Ömer Faruk SERT

040170244

Verilog Code

Blocks

```
`timescale 1ns / 1ps

module MAC_NORMALIZE(
        input [19:0] data,
        output reg [7:0] result
    );
    always @(*) begin
        if( data < 20'd0 )begin
            result = 8'h00;
        end
        else if( data > 20'd255) begin
            result = 8'b11111111;
        end
        else begin
            result = data[7:0];
        end
    end

endmodule

module CONV(
    input clk,reset,
    input [23:0] data,weight,
    output [7:0] result
);
    wire [19:0] resultp;
        MAC
mac1(.clk(clk),.reset(reset),.data(data),.weight(weight),.result(resultp));
        MAC_NORMALIZE macn1(.data(resultp),.result(result));

endmodule

module CONV128(
    input clk,reset,
    input [1039:0] data,
    input [23:0] weight,
    output [1023:0] result
);
    genvar i;
        generate
            for(i=0;i<128;i=i+1) begin
```

```
                CONV
convi(.clk(clk),.reset(reset),.data(data[23+8*i:8*i]),.weight(weight),.result(res
ult[8*i+7:8*i]));
            end
        endgenerate
endmodule

module control_input(
    input clk,reset,conv_run,
    input [71:0] kernel,
    output reg enable_ram,
    output reg [7:0] address_ram,
    output reg [23:0] weight
);
    integer i;
    reg [1:0] count;
    reg [7:0] addr;
    always @(posedge clk) begin
        if (reset == 1'b1)begin
            addr <= 8'b00000000;
            count <= 2'b00;
        end
        else if(conv_run == 1'b1) begin
            enable_ram <= 1'b1;
                if(count == 2'b00)begin
                    weight <= kernel[23:0];
                    address_ram <= addr;
                    count <= count + 1'b1;
                end
                else if(count == 2'b01) begin
                    weight <= kernel[47:24];
                    address_ram <= addr+1'b1;
                    count <= count + 1'b1;
                end
                else if(count == 2'b10) begin
                    weight <= kernel[71:48];
                    address_ram <= addr+2'b10;
                    count<=2'b00;
                    addr <= addr +1'b1;
                end
            else if(addr == 8'b10000010)begin
                addr <= 8'b00000000;
            end
        end
    end
```

```
endmodule

module output_control(
    input clk,reset,
    input [1023:0] data,
    output reg conv_done,
    output reg [6:0] ram_address,
    output reg [1023:0] data_out
);
    integer i;
    reg [1:0] count;
    reg [6:0] addr;
    reg [1023:0] datap1,datap2,datap3;
    always @(posedge clk | reset) begin
        if (reset == 1'b1) begin
            count=2'b00;
            addr=7'b0000000;
        end
        else if(reset == 1'b0 )begin
                if(count == 2'b00)begin
                    datap1 <= data;
                    count <= 2'b01;
                end
                else if(count == 2'b01)begin
                    datap2 <= data;
                    count <= 2'b10;
                    ram_address <= addr;
                end
                else if(count == 2'b10)begin
                    datap3 <= data;
                    count <= 2'b11;
                end
                else if(count == 2'b11) begin
                    data_out <= datap1+datap2+datap3;
                    addr <= addr+1'b1;
                    count <= 2'b01;
                    datap1 <= data;
                end
            end
    end
endmodule
// her 3 clk de bir output gelmesi lazim
```

Top Module

```
`timescale 1ns / 1ps


module TOP(
    input clk,reset,conv_run,
    input [71:0] kernel,
    output [1023:0] data_outl
    );
    wire [7:0] address_ram;
    wire [23:0] weight;
    wire [1039:0] data;
    wire [1023:0] data2;
    wire wea,ena;
    wire [6:0] ram_address;
    wire [1023:0] data3;
    control_input
ci(.clk(clk),.reset(reset),.conv_run(conv_run),.kernel(kernel),.address_ram(addre
ss_ram),.weight(weight),.enable_ram(ena)); // all connections done
    blk_mem_gen_0
BRAM(.douta(data),.clka(clk),.addra(address_ram),.ena(ena),.wea(1'b0));// all
connections done
    CONV128
CV1281(.weight(weight),.data(data),.clk(clk),.reset(reset),.result(data2)); //
all connections done
    output_control
co(.clk(clk),.reset(reset),.data(data2),.ram_address(ram_address),.data_out(data3
),.conv_done(wea));//all connections done
    blk_mem_gen_1
BRAM2(.dina(data_outl),.clka(clk),.addra(ram_address),.wea(wea),.ena(ena2));
endmodule
```

Designing Submodules

Input Controller

This submodule deals with address of ram that we and also arranges weights to be send by for every clock first it selects the first row of kernel and first row ram address that chosen then second row of kernel and chosen+1 row of ram then third row of kernel and chosen +2 row of ram in the last part it increases the chosen by one and then it does all the process again, in every three clock it finishes convolution of a row.

Conv 128

This part we make parallel convolutions by 128 convolution blocks because in input row there is 130 value and our kernel have 3 value for every row so 128th convolution takes last three value so it can not continue.

Output Controller

For every clock output controller stores the data and every three clock it calculates the value of a row’s convolution and writes it to the memory and increases the address of that will be written also sets it high the write enable input of the BRAM2.

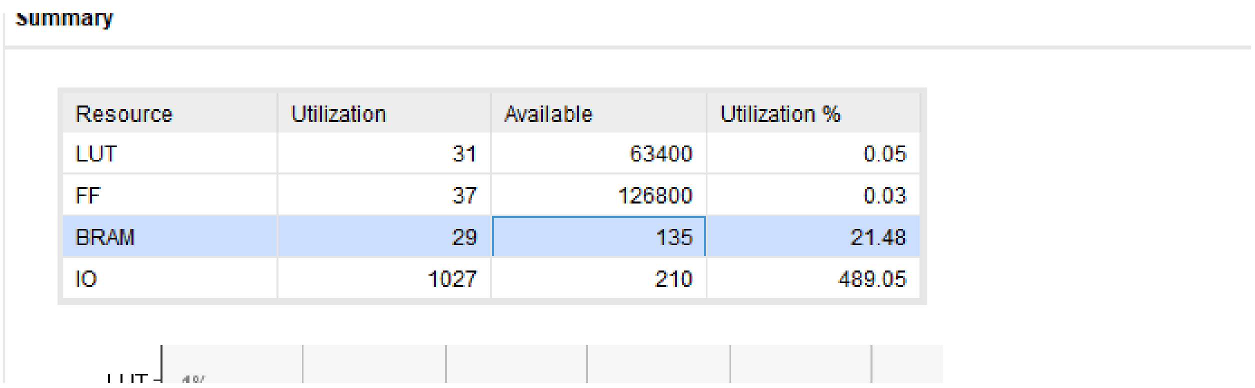
BRAM 1

The input data is stored in this ram

BRAM 2

The output data is stored in this ram.

Utilization Report


Summary

Resource	Utilization	Available	Utilization %
LUT	31	63400	0.05
FF	37	126800	0.03
BRAM	29	135	21.48
IO	1027	210	489.05

Maximum Clock Frequency

There was no combinational delay because my design did not fit in the fpga

Testbench Code

```
`timescale 1ns / 1ps
module tb();
    reg clk,reset,conv_run;
    reg signed [71:0] kernel;
    wire [1023:0] data_outl;

    TOP
uut(.clk(clk),.reset(reset),.kernel(kernel),.data_outl(data_outl),.conv_run(conv_
run));

    integer f;
    integer i;
    integer l;
    integer k;


    initial begin
        k=0;
        l=0;
        f = $fopen("output.txt","w");
        reset = 1'b1;
        clk=1'b1;
        #2
        reset = 1'b0;
        clk=1'b0;
        conv_run=1'b1;
        kernel =
72'b111111111111111111111111111111110000100011111111111111111111111111111111;
        for(i=0;i<410;i=i+1)begin
            clk = ~clk;
            #2;
            l=l+1;
            if(l >19)begin
                if(k!=2)begin
                k=k+1;
            end
            else if(k==2)begin
                $fwrite(f,"%H\n",data_outl);
                k=0;
            end
            end
        end

        $fclose(f);
        $finish;
    end

endmodule
```

MATLAB Screenshots

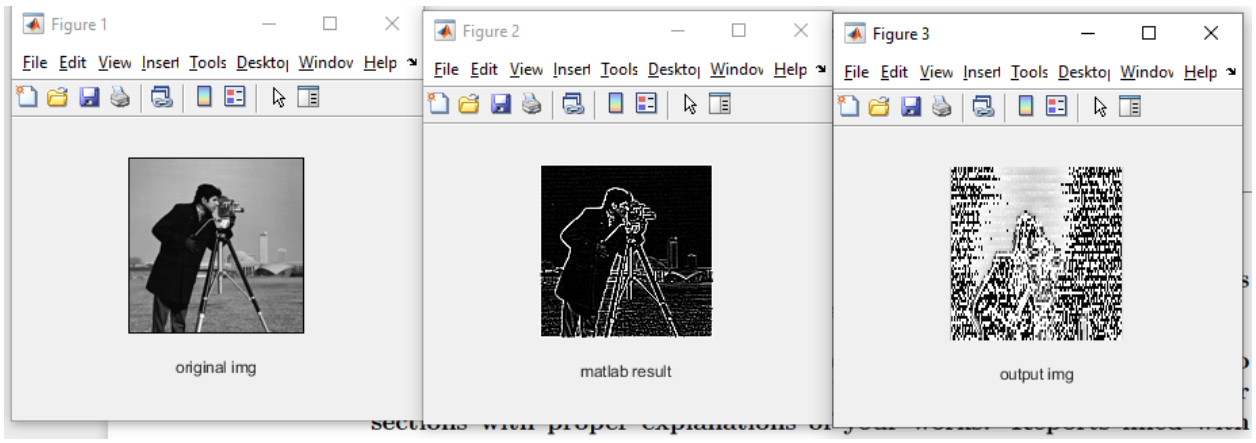


Figure 1
File Edit View Insert Tools Desktop Window Help
original img
Figure 2
File Edit View Insert Tools Desktop Window Help
matlab result
Figure 3
File Edit View Insert Tools Desktop Window Help
output img