DIGITAL SYSTEM DESIGN APPLICATIONS

Experiment 8

Image Processing System

Berfin Duman
040190108

Contents

1 Code Description

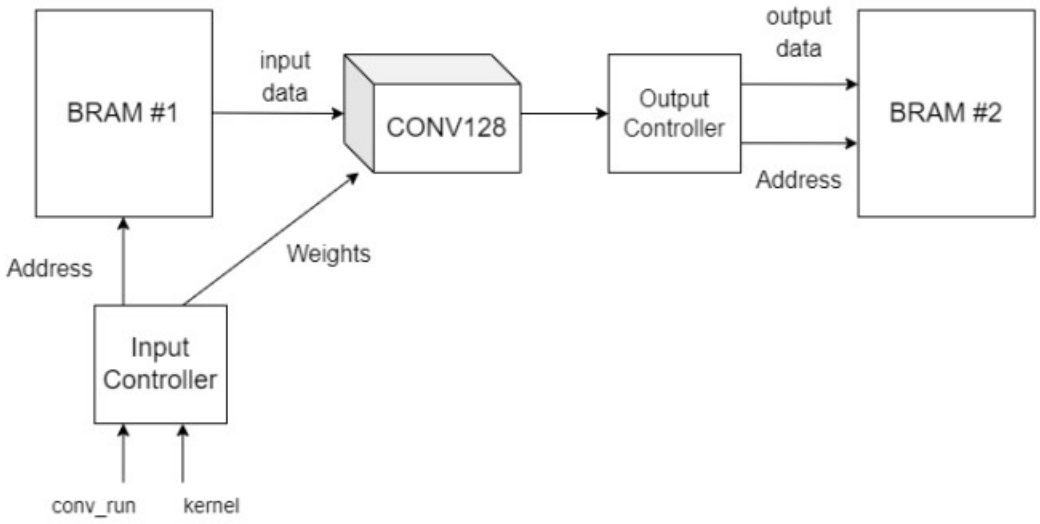


Figure 1: Top level schematic of the image processing system.

Figure 1: image processing system

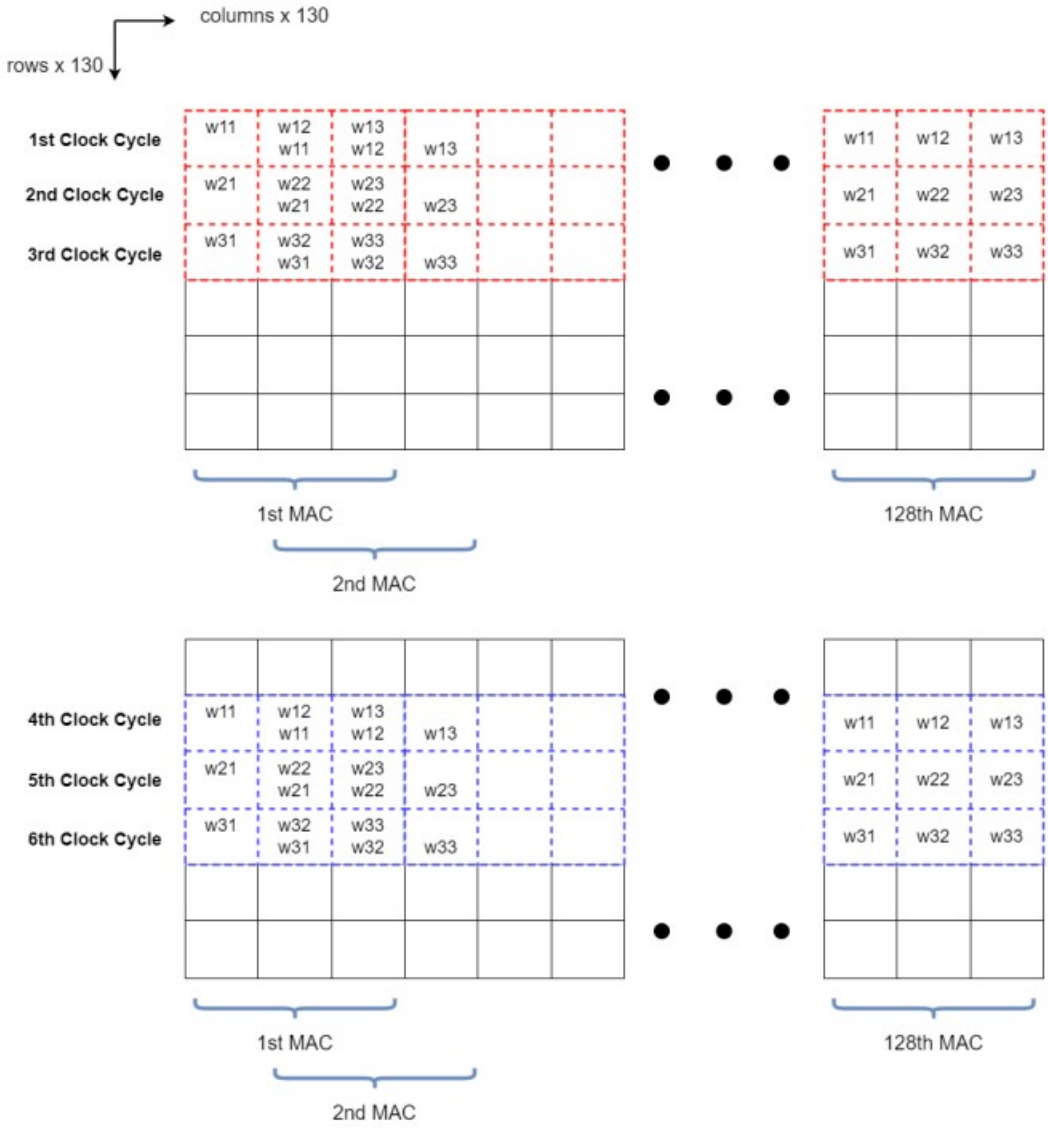


Figure 2: parallel convolution

Before creating the Bram diagram, I wrote the RTL modules that I would use in the image processing system. These:

- input_controller
- CONV128
- output_controller

Input_Controller: It takes Conv_run and kernel parameters, sends the address variable to BRAM 1 and the relevant weight to the Conv128 module. Frankly, this is an important module and its algorithm must be well designed. Here, we performed the addressing and related weighting process using the counter counter. When counter is at 00, weight is -1,-1,-1; when counter is at 01, weight is -1.8-1; When counter is at 10, weight is -1,-1,-1;

The addressing process proceeds line by line as 0,1,2,1,2,3,2,3,4 as shown in the parallel convolution image by keeping temperature as an adr variable.
BRAM1: It is the photo that has not undergone convolution.
BRAM2: It is the version of the photo created after convolution.
The Conv128 module first assigns the data it receives from BRAM to 24-bit values, and this assignment process also takes place in accordance with parallel convolution. It puts these 24 data into 128 convolution calculations in parallel, using the weights from the control module. It transmits the output to the output controls.
While the output controls write to BRAM2 again, it fills BRAM2 by using the counter counter to print one line after every three clocks. I also printed the testbench results to the file named output.txt while BRAM2 was filling up.

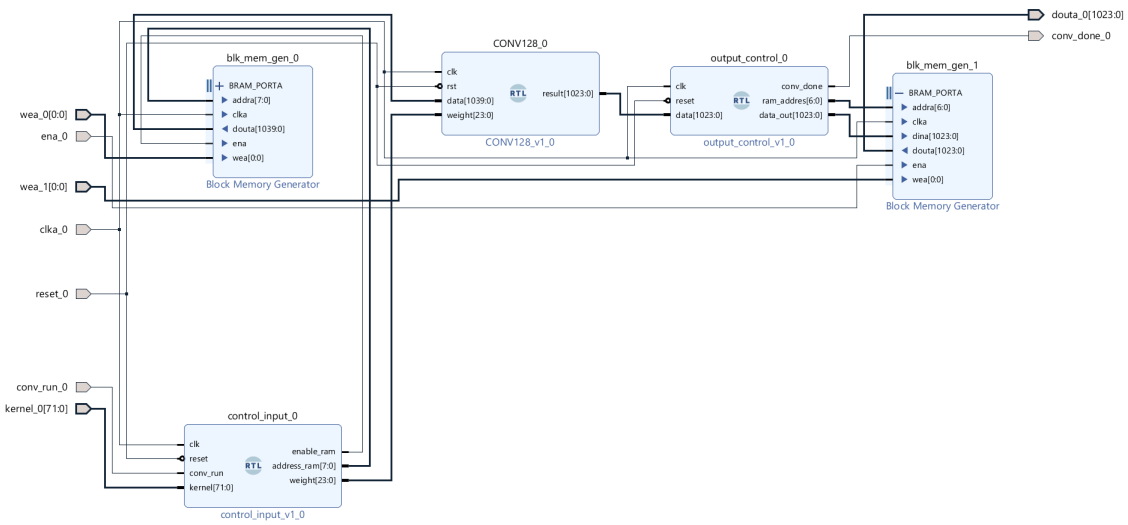


Figure 3: Block Design Diagram

2 Codes

Listing 1: mults module

```
‘timescale 1ns / 1ps
‘timescale 1ns / 1ps

module multb(
input signed [8:0] A,
input signed [8:0] B,
output reg signed [17:0] result
);
always @(*)
begin
     result <= A & B ;
end
```

```
endmodule


module adderb(
input signed [17:0] A,
input signed [17:0] B,
output reg signed [17:0] result
);
always @(*)
begin
     result <= A + B ;
end
endmodule

module MAC(
input clk, rst,
input signed [23:0] data,
input signed [26:0] weight,
output reg signed [19:0] result);
wire signed [17:0] product [2:0];
wire signed [17:0] sum[1:0];
multb multb1(.A($signed({1'b0,data[7:0]})), .B(weight[8:0]),
   .result(product[0]));
multb multb2(.A($signed({1'b0,data[15:8]})), .B(weight[17:9]),
   .result(product[1]));
multb multb3(.A($signed({1'b0,data[23:16]})),
   .B(weight[26:18]), .result(product[2]));
adderb adderb1(.A(product[0]),.B(product[1]),.result(sum[0]));
adderb adderb2(.A(product[2]),.B(sum[0]),.result(sum[1]));

reg [1:0] count;
always @(posedge clk or posedge rst )
        if (rst) begin
            result <= 20'd0;
            count <= 2'b00;
        end

        else begin
            if (count==2'b00) begin
                count <= count + 1;
                result =sum[1];
                //result_temp =sum[1];
            end
            else if (count==2'b01) begin
                count <= count + 1;
```

```
                //result_temp <= result_temp + sum[1];
                //result= result_temp + sum[1];
                result= result + sum[1];
            end
            else if (count == 2'b10) begin
                count <= 2'b00;
                result = result + sum[1];
            end
        end
endmodule

module MAC_Normalize(
input signed [19:0] data,
output  reg  [7:0] result
);
  always @(*)
  begin
    // Normalize the 20-bit data to an 8-bit result
    if (data>255)        result <= 8'b11111111;  // Map values greater than 255 to 255
    else if (data<0)     result <= 8'b0;    // Map values less than 0 to 0
    else                   result <= data[7:0];  // Map values within [0, 255] directly
  end
endmodule

module CONV(
  input clk, rst,
  input signed [23:0] data,
  input signed [26:0] weight,
  output  [7:0] result
);
wire signed [19:0] temp_res;
MAC mac1 (.clk(clk), .rst(rst), .data(data), .weight(weight), .result(temp_res));
MAC_Normalize mac_norm (.data(temp_res), .result(result));

endmodule

module CONV128(
  input clk, rst,
  input signed [1039:0] data ,
  input signed [26:0] weight,
  output [1023:0] result ); //reg
```

```
96  genvar i;
97  //wire  [7:0] res_temp [127:0];
98  generate
99  for (i=0; i<128; i=i+1)
100 begin
101     CONV conv2 (.clk(clk), .rst(rst),.data(data[(i+3)*8-1:
            i*8]),.weight(weight),.result(result[(i+1)*8-1:i*8]));
102 /*
103     always @(*)
104     if (rst)
105     begin
106     result <= 1024'b0;
107     end
108     else  begin
109     result[(i+1)*8-1:i*8] <= res_temp;
110     end
111 */
112 end
113
114 endgenerate
115 endmodule
116 /*
117 module top_bram1(
118 input ena, clk,
119 input [7:0] address,
120 output [1023:0] data);
121 bram1
    bram_ins(.clka_0(clk),.addra_0(address),.ena_0(ena),.douta_0(data));
122 endmodule
123 */
124 module control_input(
125 input clk, reset,
126 input conv_run,
127 input signed [80:0] kernel,
128 output reg enable_ram,
129 output reg [7:0] address_ram,
130 output reg signed [26:0] weight
131 );
132 reg [7:0] adr;
133 reg [1:0] counter;
134 always @(posedge clk or posedge reset )
135         if (reset) begin
136             address_ram <= 8'b00;
137             enable_ram <= 8'b0;
138             counter <= 2'b00;
```

```
139             weight <= kernel[26:0]; //27b
140         end
141         else begin
142         if (conv_run)
143         begin
144             enable_ram <= 1;
145             case(counter)
146             2'b00:
147             begin
148             counter <= counter + 1;
149             address_ram <= address_ram+ 1;
150             weight <= kernel[53:27];
151             end
152             2'b01:
153             begin
154             counter <= counter + 1;
155             adr<= address_ram;
156             address_ram <= address_ram + 1;
157             weight<= kernel[80:54];
158 
159             end
160             2'b10:
161             begin
162             counter <= 2'b00;
163             address_ram <= adr;
164             weight<=  kernel[26:0];
165 
166             end
167             default: counter=00;
168             endcase
169         end
170         else
171        enable_ram<=0;
172        end
173 endmodule
174 
175 module output_control(
176 input clk, reset,
177 input signed [1023:0] data,
178 output reg conv_done,
179 output reg [6:0] ram_addres,
180 output reg signed [1023:0] data_out);
181 reg [1:0] counter;
182 always @(posedge clk or posedge reset )
183 begin
```

```
        if (reset) begin
            counter <= 2'b00;
            ram_addres <=0;
            data_out <=0;
            conv_done <=0;
        end
        else begin

            counter <= counter + 1;

            if (counter == 2'b10)
            begin
            data_out<=data;
            ram_addres <= ram_addres + 1;
            counter <= 2'b00;
            end
        end
end
always @(posedge clk)
begin
    if (ram_addres==127)
    begin
    conv_done <=1;
    ram_addres <=0;
    counter <= 2'b00;
    data_out <=0;
    end
end
endmodule
```

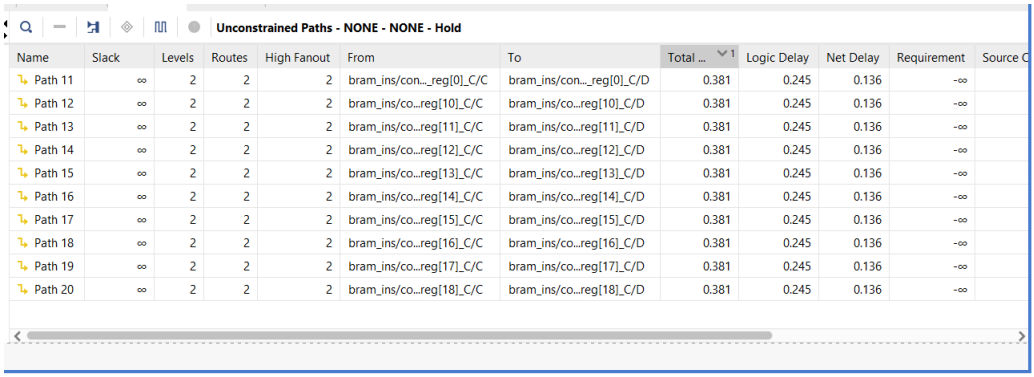
Listing 2: top_tb.v

```
`timescale 1ns / 1ps
`timescale 1ns / 1ps
module TOP(
input clk,rst,conv_run,
output [1023:0] data_end,
output conv_done);

wire signed [7:0] result [127:0];
wire ram_addres;
reg [71:0] kernel;
reg wea_b2;
initial
begin
assign wea_b2=1'b1;
```

```
assign kernel= 72'b11111111_11111111_11111111_11111111_
10000000_11111111_11111111_11111111_11111111;
end

bram1 bram_ins(.ena_0(1'b1),.wea_0(1'b0),
    .wea_1(wea_b2),.clka_0(clk),.reset_0(rst),
    .conv_run_0(conv_run),.kernel_0(kernel),.douta_0(data_end),
    .conv_done_0(conv_done));

always @(posedge conv_done)
if (conv_done==1)
begin
assign wea_b2 = 1'b0;
end

endmodule
```

Listing 3: mults_signed module

```
`timescale 1ns / 1ps


module top_tb();
reg clk;
  reg reset;
  reg conv_run;

  wire conv_done;
  wire [1023:0] data_out;

  TOP top_unit (
    .clk(clk),
    .rst(reset),
    .conv_run(conv_run),
       .conv_done(conv_done),.data_end(data_out));

  initial begin
    #10 reset = 1; clk=1'b1; conv_run = 1;
    #2 reset = 0;

  end

  always #20 clk = ~clk;

endmodule
```

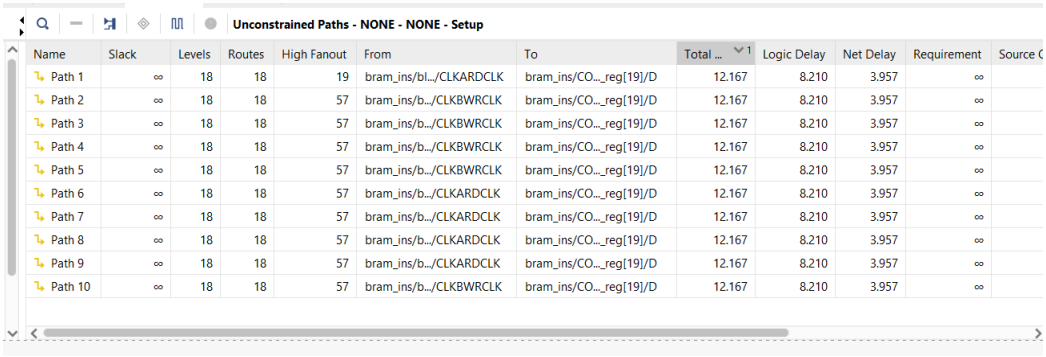


Unconstrained Paths - NONE - NONE - Hold

Name	Slack	Levels	Routes	High Fanout	From	To	Total ...	Logic Delay	Net Delay	Requirement	Source C
Path 11	∞	2	2	2	bram_ins/con..._reg[0]_C/C	bram_ins/con..._reg[0]_C/D	0.381	0.245	0.136	-∞	
Path 12	∞	2	2	2	bram_ins/co...reg[10]_C/C	bram_ins/co...reg[10]_C/D	0.381	0.245	0.136	-∞	
Path 13	∞	2	2	2	bram_ins/co...reg[11]_C/C	bram_ins/co...reg[11]_C/D	0.381	0.245	0.136	-∞	
Path 14	∞	2	2	2	bram_ins/co...reg[12]_C/C	bram_ins/co...reg[12]_C/D	0.381	0.245	0.136	-∞	
Path 15	∞	2	2	2	bram_ins/co...reg[13]_C/C	bram_ins/co...reg[13]_C/D	0.381	0.245	0.136	-∞	
Path 16	∞	2	2	2	bram_ins/co...reg[14]_C/C	bram_ins/co...reg[14]_C/D	0.381	0.245	0.136	-∞	
Path 17	∞	2	2	2	bram_ins/co...reg[15]_C/C	bram_ins/co...reg[15]_C/D	0.381	0.245	0.136	-∞	
Path 18	∞	2	2	2	bram_ins/co...reg[16]_C/C	bram_ins/co...reg[16]_C/D	0.381	0.245	0.136	-∞	
Path 19	∞	2	2	2	bram_ins/co...reg[17]_C/C	bram_ins/co...reg[17]_C/D	0.381	0.245	0.136	-∞	
Path 20	∞	2	2	2	bram_ins/co...reg[18]_C/C	bram_ins/co...reg[18]_C/D	0.381	0.245	0.136	-∞	

Figure 5: Hold Delay

2.1 Timing Report

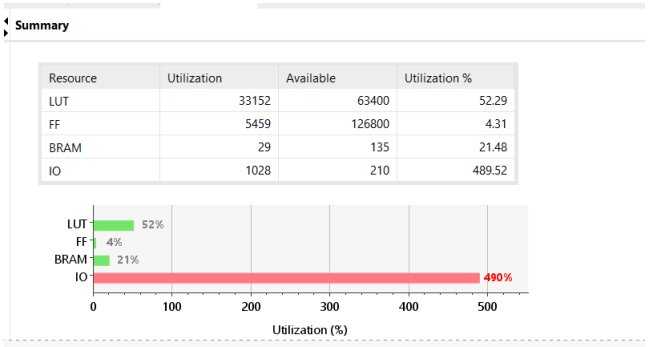


Unconstrained Paths - NONE - NONE - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total ...	Logic Delay	Net Delay	Requirement	Source C
Path 1	∞	18	18	19	bram_ins/bl.../CLKARDCLK	bram_ins/CO..._reg[19]/D	12.167	8.210	3.957	∞	
Path 2	∞	18	18	57	bram_ins/b.../CLKBWRCLK	bram_ins/CO..._reg[19]/D	12.167	8.210	3.957	∞	
Path 3	∞	18	18	57	bram_ins/b.../CLKBWRCLK	bram_ins/CO..._reg[19]/D	12.167	8.210	3.957	∞	
Path 4	∞	18	18	57	bram_ins/b.../CLKBWRCLK	bram_ins/CO..._reg[19]/D	12.167	8.210	3.957	∞	
Path 5	∞	18	18	57	bram_ins/b.../CLKBWRCLK	bram_ins/CO..._reg[19]/D	12.167	8.210	3.957	∞	
Path 6	∞	18	18	57	bram_ins/b.../CLKARDCLK	bram_ins/CO..._reg[19]/D	12.167	8.210	3.957	∞	
Path 7	∞	18	18	57	bram_ins/b.../CLKARDCLK	bram_ins/CO..._reg[19]/D	12.167	8.210	3.957	∞	
Path 8	∞	18	18	57	bram_ins/b.../CLKARDCLK	bram_ins/CO..._reg[19]/D	12.167	8.210	3.957	∞	
Path 9	∞	18	18	57	bram_ins/b.../CLKARDCLK	bram_ins/CO..._reg[19]/D	12.167	8.210	3.957	∞	
Path 10	∞	18	18	57	bram_ins/b.../CLKBWRCLK	bram_ins/CO..._reg[19]/D	12.167	8.210	3.957	∞	

Figure 4: Setup Delay

Delay is equal to 12.167ns and max clock freq is equal to 1/12.167ns=82.218Mhz



Summary

Resource	Utilization	Available	Utilization %
LUT	33152	63400	52.29
FF	5459	126800	4.31
BRAM	29	135	21.48
IO	1028	210	489.52

Figure 6: Utilization Report

2.2 Where We Made Mistakes

When we look at the photos, in the first MATLAB images, contrary to expectations, it seems that the white parts of our output are black and the white parts are white. In the last output in txt format, it is clearly seen that the edges of the result are perceived as

expected. Here, we see that the error is in the multiplication part in the MAC as a result of our analysis. For this purpose, when we test it by assigning 0 to the values after 255 and 255 to the negative values for trial purposes, we can see that the corners are actually found.

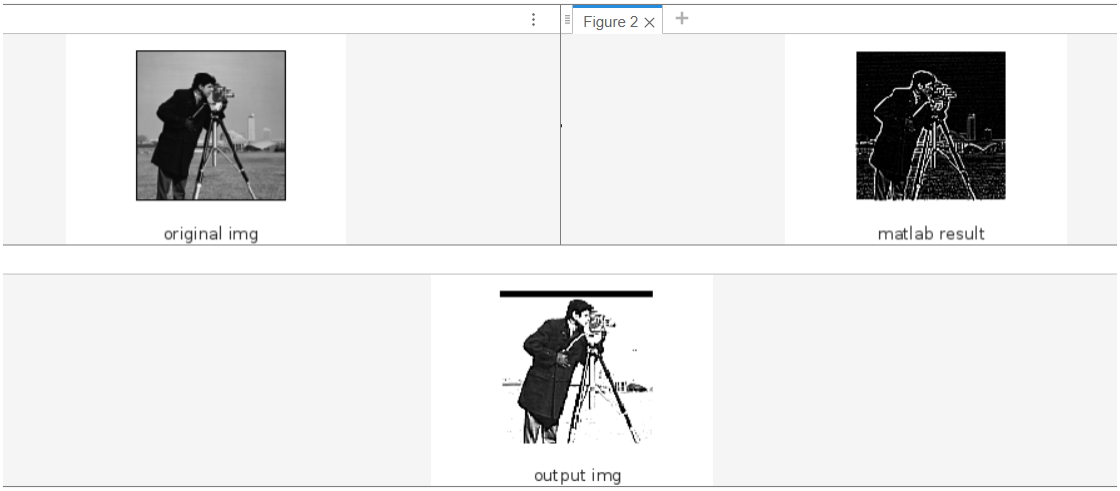


Figure 7: Matlab Results

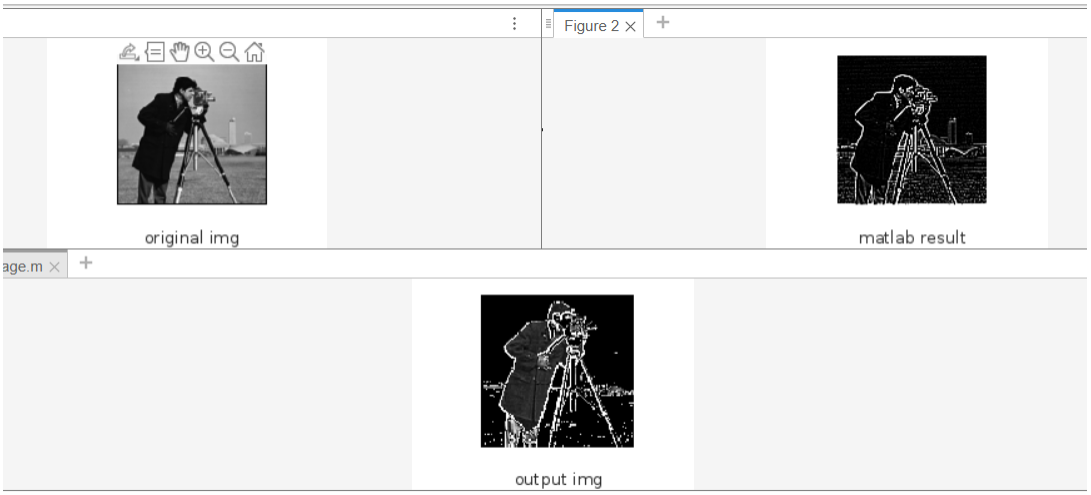


Figure 8: Another Matlab Results

Figure 9: Output IMG

Figure 10: Output IMG TXT Figure