

**Princess Sumaya University for Technology**

The King Abdullah II School for Electrical Engineering Computer Engineering Department

# 

**Computer Architecture (2)**

Project: Median Filter Pipelined Architecture

Group # 8

Student’s name: Ameen Abo Diak ID: 20210733

Student’s name: Zaid Hamarneh ID: 20190095

Table of Content

[**Abstract**](#_heading=h.30j0zll) **3**

[**Introduction**](#_heading=h.q5kr2jhff427) **3**

[**Project description**](#_heading=h.1fob9te) **3**

[**Objectives**](#_heading=h.2et92p0) **4**

[**Proposed Design**](#_heading=h.lnxbz9) **4**

[**Block Diagram**](#_heading=h.d8m1pkpydu3p) **5**

[**Design Analysis**](#_heading=h.1rzecj9851yl) **7**

[**Github project**](#_heading=h.llrvb8nh9cd4) **10**

[**Bibliography**](#_heading=h.cgkmo28ls601) **11**

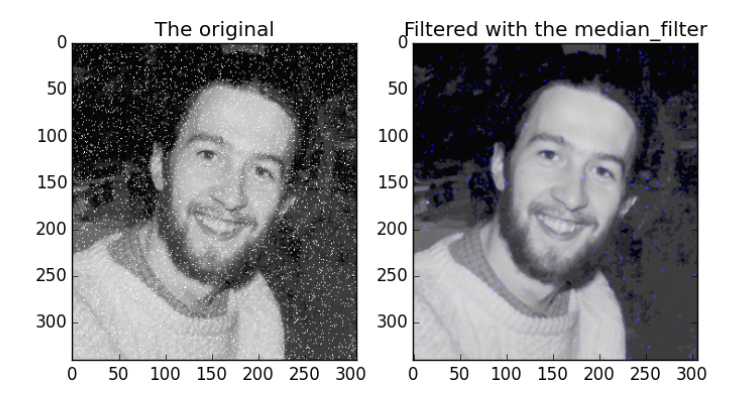
# Abstract

This project takes about the implementation of a 3\*3 median filter in Pipelined Architecture with Verilog language, which has nine stages and thirty-two registers, pipelined implemented by Verilog code has better performance than the nonpipelined implementation by c++ programming languages also with the result from Matlab which need extreme time to give the result. Matlab is a program for image processing that allows entering gray images by applying a noise than using a median filter to recover the image, the idea of pipelining applied throw calculating the median each time instead of using the sorting method and repeating the method on each window.

# Introduction

Image processing is a method to perform some operations on an image, to get an enhanced image, or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be an image or characteristics/features associated with that image, and one way to make image processing is by using an image filter that use to remove noise and improve image quality and many think according to the domain, there more than 4 image filters such as :

1. Low pass filters
2. High pass filters
3. Mean Filter
4. Median filter



# Project description

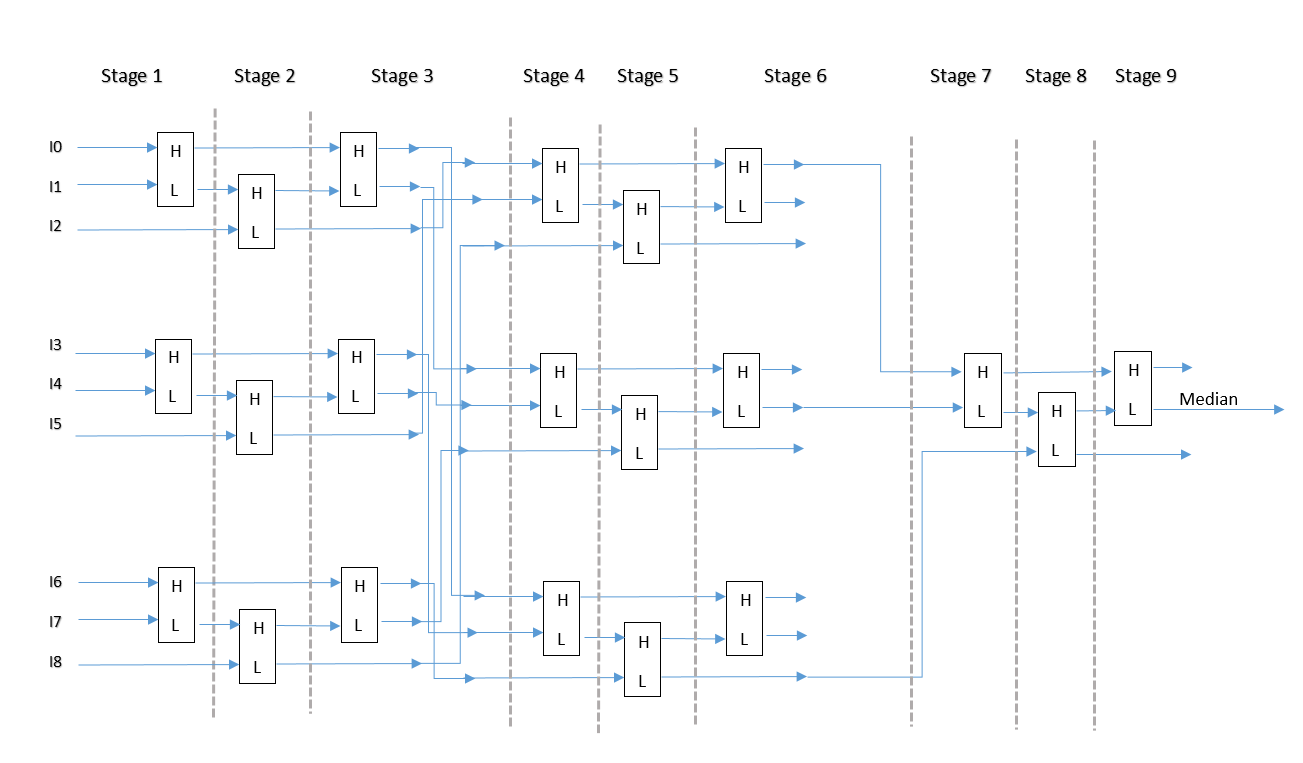
This project presents a hardware implementation of a 3× 3 median filter using Pipelined Architecture to reduce the time extinction, the median filter is a non-linear digital filtering technique, often used to remove [noise](https://en.wikipedia.org/wiki/Signal_noise) from an image or signal. Such [noise reduction](https://en.wikipedia.org/wiki/Noise_reduction) is a typical pre-processing step to improve the results of later processing (for example, [edge detection](https://en.wikipedia.org/wiki/Edge_detection) on an image). Median filtering is very widely used in digital [image processing](https://en.wikipedia.org/wiki/Image_processing) because, under certain conditions, it preserves edges while removing noise (but see the discussion below), also having applications in [signal processing](https://en.wikipedia.org/wiki/Signal_processing), the median filter is better than the LPF and HPF. (Subramaniam et al. 69-72).

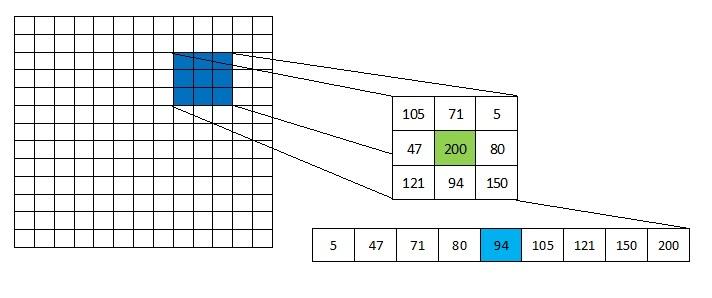
# Objectives

1. Learn how to use the Median Filter Pipelined Architecture to sort any discontinuities in a 3X3 radius.
2. Learn how to use Verilog in learning how to build a pipelined architecture for the median filter.
3. Understand how this Pipelined architecture can speed up the performance of the filter and how it can be used in different areas.

# Proposed Design

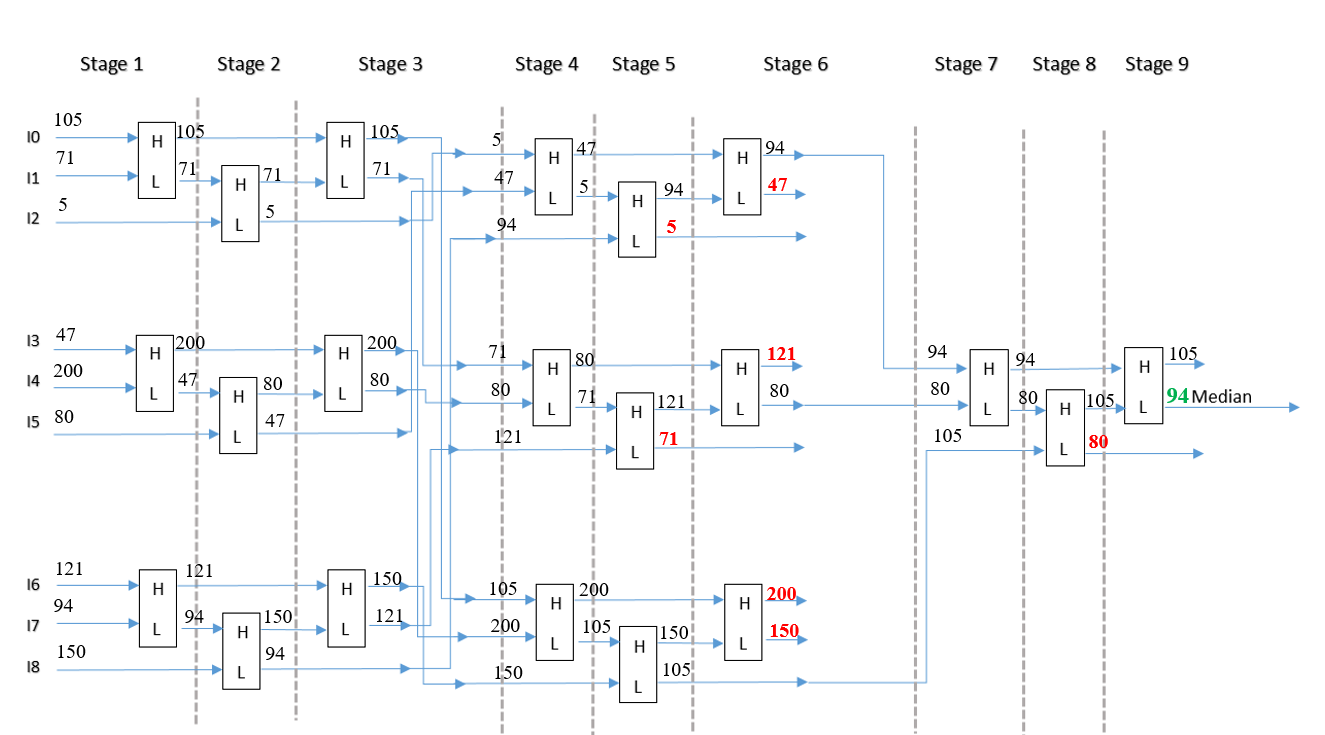
We will have in this design nine stages and thirty-two buffers, each stage will contain different amounts of comparators which can be shown in the figure below, where the comparator is a two-input-two-output comparator where the high output will be at the top and the lower output at the bottom. To increase the productivity of the pipeline, each 3 stages combined can be done in one stage, but that will have a slow transition between each stage, so to increase the productivity of the pipeline it has been divided into many stages.





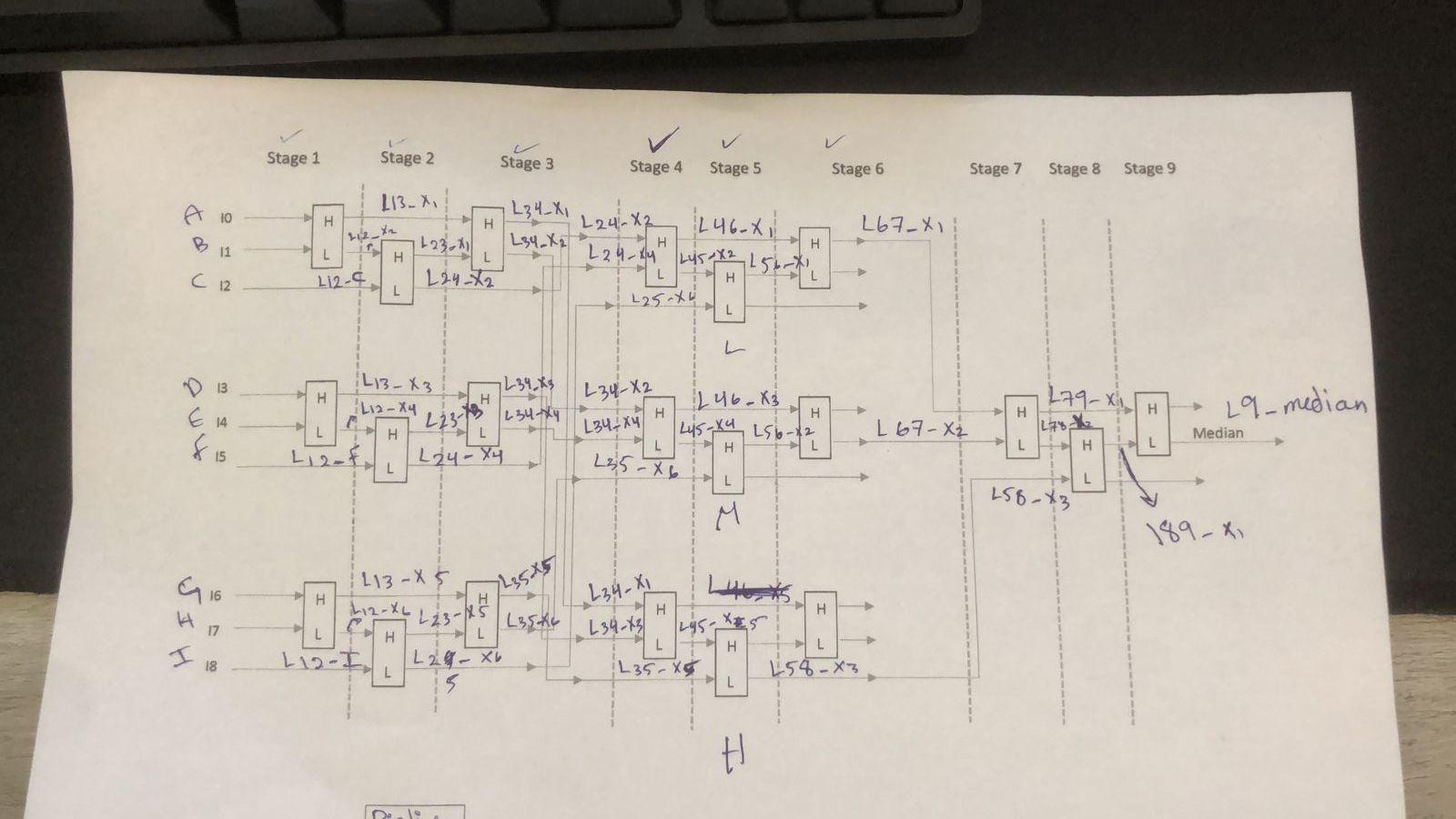
# Block Diagram

A Block diagram has been designed as shown below

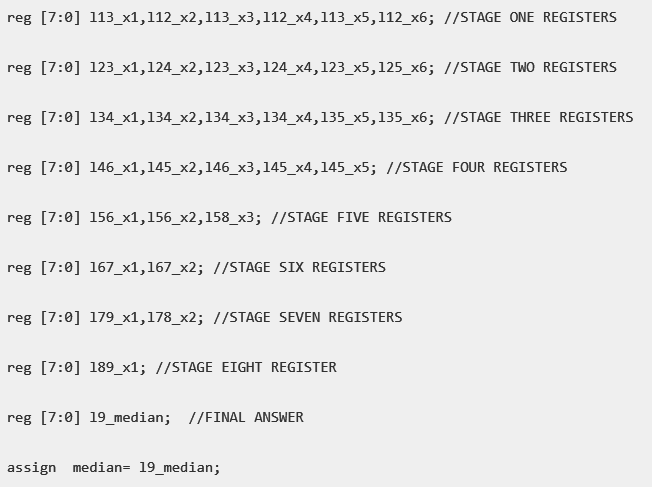


In the figure shown above, an example of inputs and outputs was given, as shown in stage 4, all the low values of the output in stage 3 were sorted into the top three comparators, and the median values in the middle three comparators, and finally the three high values at the three bottom comparators.

The image below shows the names of wires and registers in the block diagram which match the same names in the Verilog code.



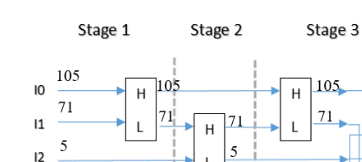
The image below from the Verilog code shows the register names which are the same as the image above and the register’s name is compatible with the two stages that link between them like reg L13 which links between stage one and stage three.



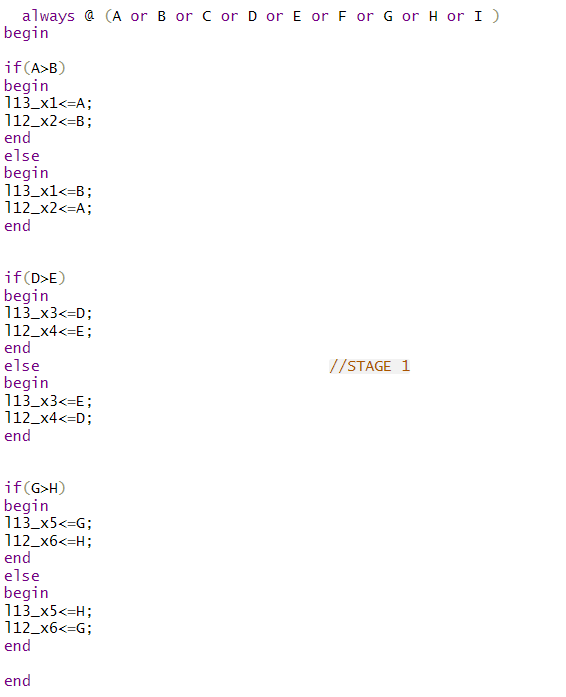
# Design Analysis

Mainly we sore the value of the image which is an about a two-dimensional array as a row-wise in the one-dimensional array which helps to get the cash friendly since we can reach the neighbor value without the need to return to main memory since we get block by block so at least in each block we will get the median size value or more if the block size more than the row or image size.

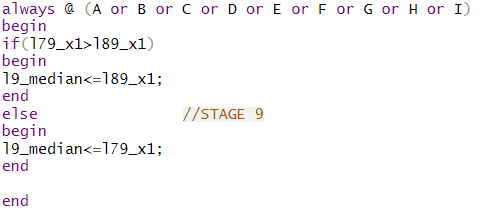
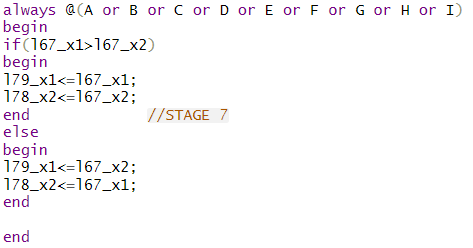
We use the first three stages to order and store the row value which is the main input- pixels value from the high to low and this operation will repeat each row by using comparators logic, so by end of stage three we get a sored row value which will be ready for replacement in next stages.



The image below describes the Verilog code in the first stage which is three comparators for the six input and use the same logic in stage two to six with input value from the main input and from the output of the previous stage.

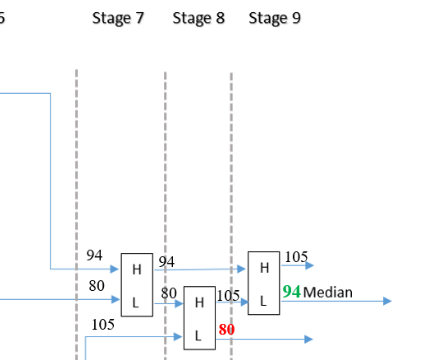


From stage seven to nine we have only one comparators logic in each stage as the Verilog code is shown in the images below, the output median result is assigned after stage nine from low comparators logic as the block digram.

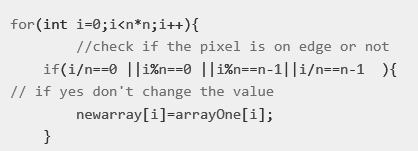


The idea of this is that in a normal array, to find the median, you first have to sort the entire array, element by element. But in the Median Filter Pipelined Architecture all that has to be done is to find the three closest elements of the array to the middle, and then find the median of those three.

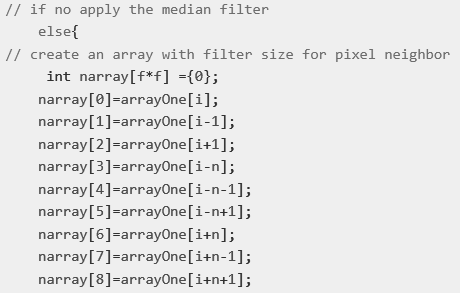
That is why, we take the highest output of the low terms(top three comparators) in stage 6, the median output of the median values(middle three comparators), and the lowest output of the highest values(bottom three comparators).



By end of stage 6, we get the candidate's three values for the median and we will apply the competitor method between the three values to get the median by applying the Verilog code as in the GitHub project compare that with the c++ code to the implementation that required many stages with check if the pixel value that we need to apply the median filter is on edge or not, if yes we don’t need to change the value.

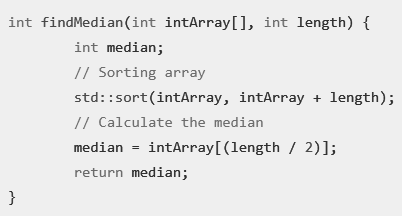


But if the pixel is not located at the edge we need to apply the median filter which is a 3\*3 filter, since we are talking about an image size more than the filter size we need to create a temporary array to store the nine neighbors value, and in this stage the meaning if cash friendly will appears by creat the array using the pixel index that we will apply the filter on it and the filter size to get the values in the row before and row after the main pixel.

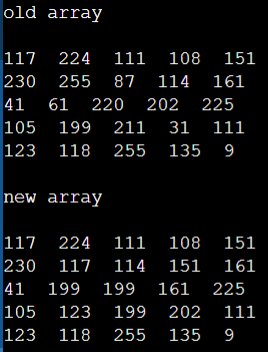
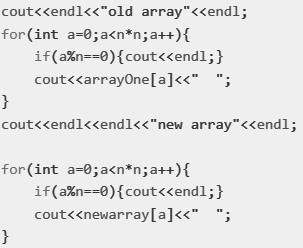


Now it is the time to call the find median function that basically in c++ depends on sorting the value and since the filter size in image processing is only an odd value, the function will be simple with an exact value not like when we have an even array size.





Here you can see the result after applying the filter and since we using a 3\*3 filter and the image size is 5\*5 the nine pixels values change which is the inner pixel also in the image processing we don’t apply the filter on each inner pixel, it is only about the pixel which it is value 0 or 255 which means it is white or black according to noise type.



Test runs and discussion  
As the image below shows the waveforms of the output of the first and second stages according to the input image values that appear in the Verilog code.

# 

# Github project

We work with Github as Version control that helps to track and manage changes to the project’s code and files and return to any version you saved before by committing the changes and pushing them when finishing a module we can merge it as a branch with the main or master branch, throw the GitHub repository link you can access the project files and folder for review since it is a public project and you can download it and edit or update without the ability to push your change until getting permission from System admin and in our case, it is the team member who creates the project and anyone with role Developer, Maintainer, and Owner, you can access the project repository from this link: <https://github.com/ameenabudiak/Median-Filter-Pipelined-Architecture>

Conclusion and Future Work

Median Filter Pipelined Architecture topic is one of the important topics in image processing and since the image processing course has been studied as an applied course, applying a median filter on the noisy image using Matlab came to Amazing results in image restoration and removing noise which not achieved with many other filters. However, the main disadvantage was that it consumed a lot of time to show the result, lot of time means that the user needs to wait for some seconds which is slower than many other filters, time consumption is considered a big challenge and we need to take it as a graduation project since if we succeed on applying the Median Filter Pipelined Architecture in a new way we can take this project for commercial and business company.

# Bibliography

J. Subramaniam, R. J. Kannan and D. Ebenezer, "Parallel and Pipelined 2-D Median Filter Architecture," in IEEE Embedded Systems Letters, vol. 10, no. 3, pp. 69-72, Sept. 2018, doi: 10.1109/LES.2017.2771453.