# Filtering & Histogram Module

## Overview

The filtering block performs median filtering on a binary image using programmable window sizes and strides. The median filter is performed on an image stored in a RAM and the resulting image (filtered) is stored in a different block memory.

## Top level Interface

## 

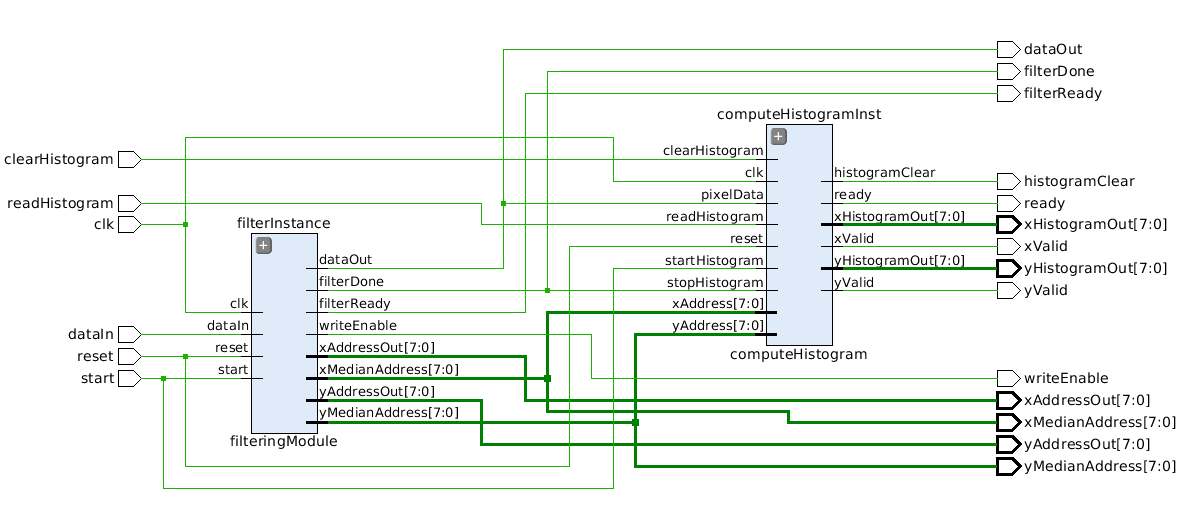
Histogram Interface

Filtering module interface

## Signals

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Signal name** | **Width** | **Direction** | **Brief** |
| 1 | clk | 1 | IN | System Clock |
| 2 | dataIn | 1 | IN | Binary image pixel value |
| 3 | reset | 1 | IN | System reset |
| 4 | start | 1 | IN | Signal to begin filtering and histogram computation |
| 5 | readHistogram | 1 | IN | Signal to read out the histogram |
| 6 | clearHistogram | 1 | IN | Signal to clear histogram |
| 7 | filterReady | 1 | OUT | Goes high when filtering module is ready to perform computation |
| 8 | filterDone | 1 | OUT | Goes high when the filtering computation is done |
| 9 | xAddressOut | 8 | OUT | Input address to read out the Raw image memory |
| 10 | yAddressOut | 8 | OUT | Input address to read out the Raw image memory |
| 11 | xMedianAddress | 8 | OUT | Input address to the filtered image memory |
| 12 | yMedianAddress | 8 | OUT | Input address to the filtered image memory |
| 13 | writeEnable | 1 | OUT | Write enable signal to the filtered image memory |
| 14 | dataOut | 1 | OUT | Pixel data out of the median pixel |
| 15 | xHistogramOut | 8 | OUT | X Histogram value |
| 16 | yHistogramOut | 8 | OUT | Y histogram value |
| 17 | xValid | 1 | OUT | Stays high for all valid x histogram values |
| 18 | yValid | 1 | OUT | Stays high for all valid Y histogram values |
| 19 | histogramClear | 1 | OUT | Goes high when the histogram is cleared out |
| 20 | Ready | 1 | OUT | Ready signal is asserted when the histogram is ready to be read out |

## Block description and flow

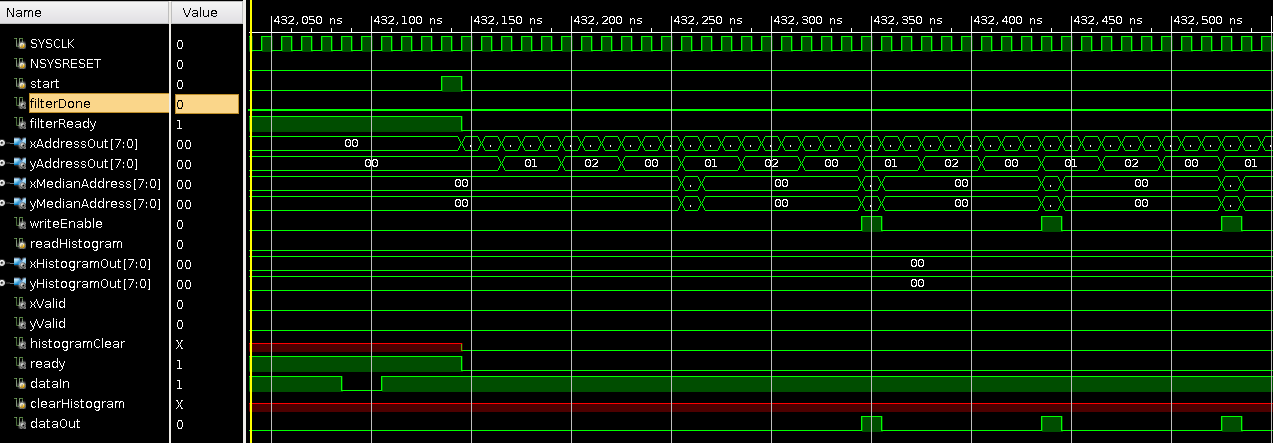


The block contains two major submodules the filtering module and histogram computation module. The block needs two memory modules (1 to read the raw image from & 1 to write the filtered image). The flow for running the module is

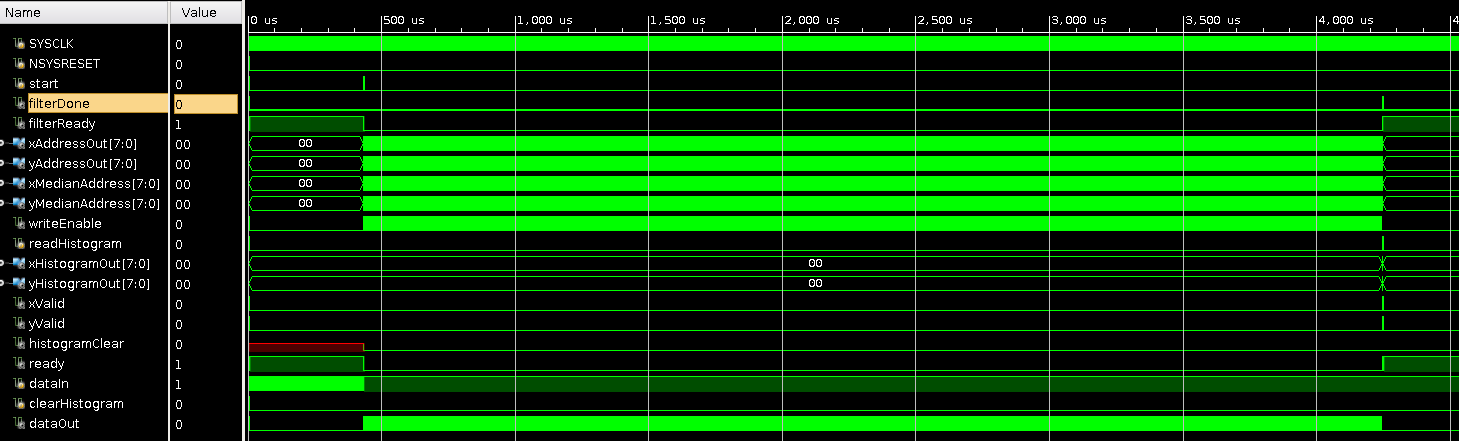
* Once the binary image is written to the memory the user needs to monitor the *filterReady* signal
* If *filterReady* is high it indicates that the filtering module is ready to start the computation
* User then pulses the *start* signal high for one clock cycle to begin filtering computation
  + The filtering module will then access the raw image memory so other blocks should release control of the memory
* While the block is reading the memory contents and performing computation the *filterReady* signal goes low. The filtering module is also accessing the memory used to store the filtered image
* Once the filtering is done the *filterDone* signal is asserted high which means that the memories are free to use by the other blocks and the median filtered image is available in the memory
* The histogram computation is done in parallel to the filtering computation. *ready* signal is asserted high when the histogram computation is finished and the histogram is ready to be read out.
* To read out the histogram user pulses the *readHistogram* signal. *xHistogramOut & yHistogramOut* will then output the histogram values at each cycle starting from x = 0 and y = 0.
* *xValid & yValid* remains high for valid x & y histogram values respectively. Values should be discarded if valid signals are low. Read can be performed multiple times.
* Histogram needs to be cleared after every computation. To clear the histogram *clearHistogram* signal should be pulsed. After the histogram is cleared the *histogramCleared* signal will be asserted after which a new histogram computation can begin.

## Functional waveforms

Start signal is pulsed to start the computation of filtering. The *filterReady* signals goes low after a clock cycle



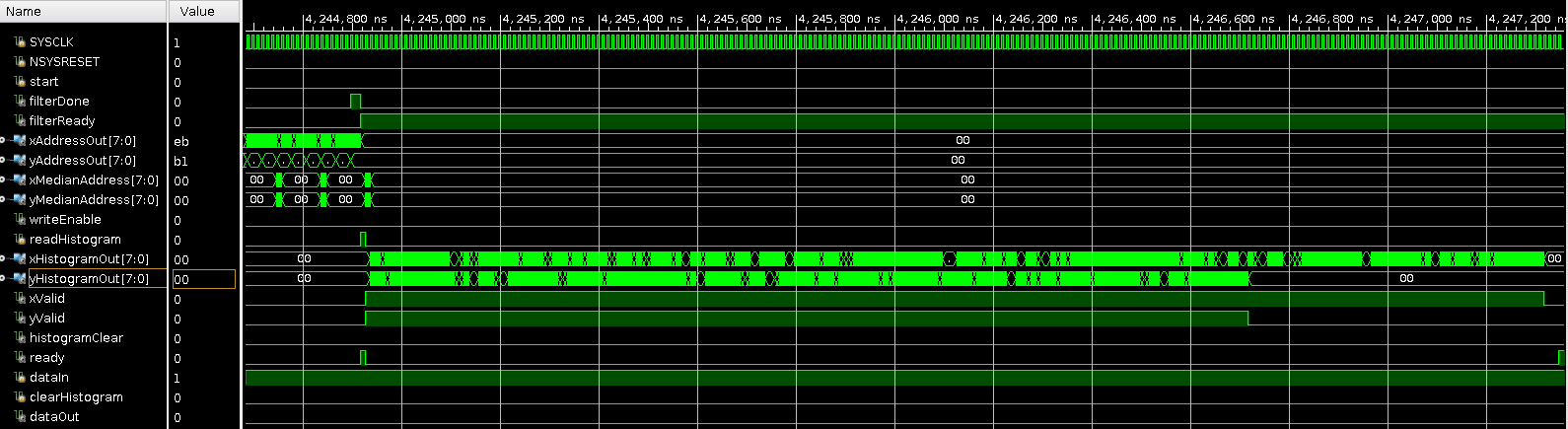
Once the *filterReady* signal goes low the block is performing filtering and histogram computation and writing the median filter computation in the memory to hold the filtered image. During this phase both the memories are being accessed by this block and the other blocks should not access the memories



Filtering & histogram computation phase

The below timing diagram shows the histogram reading mechanism. Once the *ready* signal is asserted user can pulse redHistogram signal to read out the histogram. X & Y values are outputted (one value every clock cycle starting from x = 0,1,2 … 239 & y = 0,1,2 … 179).

xValid stays high for 240 (image width) clock cycles and yValid stays high for 180 (image height) clock cycles



## User editable parameters & caveats

IMAGE\_WIDTH: Width (in pixels) of the binary image to be filtered (default = 240),   
IMAGE\_HEIGHT: Height (in pixels) of the binary image to be filtered (default = 180),  
WINDOW\_SIZE: Size of the window for median calculation (default = 3 for a 3x3 window), WINDOW\_STEP: Step size of the moving window (default = 1)

* The median filter block does not pad the image with 0’s hence the output image is smaller than the original image by *floor*(WINDOW\_SIZE/2) on each side. So a 240x180 image with a 3x3 filter will output 238x178 median filtered image.
* The median filtering block assumes a 2 cycle delay between address and the data out from the raw binary image memory any change to the latency from the memory will render the filtering calculation invalid.
* The histogram needs to be cleared before performing a new computation and cannot be overwritten. Though after a computation is done it can be read multiple times before erasing.