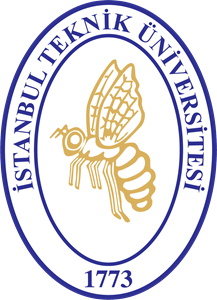
**DIGITAL SYSTEM DESIGN APPLICATIONS**

**(CRN: 11275)**

**THE REPORT OF EXPERIMENT - 8**



Faculty of Electrical and Electronics Engineering

Electronics and Communication Engineering

Yusuf Tekin - 040200043

1. **Conv\_unit**

Table - Convolution Unit Verilog Code

`timescale 1ns **/** 1ps

**module** conv\_unit**(**

**input** pixel\_clk**,**

**input** rst**,**

**input** enable**,**

**input** **[**11**:**0**]** pixel1**,**

**input** **[**11**:**0**]** pixel2**,**

**input** **[**11**:**0**]** pixel3**,**

**input** **[**11**:**0**]** kernel1**,**

**input** **[**11**:**0**]** kernel2**,**

**input** **[**11**:**0**]** kernel3**,**

**output** **reg** **[**3**:**0**]** pixel\_out

**);**

**wire** **signed** **[**8**:**0**]** mult1\_1**,** mult1\_2**,** mult1\_3**,** mult2\_1**,** mult2\_2**,** mult2\_3**,** mult3\_1**,** mult3\_2**,** mult3\_3**;**

**wire** **[**12**:**0**]** product**;**

**wire** **signed** **[**4**:**0**]** pixel11**,** pixel12**,** pixel13**,** pixel21**,** pixel22**,** pixel23**,** pixel31**,** pixel32**,** pixel33**;**

**wire** **signed** **[**3**:**0**]** kernel11**,** kernel12**,** kernel13**,** kernel21**,** kernel22**,** kernel23**,** kernel31**,** kernel32**,** kernel33**;**

**assign** pixel11 **=** **{**1'b0**,** pixel1**[**3**:**0**]};**

**assign** pixel12 **=** **{**1'b0**,** pixel1**[**7**:**4**]};**

**assign** pixel13 **=** **{**1'b0**,** pixel1**[**11**:**8**]};**

**assign** pixel21 **=** **{**1'b0**,** pixel2**[**3**:**0**]};**

**assign** pixel22 **=** **{**1'b0**,** pixel2**[**7**:**4**]};**

**assign** pixel23 **=** **{**1'b0**,** pixel2**[**11**:**8**]};**

**assign** pixel31 **=** **{**1'b0**,** pixel3**[**3**:**0**]};**

**assign** pixel32 **=** **{**1'b0**,** pixel3**[**7**:**4**]};**

**assign** pixel33 **=** **{**1'b0**,** pixel3**[**11**:**8**]};**

**assign** kernel11 **=** kernel1**[**3**:**0**];**

**assign** kernel12 **=** kernel1**[**7**:**4**];**

**assign** kernel13 **=** kernel1**[**11**:**8**];**

**assign** kernel21 **=** kernel2**[**3**:**0**];**

**assign** kernel22 **=** kernel2**[**7**:**4**];**

**assign** kernel23 **=** kernel2**[**11**:**8**];**

**assign** kernel31 **=** kernel3**[**3**:**0**];**

**assign** kernel32 **=** kernel3**[**7**:**4**];**

**assign** kernel33 **=** kernel3**[**11**:**8**];**

**assign** mult1\_1 **=** pixel13 **\*** kernel13**;**

**assign** mult1\_2 **=** pixel12 **\*** kernel12**;**

**assign** mult1\_3 **=** pixel11 **\*** kernel11**;**

**assign** mult2\_1 **=** pixel23 **\*** kernel23**;**

**assign** mult2\_2 **=** pixel22 **\*** kernel22**;**

**assign** mult2\_3 **=** pixel21 **\*** kernel21**;**

**assign** mult3\_1 **=** pixel33 **\*** kernel33**;**

**assign** mult3\_2 **=** pixel32 **\*** kernel32**;**

**assign** mult3\_3 **=** pixel31 **\*** kernel31**;**

**assign** product **=** **(((**mult1\_1 **+** mult1\_2**)** **+** **(**mult1\_3 **+** mult2\_1**))** **+** **((**mult2\_2 **+** mult2\_3**)** **+** **(**mult3\_1 **+** mult3\_2**)))** **+** mult3\_3**;**

**always** **@(posedge** pixel\_clk **or** **posedge** rst**)** **begin**

**if(**rst**)** **begin**

pixel\_out **<=** 4'b0000**;**

**end** **else** **if(**enable**)** **begin**

**if(**product **<** 0**)** **begin**

pixel\_out **<=** 4'b0000**;**

**end** **else** **if(**product **>** 15**)** **begin**

pixel\_out **<=** 4'b1111**;**

**end** **else** **begin**

pixel\_out **<=** product**[**3**:**0**];**

**end**

**end** **else** **begin**

pixel\_out **<=** 4'b0000**;**

**end**

**end**

**endmodule**

1. **Controller Unit**

Table 2 - Controller Unit Verilog Code

`timescale 1ns **/** 1ps

**module** controller**(**

**input** pixel\_clk**,**

**input** rst**,**

**input** enable**,**

**input** **[**11**:**0**]** data\_in**,**

**output** **reg** done**,**

**output** **reg** **[**16**:**0**]** address**,**

**output** **reg** **[**11**:**0**]** kernel1**,**

**output** **reg** **[**11**:**0**]** kernel2**,**

**output** **reg** **[**11**:**0**]** kernel3**,**

**output** **reg** **[**11**:**0**]** pixel1**,**

**output** **reg** **[**11**:**0**]** pixel2**,**

**output** **reg** **[**11**:**0**]** pixel3

**);**

**parameter** FIRST\_LINE **=** 3'b000**,** SECOND\_LINE **=** 3'b001**,**

PROC1 **=** 3'b010**,** PROC2 **=** 3'b011**,** PROC3 **=** 3'b100**,**

DONE **=** 3'b101**;**

**reg** **[**2**:**0**]** state**;**

**reg** **[**3**:**0**]** buffer1 **[**641**:**0**];**

**reg** **[**3**:**0**]** buffer2 **[**641**:**0**];**

**reg** **[**9**:**0**]** index**;**

**reg** **[**11**:**0**]** previous\_data**;**

**integer** i**;**

**always** **@(posedge** pixel\_clk **or** **posedge** rst**)** **begin**

**if(**rst**)** **begin**

state **<=** FIRST\_LINE**;**

address **<=** 17'b0**;**

done **<=** 1'b0**;**

pixel1 **<=** 12'b0**;**

pixel2 **<=** 12'b0**;**

pixel3 **<=** 12'b0**;**

index **<=** 10'b0**;**

kernel1 **<=** 12'b1111\_1111\_1111**;** // -1

kernel2 **<=** 12'b1111\_1000\_1111**;** // -1, 8, -1 (-113)

kernel3 **<=** 12'b1111\_1111\_1111**;** // -1

previous\_data **<=** 12'b0**;**

**for** **(**i**=**0**;** i**<**642**;** i **=** i **+**1**)** **begin**

buffer1**[**i**]** **<=** 4'b0**;**

buffer2**[**i**]** **<=** 4'b0**;**

**end**

**end** **else** **if(**enable**)** **begin**

**if(**address **==** 103148**)** **begin**

state **<=** DONE**;**

**end**

**case(**state**)**

FIRST\_LINE**:** **begin**

**if(**index **<** 640**)** **begin**

buffer1**[**index**]** **<=** data\_in**[**3**:**0**];**

buffer1**[**index **+** 1**]** **<=** data\_in**[**7**:**4**];**

buffer1**[**index **+** 2**]** **<=** data\_in**[**11**:**8**];**

index **<=** index **+** 3**;**

address **<=** address **+** 1**;**

state **<=** FIRST\_LINE**;**

**end** **else** **begin**

index **<=** 10'b0**;**

state **<=** SECOND\_LINE**;**

**end**

**end**

SECOND\_LINE**:** **begin**

**if(**index **<** 640**)** **begin**

buffer2**[**index**]** **<=** data\_in**[**3**:**0**];**

buffer2**[**index **+** 1**]** **<=** data\_in**[**7**:**4**];**

buffer2**[**index **+** 2**]** **<=** data\_in**[**11**:**8**];**

index **<=** index **+** 3**;**

address **<=** address **+** 1**;**

state **<=** SECOND\_LINE**;**

**end** **else** **begin**

index **<=** 10'b0**;**

state **<=** PROC1**;**

**end**

**end**

PROC1**:** **begin**

pixel1 **<=** **{**buffer1**[**index**],**buffer1**[**index**+**1**],**buffer1**[**index**+**2**]};**

pixel2 **<=** **{**buffer2**[**index**],**buffer2**[**index**+**1**],**buffer2**[**index**+**2**]};**

pixel3 **<=** data\_in**;**

buffer1**[**index**]** **<=** buffer2**[**index**];**

buffer2**[**index**]** **<=** data\_in**[**3**:**0**];**

previous\_data **<=** data\_in**;**

index **<=** index **+**1**;**

**if(**index **==** 639**)** **begin**

state **<=** PROC1**;**

index **<=** 0**;**

address **<=** address **+** 1**;**

**end** **else** **begin**

address **<=** address **+** 1**;**

state **<=** PROC2**;**

**end**

**end**

PROC2**:** **begin**

pixel1 **<=** **{**buffer1**[**index**],**buffer1**[**index**+**1**],**buffer1**[**index**+**2**]};**

pixel2 **<=** **{**buffer2**[**index**],**buffer2**[**index**+**1**],**buffer2**[**index**+**2**]};**

pixel3 **<=** **{**previous\_data**[**7**:**4**],**previous\_data**[**11**:**8**],** data\_in**[**3**:**0**]};**

buffer1**[**index**]** **<=** buffer2**[**index**];**

buffer2**[**index**]** **<=** previous\_data**[**7**:**4**];**

index **<=** index **+**1**;**

**if(**index **==** 639**)** **begin**

state **<=** PROC1**;**

index **<=** 0**;**

address **<=** address **+** 1**;**

**end** **else** **begin**

state **<=** PROC3**;**

**end**

**end**

PROC3**:** **begin**

pixel1 **<=** **{**buffer1**[**index**],**buffer1**[**index**+**1**],**buffer1**[**index**+**2**]};**

pixel2 **<=** **{**buffer2**[**index**],**buffer2**[**index**+**1**],**buffer2**[**index**+**2**]};**

pixel3 **<=** **{**previous\_data**[**11**:**8**],** data\_in**[**3**:**0**],** data\_in**[**7**:**4**]};**

buffer1**[**index**]** **<=** buffer2**[**index**];**

buffer2**[**index**]** **<=** previous\_data**[**11**:**8**];**

index **<=** index **+** 1**;**

**if(**index **==** 639**)** **begin**

address **<=** address **+** 1**;**

state **<=** PROC1**;**

index **<=** 0**;**

**end** **else** **begin**

state **<=** PROC1**;**

**end**

**end**

DONE**:** **begin**

state **<=** DONE**;**

done **<=** 1**;**

**end**

**endcase**

**end**

**end**

**endmodule**

1. **Top\_Module**

Table 3 - Top Module Verilog Code

`timescale 1ns **/** 1ps

**module** top\_module**(**

**input** **wire** clk**,**

**input** **wire** rst**,**

**output** **wire** VGA\_HS**,**

**output** **wire** VGA\_VS**,**

**output** **wire** **[**3**:**0**]** VGA\_R**,**

**output** **wire** **[**3**:**0**]** VGA\_G**,**

**output** **wire** **[**3**:**0**]** VGA\_B

**);**

**wire** clk\_25MHz**,** data\_en**;**

**reg** **[**1**:**0**]** counter**=**0**;**

**assign** clk\_25MHz **=** counter**[**1**];**

**always** **@(posedge** clk**)** **begin**

counter **<=** counter **+** 1**;**

**end**

vga\_driver VGA**(**

**.**pixel\_clk**(**clk\_25MHz**),**

**.**rst**(**rst**),**

**.**VGA\_HS**(**VGA\_HS**),**

**.**VGA\_VS**(**VGA\_VS**),**

**.**data\_en**(**data\_en**)**

**);**

//------------------------------------RED--------------------------------

**wire** **[**11**:**0**]** R\_K1**,** R\_K2**,** R\_K3**,** R\_P1**,** R\_P2**,** R\_P3**;**

**wire** **[**16**:**0**]** R\_addr**;**

**wire** **[**11**:**0**]** R\_data**;**

blk\_mem\_red RAM\_R**(**

**.**clka**(**clk\_25MHz**),** // input wire clka

**.**ena**(**data\_en**),** // input wire ena

**.**wea**(**1'b0**),** // input wire [0 : 0] wea

**.**addra**(**R\_addr**),** // input wire [16 : 0] addra

**.**dina**(**1'b0**),** // input wire [11 : 0] dina

**.**douta**(**R\_data**)**

**);**

controller controller\_R**(**

**.**pixel\_clk**(**clk\_25MHz**),**

**.**rst**(**rst**),**

**.**enable**(**data\_en**),**

**.**data\_in**(**R\_data**),**

**.**done**(),**

**.**address**(**R\_addr**),**

**.**kernel1**(**R\_K1**),**

**.**kernel2**(**R\_K2**),**

**.**kernel3**(**R\_K3**),**

**.**pixel1**(**R\_P1**),**

**.**pixel2**(**R\_P2**),**

**.**pixel3**(**R\_P3**)**

**);**

conv\_unit conv\_R**(**

**.**pixel\_clk**(**clk\_25MHz**),**

**.**rst**(**rst**),**

**.**enable**(**data\_en**),**

**.**kernel1**(**R\_K1**),**

**.**kernel2**(**R\_K2**),**

**.**kernel3**(**R\_K3**),**

**.**pixel1**(**R\_P1**),**

**.**pixel2**(**R\_P2**),**

**.**pixel3**(**R\_P3**),**

**.**pixel\_out**(**VGA\_R**)**

**);**

//------------------------------------GREEN--------------------------------

**wire** **[**11**:**0**]** G\_K1**,** G\_K2**,** G\_K3**,** G\_P1**,** G\_P2**,** G\_P3**;**

**wire** **[**16**:**0**]** G\_addr**;**

**wire** **[**11**:**0**]** G\_data**;**

blk\_mem\_green RAM\_G**(**

**.**clka**(**clk\_25MHz**),** // input wire clka

**.**ena**(**data\_en**),** // input wire ena

**.**wea**(**1'b0**),** // input wire [0 : 0] wea

**.**addra**(**G\_addr**),** // input wire [16 : 0] addra

**.**dina**(**1'b0**),** // input wire [11 : 0] dina

**.**douta**(**G\_data**)**

**);**

controller controller\_G**(**

**.**pixel\_clk**(**clk\_25MHz**),**

**.**rst**(**rst**),**

**.**enable**(**data\_en**),**

**.**data\_in**(**G\_data**),**

**.**done**(),**

**.**address**(**G\_addr**),**

**.**kernel1**(**G\_K1**),**

**.**kernel2**(**G\_K2**),**

**.**kernel3**(**G\_K3**),**

**.**pixel1**(**G\_P1**),**

**.**pixel2**(**G\_P2**),**

**.**pixel3**(**G\_P3**)**

**);**

conv\_unit conv\_G**(**

**.**pixel\_clk**(**clk\_25MHz**),**

**.**rst**(**rst**),**

**.**enable**(**data\_en**),**

**.**kernel1**(**G\_K1**),**

**.**kernel2**(**G\_K2**),**

**.**kernel3**(**G\_K3**),**

**.**pixel1**(**G\_P1**),**

**.**pixel2**(**G\_P2**),**

**.**pixel3**(**G\_P3**),**

**.**pixel\_out**(**VGA\_G**)**

**);**

//------------------------------------BLUE--------------------------------

**wire** **[**11**:**0**]** B\_K1**,** B\_K2**,** B\_K3**,** B\_P1**,** B\_P2**,** B\_P3**;**

**wire** **[**16**:**0**]** B\_addr**;**

**wire** **[**11**:**0**]** B\_data**;**

blk\_mem\_blue RAM\_B**(**

**.**clka**(**clk\_25MHz**),** // input wire clka

**.**ena**(**data\_en**),** // input wire ena

**.**wea**(**1'b0**),** // input wire [0 : 0] wea

**.**addra**(**B\_addr**),** // input wire [16 : 0] addra

**.**dina**(**1'b0**),** // input wire [11 : 0] dina

**.**douta**(**B\_data**)**

**);**

controller controller\_B**(**

**.**pixel\_clk**(**clk\_25MHz**),**

**.**rst**(**rst**),**

**.**enable**(**data\_en**),**

**.**data\_in**(**B\_data**),**

**.**done**(),**

**.**address**(**B\_addr**),**

**.**kernel1**(**B\_K1**),**

**.**kernel2**(**B\_K2**),**

**.**kernel3**(**B\_K3**),**

**.**pixel1**(**B\_P1**),**

**.**pixel2**(**B\_P2**),**

**.**pixel3**(**B\_P3**)**

**);**

conv\_unit conv\_B**(**

**.**pixel\_clk**(**clk\_25MHz**),**

**.**rst**(**rst**),**

**.**enable**(**data\_en**),**

**.**kernel1**(**B\_K1**),**

**.**kernel2**(**B\_K2**),**

**.**kernel3**(**B\_K3**),**

**.**pixel1**(**B\_P1**),**

**.**pixel2**(**B\_P2**),**

**.**pixel3**(**B\_P3**),**

**.**pixel\_out**(**VGA\_B**)**

**);**

**endmodule**

Table 4 - Top Module Testbench Code

`timescale 1ns **/** 1ns

**module** top\_module\_tb**;**

// Clock and reset signals

**reg** clk**;**

**reg** rst**;**

**reg** slw\_clk**;**

// Outputs from the DUT

**wire** VGA\_HS**;**

**wire** VGA\_VS**;**

**wire** **[**3**:**0**]** VGA\_R**;**

**wire** **[**3**:**0**]** VGA\_G**;**

**wire** **[**3**:**0**]** VGA\_B**;**

// Instantiate the top\_module

top\_module dut **(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**VGA\_HS**(**VGA\_HS**),**

**.**VGA\_VS**(**VGA\_VS**),**

**.**VGA\_R**(**VGA\_R**),**

**.**VGA\_G**(**VGA\_G**),**

**.**VGA\_B**(**VGA\_B**)**

**);**

// Clock generation (100 MHz)

**always** **#**5 clk **=** **~**clk**;**

// File handling

**integer** red\_file**,** green\_file**,** blue\_file**;**

**integer** i**;**

// Test procedure

**initial** **begin**

// Initialize signals

clk **=** 1'b0**;**

rst **=** 1'b1**;**

// Open output files

red\_file **=** $fopen**(**"output\_red.txt"**,** "w"**);**

green\_file **=** $fopen**(**"output\_green.txt"**,** "w"**);**

blue\_file **=** $fopen**(**"output\_blue.txt"**,** "w"**);**

**if** **(**red\_file **==** 0 **||** green\_file **==** 0 **||** blue\_file **==** 0**)** **begin**

$display**(**"ERROR: Could not open output files."**);**

**end**

// Apply reset

**#**50 rst **=** 0**;**

// Wait for some time to let the system run and process data

//repeat (480 \* 640) begin // Simulate 640x480 image

**for(**i **=** 0**;** i **<**307200**;** i **=** i **+**1**)** **begin**

$fwrite**(**red\_file**,** "%d\n"**,** VGA\_R**);** // Write Red channel to file

$fwrite**(**green\_file**,** "%d\n"**,** VGA\_G**);** // Write Green channel to file

$fwrite**(**blue\_file**,** "%d\n"**,** VGA\_B**);** // Write Blue channel to file

**#**10**;**

**end**

$fclose**(**red\_file**);**

$fclose**(**green\_file**);**

$fclose**(**blue\_file**);**

$display**(**"Simulation completed. Output files are ready."**);**

// End simulation

$finish**();**

**end**

**endmodule**

A green and black screen

Description automatically generated

Figure - Top Module Behavioral Simulation

A screenshot of a computer

Description automatically generated

Figure - Control Unit Behavioral Simulation

A green and black screen

Description automatically generated

Figure - Convolution Unit Behavioral Simulation

Table 5 - Output Image Checking MATLAB Code

% Define the file names

r\_file **=** 'output\_red.txt'**;**

g\_file **=** 'output\_green.txt'**;**

b\_file **=** 'output\_blue.txt'**;**

% Define the image size

width **=** 640**;**

height **=** 480**;**

% Read the R, G, B values from the text files

R **=** dlmread**(**r\_file**);** % Reads all values into a vector

G **=** dlmread**(**g\_file**);**

B **=** dlmread**(**b\_file**);**

% Ensure the data matches the required dimensions

**if** length**(**R**)** **~=** width **\*** height **||** length**(**G**)** **~=** width **\*** height **||** length**(**B**)** **~=** width **\*** height

error**(**'The number of values in the files does not match the required 640x480 resolution.'**);**

**end**

% Reshape the vectors into 2D matrices (height x width)

R\_ **=** reshape**(**R**,** **[**width**,** height**])';** % Transpose to get the correct orientation

G **=** reshape**(**G**,** **[**width**,** height**])';**

B **=** reshape**(**B**,** **[**width**,** height**])';**

% Scale 4-bit values (0-15) to 8-bit values (0-255)

R\_ **=** uint8**(**R\_ **\*** **(**255 **/** 15**));**

G **=** uint8**(**G **\*** **(**255 **/** 15**));**

B **=** uint8**(**B **\*** **(**255 **/** 15**));**

% Combine the R, G, B channels into an RGB image

RGB\_image **=** cat**(**3**,** R\_**,** G**,** B**);**

% Display the image

imshow**(**RGB\_image**);**

title**(**'RGB Image'**);**

% Save the image to a file

imwrite**(**RGB\_image**,** 'output\_image.png'**);**

disp**(**'Image saved as output\_image.png'**);**

A screen shot of a television screen

Description automatically generated

Figure - MATLAB Output Image (Corrupted)

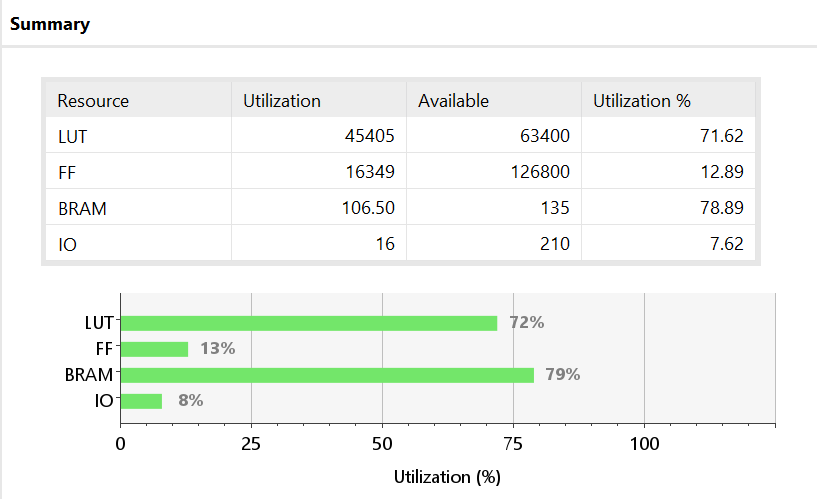


Figure - Utilization Summary

A close-up of a pattern

Description automatically generated

Figure - Text Files in Line

A screenshot of a computer

Description automatically generated

Figure - Primitives

**Explanations**

In the convolution unit, this design takes the inputs as 12-bit vectors and divides it by 3 to 4-bit for both pixels and kernels. After dividing the pixels, the fifth sign bit added to the divided pixel vectors but not the kernels. This division wires defined as signed bits for convolution process. Since the structure gives 13-bit product output after the convolution but it is needed to be 4-bit for the pixel\_out, the output chosen to be the least significant 4-bit of the product. Later, the pixel\_out is connected to VGA outputs in the top\_module.

Block RAM is the most basic 1clk RAM from IP design. The initial values are given by the text files received from MATLAB.

In the control unit, the buffers serve a purpose as a memory block to store the first two lines of the image for the FIRST\_LINE and SECOND\_LINE states. However, after switching to PROC states, the processed values in buffer2 register to buffer1 and data\_in to buffer2. The previous\_data serves as a role to restore data\_in values to be able to process the next states until the transition to the PROC1 from PROC3. This goes until the index == 639 because it is supposed to go 639 to make the [index + 2] equal to 642 which is the last pixel horizontally. However, the system does not shift vertically, it rather increases the address in PROC2. When the address == 642\*482 the system goes to DONE state.