Histogram Equalizer
ECE 520 Final Project
Christopher Johnson, Joshua Stevens

Project Responsibilities:
csjohns3. Christopher Johnson:
Input Pipeline, Testbench, Top, Report

jasteve4. Joshua Stevens:
Output Pipeline/CDF, Testbench, Top, Report

Delay (ns to run ex) Clock period: Cycles:	(delay.area)(ns-1.um-2):	Area: um^2 Logic: Memory: N/A

Delay (ns to run example) Clock period: cycles:	1/(delay.area)($ns^{-1}.um^{-2}$)

***Abstract* - This paper describes a hardware design responsible for improving the contrast of individual frames of a video capture device. The algorithm is based on a histogram equalizer and the hardware is heavily pipelined in order to allow a low clock speed and the ability to quickly process individual images. Using Synopsys Design Vision and Mentor Graphic ModelSim, we prove that it with this module, it is possible to process video at REPLACE $frames/second$ with an area of REPLACE um^2.**

I. Introduction

When uses a video capturing device, it can be quite common to achieve footage with poor contrast. This is something that can be solved relatively easy using a software histogram equalizer however, this will limit the video capture speed the speed of the software. To alleviate this bottleneck, we have hardware histogram equalizer. Our design is heavily pipelined allowing the clock speed to be quite low while allowing multiple images to be processed at a time. The result is a hardware module that can process video at **REPLACE** $frames/second$ and an area of **REPLACE** m^2. This is significant since this is much smaller and faster than comparative software performing the same function.

II. Micro-Architecture

Our overall system is divided into two major stages (”Input Pipeline/CDF Pipeline” and ”Output Pipeline”) that are individually pipelined and interface through a controller module. These two stages are pipelined such that the ”Input Pipeline/CDF Pipeline” module runs in approximately the same time as ”Output Pipeline” allowing both modules to run images simultaneously with a minimum number of unused clock cycles.

Input Pipeline

This module is responsible for reading pixel values from memory and storing how many times each pixel value occurs to SRAM 2. In order to improve efficiency of this process, a five stage pipeline is

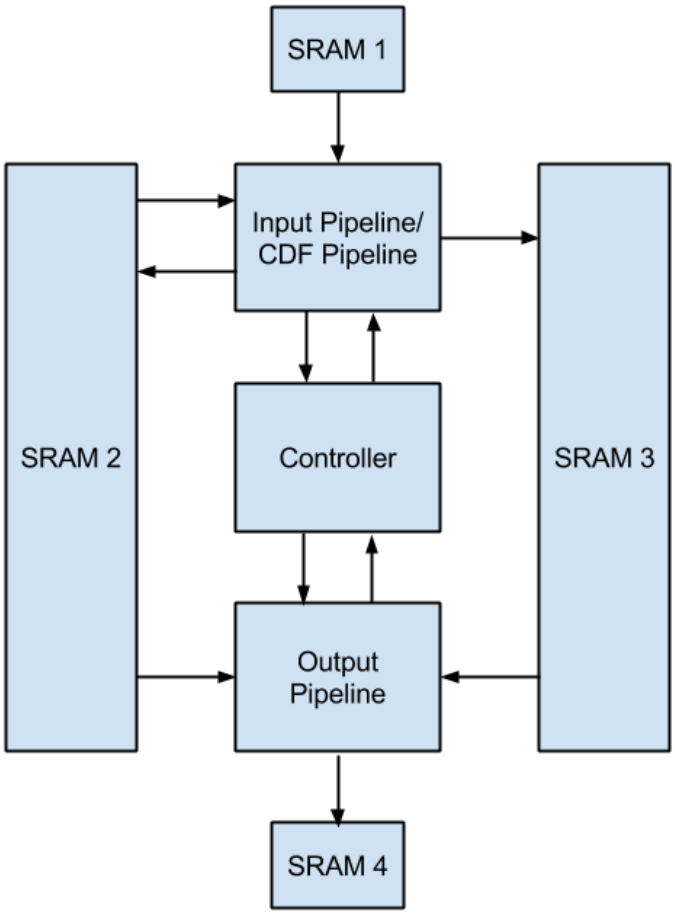

Fig. 1. System Overview

used. This allows memory operations to happen in separate cycles from the accumulation stages. This is crucial since it takes 2ns to read or write to the SRAM. In order to keep this pipeline running as quick as possible, three data bypasses were introduced for coping with the 2ns memory write.

III. Conclusion

”Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?”