Princess Sumaya University for Technology

King Abdullah II Faculty of Engineering

Electrical Engineering Department



|  |
| --- |
| **Computer** **Architecture 2 Project (Pipeline)** |

|  |  |  |
| --- | --- | --- |
| *Authors:* |  | *Supervisor:* |
| Zaid Taha  Mohammad Elayyan  Group: 5 | 20190708  20200931 | Dr. Awos Kannan |

*May 26, 2022*

***Abstract***

Median filter is a non-linear digital filtering technique for removing noise from images and signals. A typical pre-processing procedure to improve the results of later processing is noise reduction (for example, edge detection on an image). The Pipeline procedure Pipelining is a method of executing many instructions at the same time. The pipeline is divided into stages, which are then linked together to form a pipe-like structure. Instructions come in from one end and leave from the other. Pipelining improves the total flow of instructions. This project combine the two above together, so we can do a median filter using pipeline.

Table of Contents

[1 Introduction 2](#_Toc104582898)

[1.1 Objectives 2](#_Toc104582899)

[2 Procedure and Methods 2](#_Toc104582900)

[2.1 components 2](#_Toc104582901)

[2.2 Implementation 2](#_Toc104582902)

[3 Results and Discussions 3](#_Toc104582903)

[3.1 Figures 3](#_Toc104582904)

[3.2 Tables 7](#_Toc104582905)

[3.3 Outputs 7](#_Toc104582906)

[4 Conclusions 8](#_Toc104582907)

[4.1 challenges 8](#_Toc104582908)

[4.2 pipeline design 8](#_Toc104582909)

[5 References 9](#_Toc104582910)

# Introduction

## Objectives

1. Showing a high-throughput median finding architecture in which, an input pixel is sorted by a high-speed (CS) module.
2. Attempting to fill the 4 X 4 window with four clock pulses at a time, as four pixels from the incoming grey image are read at a time.
3. This median will be implemented using a pipeline and parallel median design.
4. Determine the number of processes required to produce four output pixels from eight input pixels.
5. Attempting to reduce delay as much as feasible.
6. Implementing this design with (Verilog).
7. Determining the required frequency and gate count.

# Procedure and Methods

## components

In this project we used the below components to create this module:

* **2 x 1 Mux:**

This Mux was used to build the 8-bit comparator.

* **Comparator:**

The comparator was used in the compare and select module (CS), And this comparator is to produce selection lines for the 8 x 1 Mux, the results of the comparator were divided into three types (CO1, CO2, & CO3).

* **8 x 1 Mux:**

The Mux was used in the compare and select module, it receives its selection lines from the comparator, & receives its input from the image.

* **Compare and select module:**

Used in the median filter pipeline, its components are (3 comparators & 3 8x1 Mux), its input comes from the image, and the output is divided into 3 types (Max, Min, & Mid).

* **Median filter Module:**

It used in the median filter pipeline, it takes 9 inputs and produce one output, the inputs are coming from the 4 compare and select module in the previous stages.

## Implementation

In this part we will discuss the implantation method used in this project, we implemented the pipelined architecture using Verilog language, implemented each component of the filter (were discussed above), and tested them for sake of working.

# Results and Discussions

## Diagram, schematic Description automatically generatedFigures

**Figure 1 : 8-bit comparator architecture**

Eight input pixels are read every cycle using the proposed median filtering algorithm, consider an input digital image with CR pixels, where C and R stand for the number of columns and rows, respectively, the input pixel is only sent twice to the median filter architecture in the proposed design for the read operation.

As you can see in figure one the 8-bit comparator is made from logical gates & 11(2 to 1) MUX, it works as follows if A>B, then CO =1, else CO=0. (same thing goes for C & B comparator).

Diagram, schematic

Description automatically generated

**Figure 2: Architecture of compare and select module.**

The proposed architecture of CS module is illustrated in figure 2. It is implemented using three 8 x 1 multiplexer, A>B, A>C, and B>C are three comparators. All three comparators and multiplexers are running at the same time.

As a result, processing just takes one cycle. Each comparator compares two eight-bit input streams and outputs a single-bit result.

The first comparator compares two input values A and B and outputs bit CO1. The second and third comparators, meanwhile, compare A and C, B and C, and produce CO2, CO3, respectively. Three multiplexers use the three-comparator output (CO) values as selective lines (S0, S1, and S2). The essential values are produced by the output of three multiplexers (Mux1 providing Maximum (Max), Mux2 producing Middle (Mid), and Mux3 producing Minimum (Min) values). Table 1 shows the inputs and selective lines of each multiplexer.

Diagram

Description automatically generated

**Figure 3: Block diagram of proposed median filter.**

The median filter uses a two-dimensional (2D) mask and performs spatial filtering. Each pixel in the digital input image receives this treatment. The center pixel is chosen after the input pixels are organized in ascending order.

This pixel is referred to as a median pixel. The pixel value in the produced digital image will be the obtained median value. Figure 3 shows the construction of a 2D median filter used to calculate the median value of nine input pixels.

The nine input pixels are placed in a 3 × 3 window, and the median value is determined in three stages. Three rows are sorted in the first stage using CS1-CS3 modules. Three columns are sorted using CS4-CS6 in the next stage. In the final stage, the diagonal pixels in the 3 x 3 window are sorted. A median value of 3 x3 window is calculated using the middle value of the last stage (CS7).

Diagram, schematic

Description automatically generated

**Figure 4: Architecture of proposed parallel pipelined median**

Figure 4 depicts the proposed pipe-lined median filter architecture. Eight input pixels, named IP, are required by the architecture (0-1,0-3). These eight pixels are split into two CVs, each with a word size of 32 bits. The first two columns of input pixels are IP(0,0-3) and IP(1,0-3), respectively. These two CVs are kept in D1-D12 data storage elements, which are ordered in increasing order. The sorting is done in the first stage using Compare and Select (CS) modules. After sorting, there are 24 values left, which are separated into eight groups.

## Tables

**Table 1: Table for selecting inputs and outputs of MUXes.**

Table

Description automatically generated

## Outputs

Text

Description automatically generated

In the above images you can observe that the inputs that enter the median filter are changing, that indicate that the median filter is working efficiently.

The median filter is 3 x 3 window as discussed in the proposal only two words are passed in the suggested method, rather than more than three words, to complete a single cycle operation.

# Conclusions

As a result, we achieved targeted goal that we aimed to since the beginning of our project, the results are:

* We were able to modify the inputs using the above figure in terms of Verilog code.
* The inputs that we entered are changing through the process which means that median filter works properly.
* It is observed in the previous output that the pipeline process is working correctly.

## challenges

We faced many challenges through this project, and we were successfully able to solve them all, and the challenges are:

* understanding the concept of the median filter itself.
* Connecting the components in Verilog as described above.
* Code problems:

1. High impedance (z); the reason of this problem was due to the un- connectivity of the components.
2. Unknows (x); wrong data type.
3. Wrong output.

## pipeline design

In this section we will discuss how we divided pipeline through the median filter:

* Instruction fetch (IF): basically the inputs of the image were stored in registers, so then they were transferred into the next stage.
* Instruction Decoding & Execution stage (ID/ EX): the inputs that were coming from the previous stage are sent to the Compare and select module (CS) to produce the inputs of the next stage, they are stored in registers.
* The Memory stage (M): the results of the previous stage are now stored in the memory.

# References

|  |  |
| --- | --- |
| [1] | "IEEE". [Online].[VLSI implementation of high throughput parallel pipeline median finder for IoT applications (ias.ac.in)](https://www.ias.ac.in/article/fulltext/sadh/045/0075) |