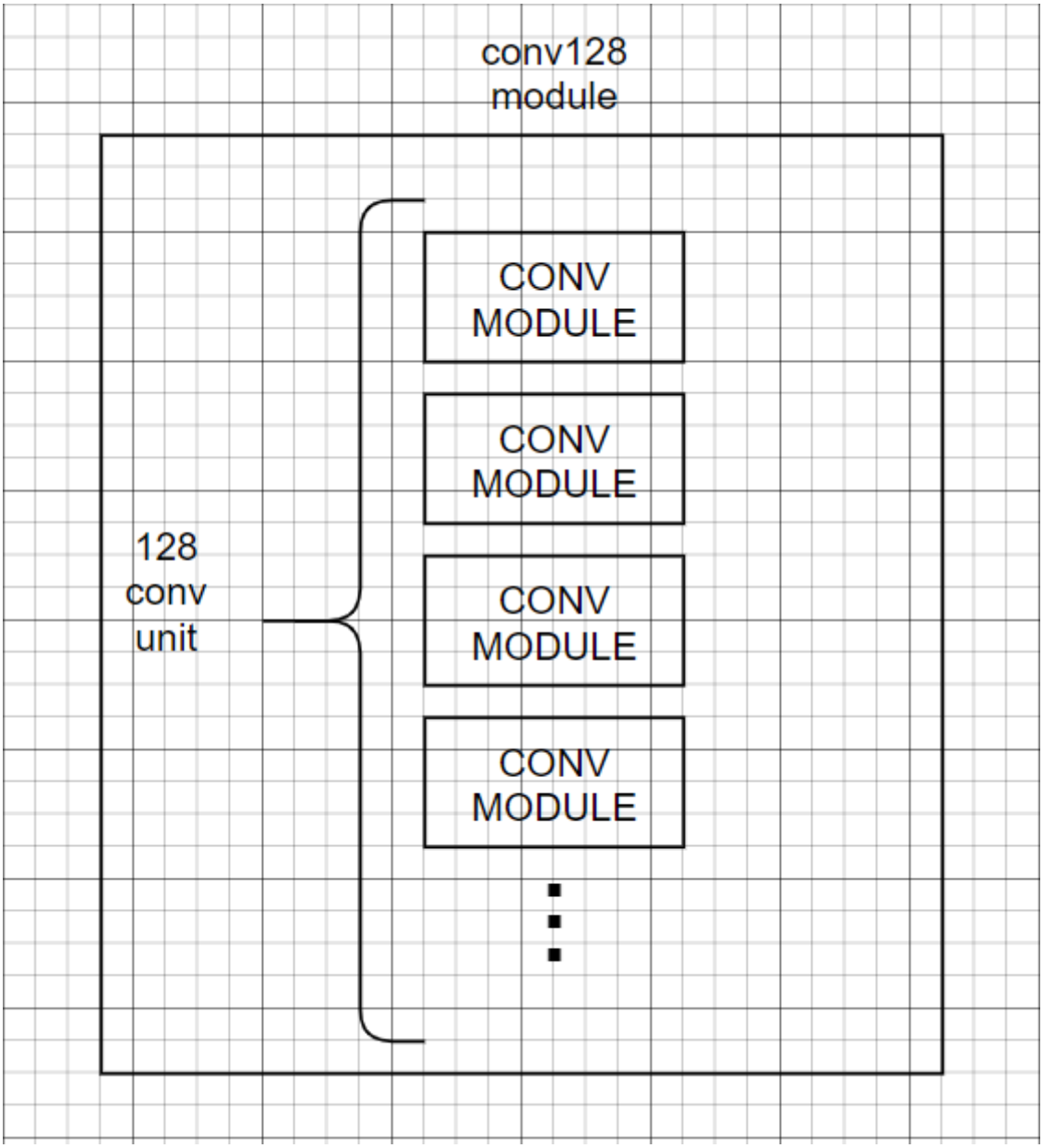


Then i created a module named CONV by using this normalization module:



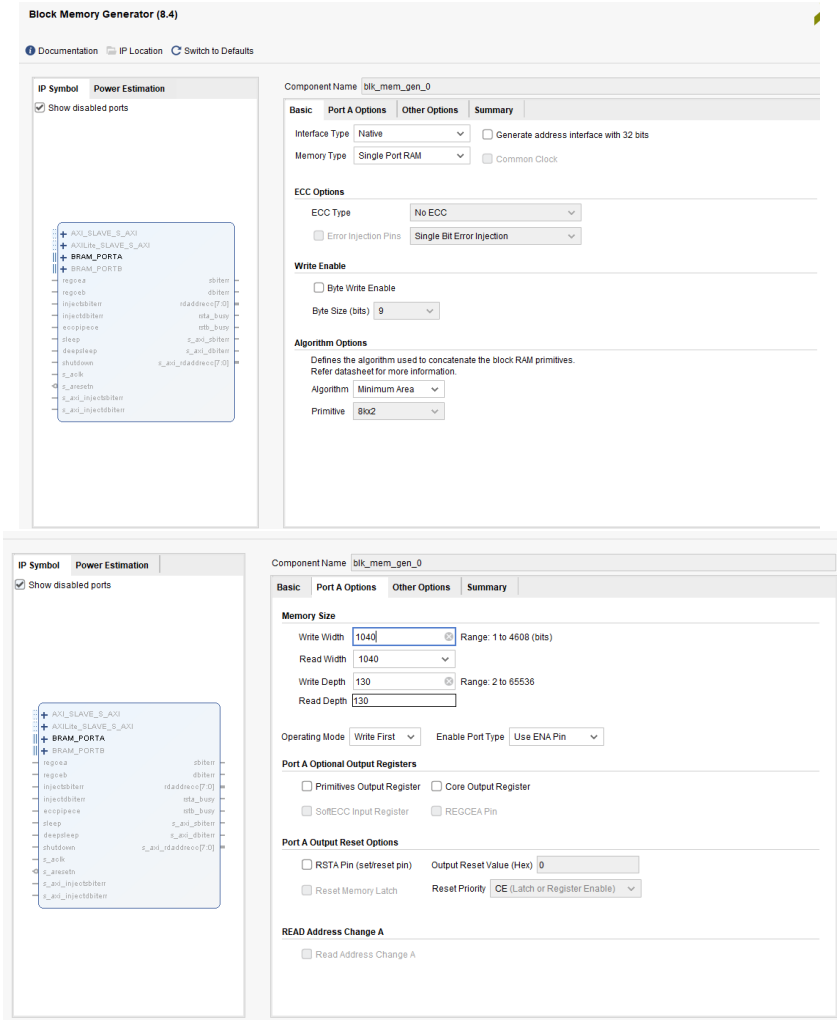
Now i have a convolution module that calculates every 3 clock cycle a convolution. But there is another clock needed to reset MAC. So counter == 0 resets value. Counter=1,2,3 is operation cycles. I controlled this counters by giving resets of mac from Input controller module.

After i created the conv module, i created another module that generates 128 conv modules.That module calculates parallelly a row step of convolution in 3 clock cycles.

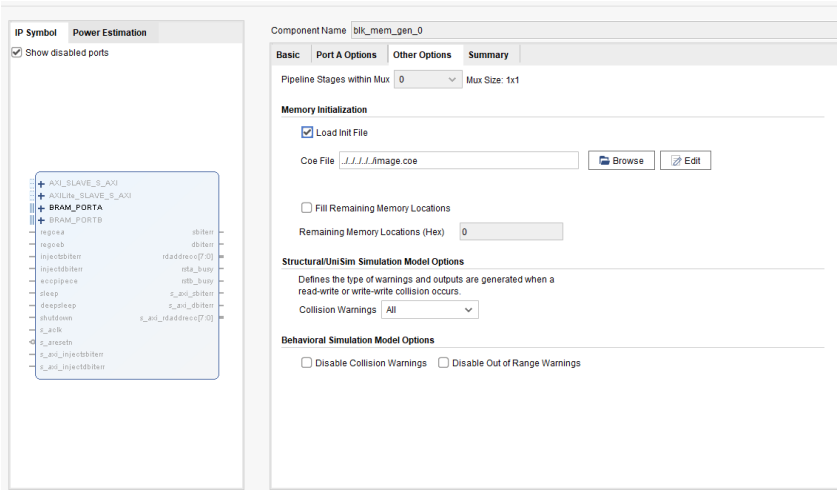


Block Ram#1

I created a 1040 bit data input/output, 130 memory depth Block Ram block from IP Generator. This block ram has 1 clock cycle delay when dataread address given.

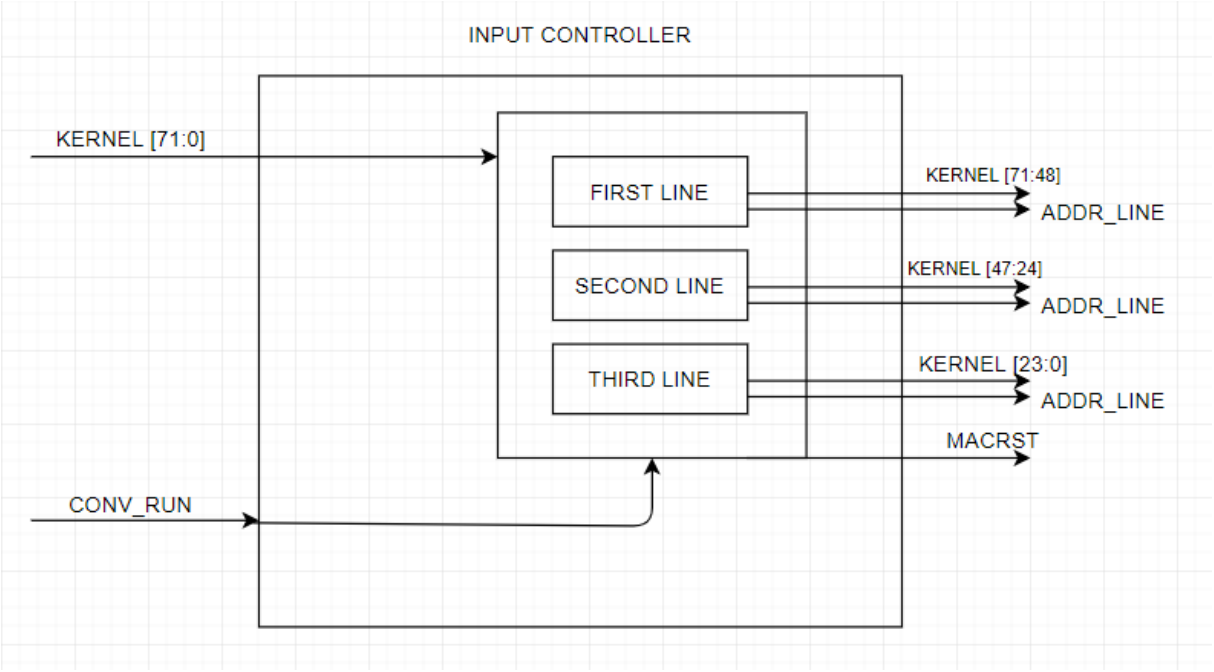


And i loaded the input image coe file to it.

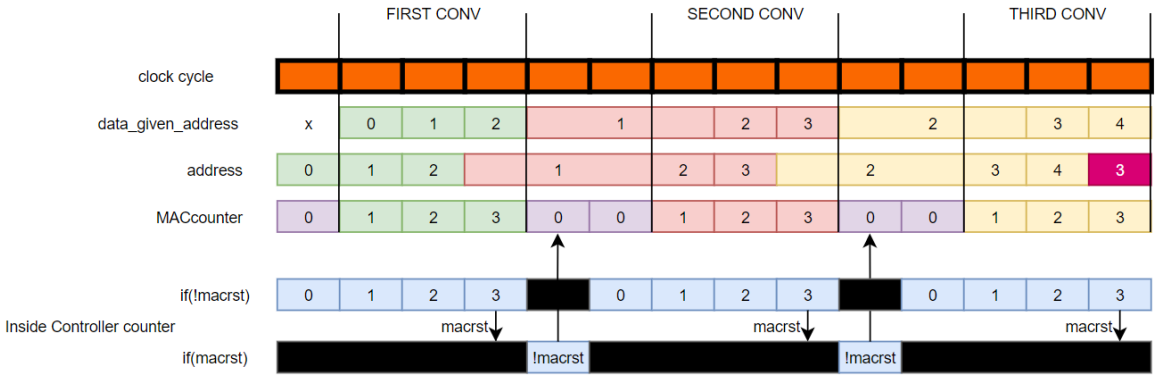


Input Controller

Input controller has 2 inputs as Kernel and Conv_run. When conv_run goes HIGH, convolution operation starts. Because of BRAM1 has 1 clock delay and giving address is another clock delay, so addresses will come in 2 clock delays.



I created an algorithm that MACRST is HIGH initially which resets the MAC units. When CONV_RUN goes HIGH, first clock i enable ram. When enable ram high, in second clock i make MACRST LOW so macs going to work. In third clock i gave the next In my MAC design, MAC calculates operations in counter == 1,2 and 3 so in counter=0 it resets itself again. In third clock i get the data, and MAC counter is 1 right now.



To create this algorithm there is an firstline register inside the input controller. Every inside controller counter == 2, that goes 1 up. That means first line of the operation is shifting when every row operation ends. Thats how everything goes right.

Block RAM #2

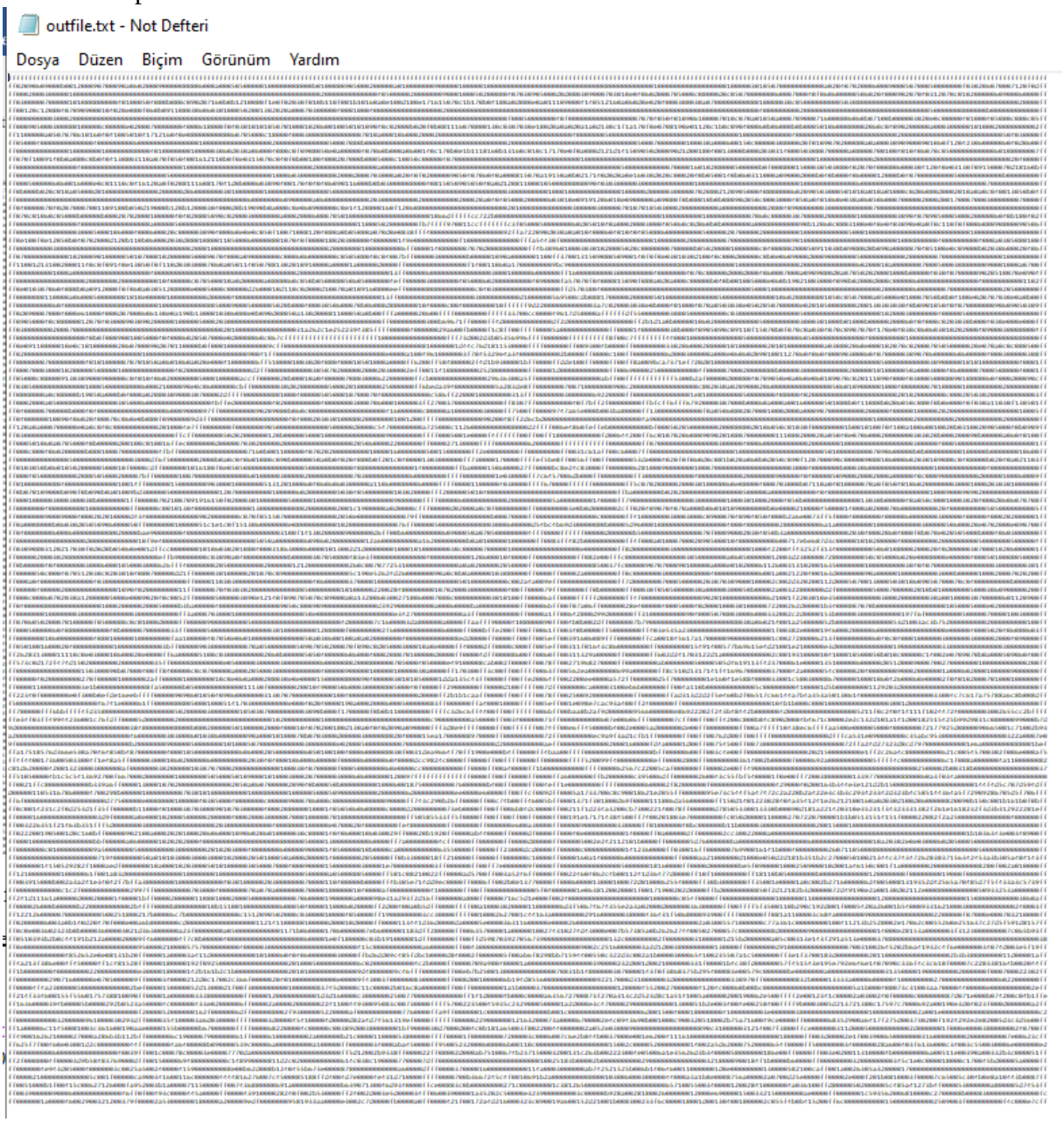
I’ve created a Block Ram Memory block from IP generator. This memory has 1024 bit inputs and outputs. Connected wen to output controller. Depth of this block ram is 128 because our convolution result will has 128 column. This has write priority so, when i give an address, first writes that data, then reads that data from this address.

Output Controller

I’ve created a Block Ram Memory block from IP generator. This memory has 1024 bit inputs and outputs. Connected wen to output controller. Depth of this block ram is 128 because our convolution result will has 128 column. This controller has one bit output conv done and when every calculated convolution result written in the blockram2, this conv done output goes high.

I coded and connected every modules in testbench first. To get the whole data as a clear output to a .txt file. My output.txt file is below.

Textfile output from testbench:

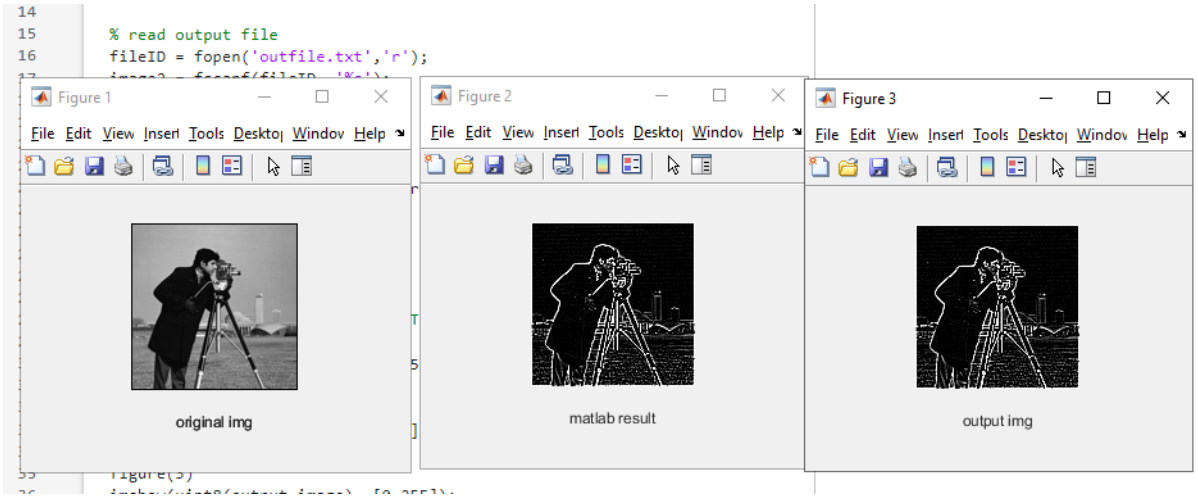


Then i copied this file to the folder where testimage.m exists.

I changed the readen txt code line from this line:

```
% read output file
fileID = fopen('outfile.txt','r');
image2 = fscanf(fileID, '%s');
image2 = upper(image2);
```

Here are results:

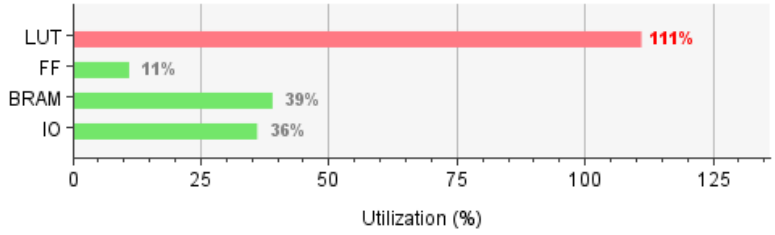


Post Synthesis Reports:

Utilization:

Summary

Resource	Utilization	Available	Utilization %
LUT	36062	32600	110.62
FF	6903	65200	10.59
BRAM	29	75	38.67
IO	76	210	36.19



Timing Summary:

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	-3,025 ns	Worst Hold Slack (WHS):	-1,058 ns	Worst Pulse Width Slack (WPWS):	4,500 ns
Total Negative Slack (TNS):	-5835,989 ns	Total Hold Slack (THS):	-5,792 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	3456	Number of Failing Endpoints:	18	Number of Failing Endpoints:	0
Total Number of Endpoints:	11835	Total Number of Endpoints:	11835	Total Number of Endpoints:	5843

Timing constraints are not met.

Max clk freq = 1/(10+3,025 nS) = 76 MHz

Source codes:

MAC:

```
`timescale 1ns / 1ps

module MAC(
        input [23:0] data,
        input clk,
        input rst,
        input [23:0] weight,
        output reg [19:0] resultout
    );

wire signed [15:0] product0,product1;
wire signed  [16:0] product2;
wire signed [16:0] sum0;
wire signed [17:0] sum1;
assign product2[16] = product2[15]? 1'b1:1'b0;
reg signed [19:0] result;
reg [1:0] count;
MULTB m1 (.A(data[7:0]),.B(weight[7:0]),.result(product0));
MULTB m2 (.A(data[15:8]),.B(weight[15:8]),.result(product1));
MULTB m3 (.A(data[23:16]),.B(weight[23:16]),.result(product2));

ADDB #(16) add1 (.A(product0),.B(product1),.result(sum0));
ADDB #(17) add2 (.A(sum0),.B(product2),.result(sum1));
always @(posedge clk or posedge rst) begin
    if(rst)
    begin
        count <=2'd0;
        result <=20'd0;
    end
    else
        begin
         if (count ==0)
         begin
         result<=0;
         count<=count+1;
         end
         else
         begin
         count<=count+1;
         result<=sum1+result;
         if(count==3)
         resultout<=sum1+result;
         end
        end
    end
endmodule



module ADDB#(parameter SIZE=16)(
    input signed [(SIZE-1):0] A,
    input signed [(SIZE-1):0] B,
    output reg signed [SIZE:0] result
    );

always @(*) begin
```

```
result <= A + B;
end

endmodule



module MULTB(
    input [7:0] A,
    input [7:0] B,
    output reg signed [15:0] result
    );

wire signed [8:0] signextA,signextB;

assign signextA = {1'b0,A};
assign signextB = {B[7],B};

always @(*) begin
result <= signextA * signextB;
end

endmodule
```

MAC normalize:

```
`timescale 1ns / 1ps

module MAC_Normalize(
    input signed [19:0] MACresult,
    output reg [7:0] normalized
    );



always @(*) begin

if(MACresult >= 255)
normalized<=8'd255;
else if (MACresult <=0)
normalized <=8'd0;
else
normalized <= MACresult[7:0];
end
endmodule
```

CONV:

```
`timescale 1ns / 1ps

module CONV(
input clk,
input rst,
input [23:0] data,
input [23:0] weight,
output [7:0] result
    );

wire [19:0] MACresult;
MAC U1 (
          .data(data),
          .clk(clk),
          .rst(rst),
          .weight(weight),
          .resultout(MACresult)
                                    );

MAC_Normalize U2 (.MACresult(MACresult),.normalized(result));

endmodule
```

CONV128:

```
`timescale 1ns / 1ps

module CONV128(
input clk,
input rst,
input [1039:0] data,
input [23:0] weight,
output [1023:0] result
);

genvar i;
generate
for(i=0;i<=127;i=i+1)
begin
CONV convgen (.data(data[23+(8*i):(8*i)]),
              .weight(weight),
              .clk(clk),
              .rst(rst),
              .result(result[7+(8*i):(8*i)]));
end
endgenerate

endmodule
```

Input Controller:

```
`timescale 1ns / 1ps

module InputController(
input clk,
input rst,
input conv_run,
input [71:0] kernel,
output reg macrst=1,
output reg enable_ram,
output reg [7:0] address_ram,
output reg [23:0] weight

    );



reg [7:0] address_ram_fake=0;

reg [7:0] firstline=0;
reg [2:0] counter_line=0;

always @(posedge clk)begin
if(rst)
        begin
            address_ram<=8'h00;
            enable_ram<=0;
        end

    if(conv_run)
        begin
            if(!enable_ram)
                begin
                enable_ram<=1;
                end
            else
                begin
                if(!macrst && firstline!=128)
                    begin
                    if(counter_line!=4)
                    begin
                        if(counter_line==0)
                            begin
                            address_ram<=firstline+1;
                            counter_line<=counter_line+1;
                            weight <= kernel[71:48];
                            end
                        if(counter_line==1)
                            begin
                            address_ram<=firstline+2;
                            counter_line<=counter_line+1;
                            weight <= kernel[47:24];
                            end
                        if(counter_line==2)
                            begin
                            address_ram<=firstline+1;
                            counter_line<=counter_line+1;
                            firstline<=firstline+1;
                            weight<=kernel[23:0];
                            end
```

```
                        if(counter_line==3)
                            macrst<=1;
                    end
                    else
                        begin

                        end
                end
                else
                begin
                macrst<=0;
                counter_line<=0;
                address_ram<=firstline;
                enable_ram<=0;
                end

                end
        end
    else;
end

endmodule
```

Output Controller:

```
module output_control(
    input clk,
    input rst,
    input [1023:0] data,
    output reg [1023:0] dataout,
    output reg conv_done,
    output reg [7:0] bram2_address,
    output reg wen
    );
reg [7:0] counter=0;
always @(data,rst) begin
    if(rst)
        begin
        dataout<=0;
        counter<=0;
        end
    else
    begin
        if(counter <=128)
            begin
            wen<=1'b1;
            dataout<=data;
            bram2_address<=counter;
            counter<=counter+1;
            end
        else
            begin
            conv_done <=1;
            wen<=1'b0;
            end
        end
    end
endmodule
```

Top Module:

```
`timescale 1ns / 1ps

module TOPMODULE(
    input clk,
    input rst,
    input conv_run,
    input [71:0] kernel,
    output conv_done
    );
wire enable_ram;
wire [7:0] address_ram;
wire [23:0] weight;
wire [1039:0] data_out;
wire [1023:0] result;
wire [1023:0] datatomem;
wire wen2;
wire [7:0] bram2_address;
wire macrst;
wire [1023:0] resultbram2;

InputController I (
                   .clk(clk),
                   .rst(rst),
                   .conv_run(conv_run),
                   .kernel(kernel),
                   .enable_ram(enable_ram),
                   .address_ram(address_ram),
                   .weight(weight),
                   .macrst(macrst)
                                 );

blk_mem_gen_0 BRAM1 (
                     .addra(address_ram),
                     .clka(clk),
                     .dina(),
                     .douta(data_out),
                     .ena(enable_ram),
                     .wea(0)
                           );


CONV128  convmod (
                  .clk(clk),
                  .rst(macrst),
                  .data(data_out),
                  .weight(weight),
                  .result(result)
                               );

output_control OUTCONTR (
                         .clk(clk),
                         .rst(rst),
                         .data(result),
                         .dataout(datatomem),
                         .conv_done(conv_done),
                         .bram2_address(bram2_address),
                         .wen(wen2));

blk_mem_gen_1 BRAM2 (
                     .addra(bram2_address),
                     .clka(clk),
```

```
                        .dina(datatomem),
                        .douta(resultbram2),
                        .ena(1),
                        .wea(wen2)
                               );


endmodule
```

When we want to simulate, just disable the top module.v file. Every other module is changed, they are not same as experiment 7 modules.

Testbench Code:

```
`timescale 1ns / 1ps

module TOPMODULE_tb;


reg clk=0;
reg rst;
reg conv_run;
reg [71:0] kernel;

wire enable_ram;
wire [7:0] address_ram;
wire [23:0] weight;
wire [1039:0] data_out;
wire [1023:0] result;
wire [1023:0] datatomem;
wire wen2;
wire [7:0] bram2_address;
wire macrst;

InputController I (
                    .clk(clk),
                    .rst(rst),
                    .conv_run(conv_run),
                    .kernel(kernel),
                    .enable_ram(enable_ram),
                    .address_ram(address_ram),
                    .weight(weight),
                    .macrst(macrst)
                                 );

blk_mem_gen_0 BRAM1 (
                      .addra(address_ram),
                      .clka(clk),
                      .dina(),
                      .douta(data_out),
                      .ena(enable_ram),
                      .wea(0)
                             );
```

```
CONV128  convmod (
                    .clk(clk),
                    .rst(macrst),
                    .data(data_out),
                    .weight(weight),
                    .result(result)
                                );

output_control OUTCONTR (
                          .clk(clk),
                          .rst(rst),
                          .data(result),
                          .dataout(datatomem),
                          .conv_done(conv_done),
                          .bram2_address(bram2_address),
                          .wen(wen2));
wire [1023:0] resultbram2;
blk_mem_gen_1 BRAM2 (
                      .addra(bram2_address),
                      .clka(clk),
                      .dina(datatomem),
                      .douta(resultbram2),
                      .ena(1),
                      .wea(wen2)
                             );
integer n;

initial n = $fopen("outfile.txt","w");

always begin
#10;
clk = ~clk;
end
integer i=0;

always @(bram2_address) begin
if(BRAM2.addra < 128 & BRAM2.douta !=0)
    begin
        $fdisplay(n, "%h", BRAM2.douta);
    end

end


initial begin
rst =1;
#200;
rst=0;
conv_run=0;
#200;
kernel =
72'b11111111__11111111__11111111_11111111__0000_1000__11111111_11111111__11111111_
_11111111;
#100;
conv_run =1;
#2000000;
$fclose(n);
end

endmodule
```