# Video Enhancement Algorithms on System on Chip

**Dr.Ch. Ravikumar, Dr. S.K. Srivatsa**

***Abstract-*** This paper presents a way to improve the computational speed of video enhancement using low-cost FPGA-based hardware. To design real-time adaptive and reusable image enhancement architecture for video signals based on a statistical processing of the video sequence. The VHDL hardware description language has been used in order to make possible a top-down design methodology. Generic design methodology has been followed by means of two features of the VHDL: global packages and generic pass. Video processing systems like this one require specific simulation tools in order to reduce the development time. Real tine image processing in an application environment needs a set of low cost implementations of various algorithms. This paper presents a median filter based on a system on chip and working at video rate. It includes its own memory and can be used without any image memory for on line processing. The architectural choices have made it possible to design a small size chip with a high performance level. A VHDL test bench has been designed specifically for Video processing applications to facilitate the simulation process. A video enhancement processor concept is proposed that enables efficient hardware implementation of enhancement procedures and hardware software co-design to achieve high performance, low-cost solutions. The processor runs on an FPGA prototyping board.

***Index Terms***- SOC, FPGA, VHDL, video enhancement, Video enhancement comparison

## I. FIR AND MEDIAN FILTERING

One of the most common video-enhancement blocks is the FIR (finite-impulse-response) filter. A FIR filter multiplies and sums a sequence of received-video-data impulses, creating a 2-D convolution process. A 2-D FIR filter can perform 2-D convolution using matrices of 3×3, 5×5, or 7×7 coefficients. A 2-D FIR filter?s key provides sharpening, smoothing, and edge detection of a video image. By designing the proper coefficients and applying the correct matrix, you can produce a crystal-clear video output. However, the electrical system can introduce video noise into a video stream during transmission in any channel. A median filter provides a simple and effective noise-filtering process. The median value of all the pixels in a population?that is, a selected neighborhood block?determines each video pixel. The median value of a population is that value in which one-half of the population have smaller values than the median and the other half has larger values than the median value.

## II. IMAGE/VIDEO PROCESSING ON FPGAS

Image and Video Processing on embedded devices is a growing trend in the industry today where security is depended on cameras placed everywhere, replacing people behind monitors. FPGAs are preferred for their parallel pixel processing power over sequential microprocessors. Newer FPGAs are packing more gates and requiring lower power, which is certainly attractive features for embedded designers. Instead of a trial-run on an expensive ASIC fabrication process of a custom design, FPGAs offer a cost effective alternative, or at least a prototype before millions of dollars are invested and sometimes to find out the ASIC doesn't perform as expected from simulation on a computer with software.

Image and Video Processing often requires DSP algorithms on multiple rows/columns of pixels/data concurrently. A typical TI DSP processor may have two ALUs (**A**rithmetic**L**ogic **U**nits) that perform MAC (**M**ultiply & **AC**cumulate) operations, an FPGA can have, for example, 200 MAC blocks processing pixels in parallel.

Some FPGAs now have dedicated hard-core DSP/MAC silicon blocks in an FPGA for faster processing power than FPGA fabric designed as MACs.

## III. IMAGE/VIDEO PROCESSING ON FPGAS

Image and Video Processing on embedded devices is a growing trend in the industry today where security is depended on cameras placed everywhere, replacing people behind monitors. FPGAs are preferred for their parallel pixel processing power over sequential microprocessors. Newer FPGAs are packing more gates and requiring lower power, which is certainly attractive features for embedded designers. Instead of a trial-run on an expensive ASIC fabrication process of a custom design, FPGAs offer a cost effective alternative, or at least a prototype before millions of dollars are invested and sometimes to find out the ASIC doesn't perform as expected from simulation on a computer with software.

Image and Video Processing often requires DSP algorithms on multiple rows/columns of pixels/data concurrently. A typical TI DSP processor may have two ALUs (**A**rithmetic**L**ogic **U**nits) that perform MAC (**M**ultiply & **AC**cumulate) operations, an FPGA can have, for example, 200 MAC blocks processing pixels in parallel.
Some FPGAs now have dedicated hard-core DSP/MAC silicon blocks in an FPGA for faster processing power than FPGA fabric designed as MACs.

## IV. THE FPGA IS THE HEART OF THE SYSTEM

In the past, dedicated DSP chips were an option, but even with high internal clock frequencies, massive I/O capacity and advanced graphics accelerators these are no longer adequate, because they are limited to a certain number of operations per clock frequency. Dedicated graphics processors provide higher performance and are more specialised, though they still have the same limitations.

Modern FPGAs do not suffer from such limitations. The capacity has been greatly increased, all of the most recent high-speed interfaces to memory etc. are fully supported, and the number of parallel operations is limited only by the total capacity of the FPGA. Even with lower internal clock frequencies versus dedicated CPUs, this is compensated for by the massive parallelisation that is achievable. The most recent FPGAs with dedicated DSP chips are ideal for this type of project, and they function as the heart of the application.

Most current DSP algorithms for video processing are fundamentally composed of multiply-accumulate operations (MACs) that are carried out in both dimensions according to the desired resolution. The DSP chips implement the MACs directly at the desired word width and high clock frequency. For applications with fixed processing paths, the separate processing units are connected together in a streaming architecture, where the processing time is the same for all the units in the processing chain and the intermediate results can be buffered internally in the FPGA.

## V. MODIFIED ADAPTIVE MEDIAN FILTER

The Modified Adaptive Median Filter is designed to eliminate the problems faced with the standard median filter. The basic difference between the two filters is that, in the Adaptive Median Filter, the size of the window surrounding each pixel is variable. This variation depends on the median of the pixels in the present window. If the median value is an impulse, then the size of the window is expanded. Otherwise, further processing is done on the part of the image within the current window specifications. „Processing" the image basically entails the following: The center pixel of the window is evaluated to verify whether it is an impulse or not. If it is an impulse, then the new value of that pixel in the filtered image will be the median value of the pixels in that window. If, however, the center pixel is not an impulse, then the value of the center pixel is retained in the filtered image. Thus, unless the pixel being considered is an impulse, the gray-scale value of the pixel in the filtered image is the same as that of the input image. Very diverse FPGA-based custom-computing boards are appearing in the market. These boards possess different interfaces for their communication with the host. But in general, boards devoted to real-time image processing have a USB interface, because it gives them the necessary speed to work as coprocessors. Also, USB bus has a growing popularity due to its interesting properties.

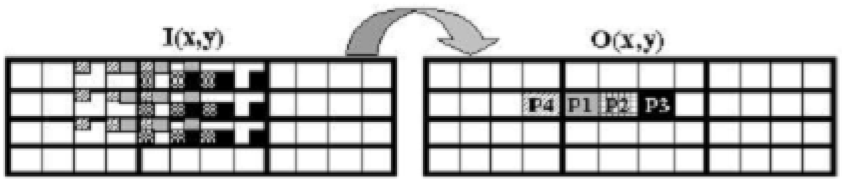


Fig 2. simultaneous computation of output pixel

The fact that we have a 32-bit data bus has a very large influence in the necessary hardware architecture for implementing image processing operations, because it causes that in each read/write operation we obtain/send four image pixels (supposing 8-bit pixels). We have gained benefit from this situation replicating the functional units in order to apply the median filter simultaneously on four pixel neighbourhoods. In this way we take advantage of the inherent neighbourhood parallelism, and we accelerate the operation four times. Figure 2 presents the approach followed for the simultaneous computation of these four output pixels.

Images are divided in pixels (squares) that are grouped in 32-bit words (4 pixels). The value of each output pixel O(x,y) is computed using the 9 pixels of the image I that are inside the 3x3 mask with centre in I(x,y). Each mask application has been represented with a different texture.

Note that the pixel P4 of the previous word is computed and not that of the current word. In this way, it is only necessary to read six words in the input image instead of nine, reducing the number of read operations, and therefore increasing the performance. Pipelining this approach using two stages it is possible to get an architecture that writes four pixels (one word) in the output image in each clock cycle, only reading three input image words by cycle

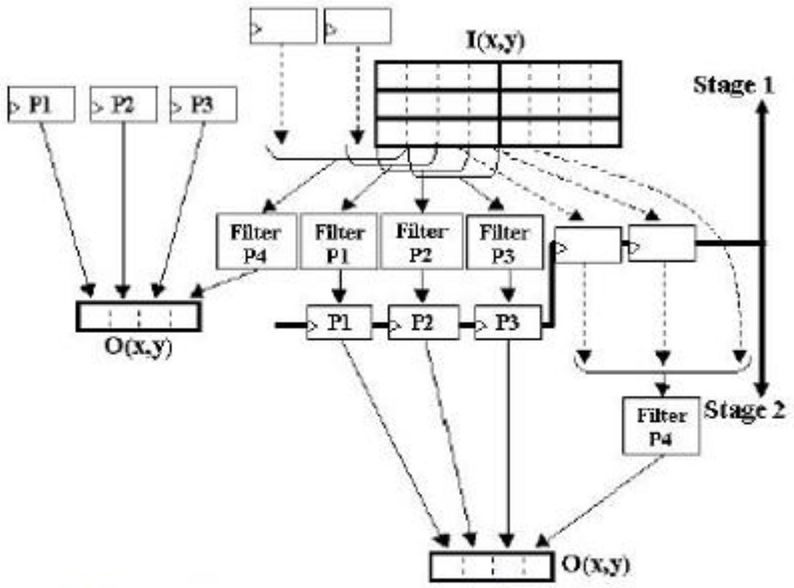


Fig 3. Pipelining approach using two stages

3. Moving Window Architecture In order to implement a moving window system in VHDL, a design was devised that took advantage of certain features of FPGAs. FPGAs generally handle flip -flops quite easily, but instantiation of memory on chip is more difficult. Still, compared with the other option, off-chip memory, the choice using on-chip memory was clear. It was determined that the output of the architecture should be vectors for pixels in the window, along with a data valid signal, which is used to inform an algorithm using the window generation unit as to when the data is ready for processing. Since it was deemed necessary to achieve maximum performance in a relatively small space, FIFO Units specific to the target FPGA were used. Importantly though, to the algorithms using the window generation architecture, the output of the window generation units is exactly the same. This useful feature allows algorithm interchangeability between the two architectures, which helped significantly, cut down algorithm development time. A window size was chosen because it was small enough to be easily fit onto the target FPGAs, and is considered large enough to be effective for most commonly used image sizes. With larger window sizes, more FIFOs and flip -flops must be used, which increases the FPGA resources used significantly. Figure 1, 2 shows a graphic representation of the FIFO and flip flop architecture used for this design for a given output pixel window.

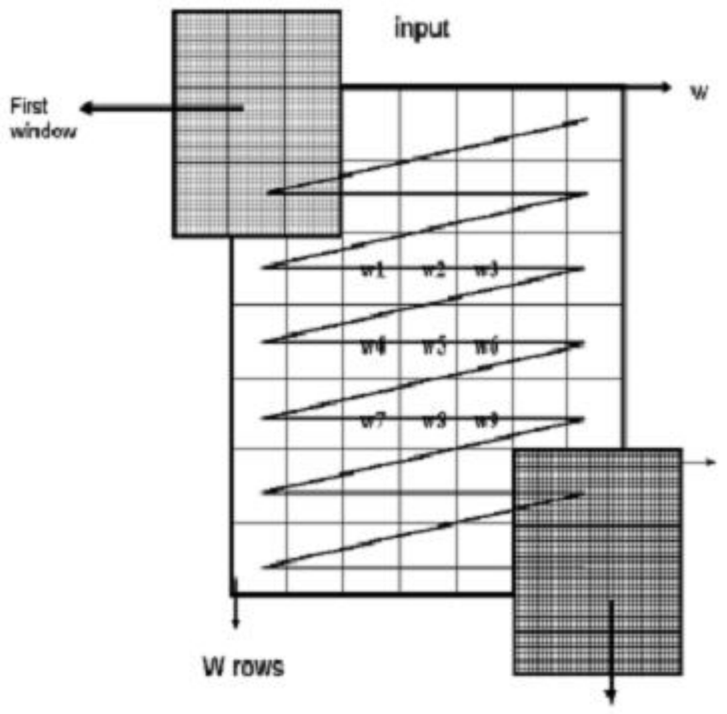


Fig.4 Moving Window Architecture

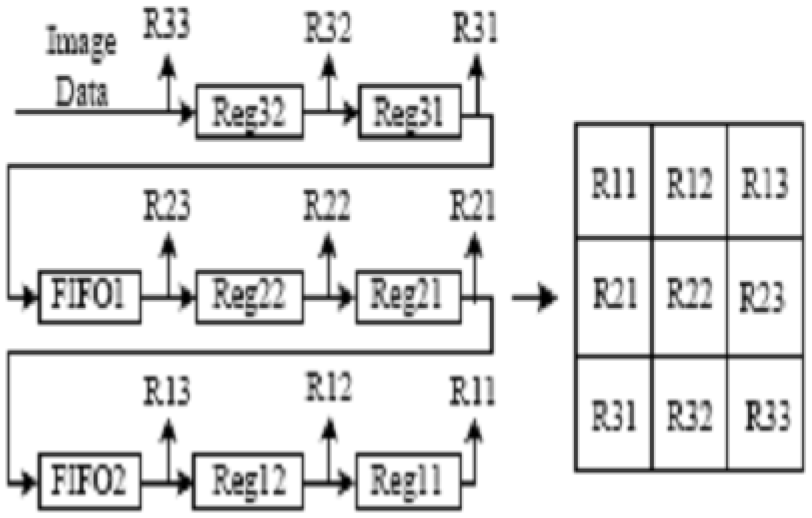
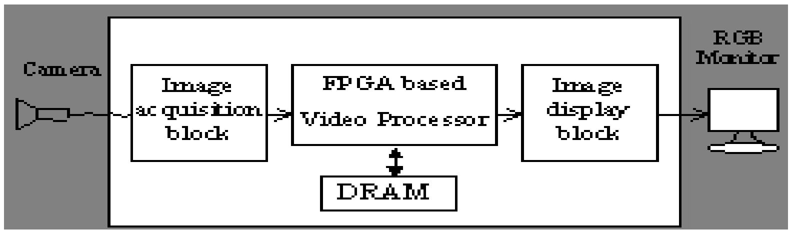


Fig. 5 Reading Pixels from Window.

## VI. HARDWARE IMPLEMENTATION ON A FPGA BASED VIDEO BOARD

To assess the effectiveness of our skeleton-based approach, we have implemented our system on a FPGA based video processing board. A functional block diagram of the FPGA board is given by figure



Bit parallel arithmetic has been chosen to implement the IP operations on the onboard FPGA. This choice is motivated by the fact that bit parallel architectures often lead to a better time-hardware product than bit serial ones. This is mainly due to the existence of dedicated fast carry logic on Xilinx FPGAs. However, in the context of processing real time video, the FPGA board influences the choice of the arithmetic. If bit serial arithmetic is to be used, there is a need to generate a bit clock from the pixel clock. The bit clock frequency is 'N' times the pixels' clock (for an 'N'-bit pixel). This implies a bit clock frequency of 108 MHz for 8-bit length pixel processing, and 216 MHz for 16-bit length pixel processing. Thus the architectures used will be implemented from bit parallel-based skeletons. A parallel implementation is easier to implement and can be efficiently implemented using dedicated fast carry logic.

Image processing is usually performed on pictures stored in an image memory. Achieving global transformations, such as a FF1, requires that one faces difficulties in communication with the computation unit (address processing, high data rates). On the other hand, most low—level image processing is performed on a m*n work window involving pixels of n adjacent image lines. If the image is provided on video format (line by line scanning), on—line processing can be performed, if one assumes that n—l lines are bufferized, whatever the image height. No random access image memory is therefore necessary.

## VII. RESULT

The adaptive median filter for video enhancement is designed to remove impulsive noise from video. Therefore, our algorithm's performance was first tested with basic salt and pepper noise with a noise density of 0.25. The next test involves processing images that contain impulsive and/or non-impulsive noise. It is well known that the median filter does not provide sufficient smoothening of non-impulsive noise. Therefore, Gaussian and 'salt and pepper' noise were added to the video which was then processed by the algorithm. The Fig a, b show the performance of the adaptive median filter.

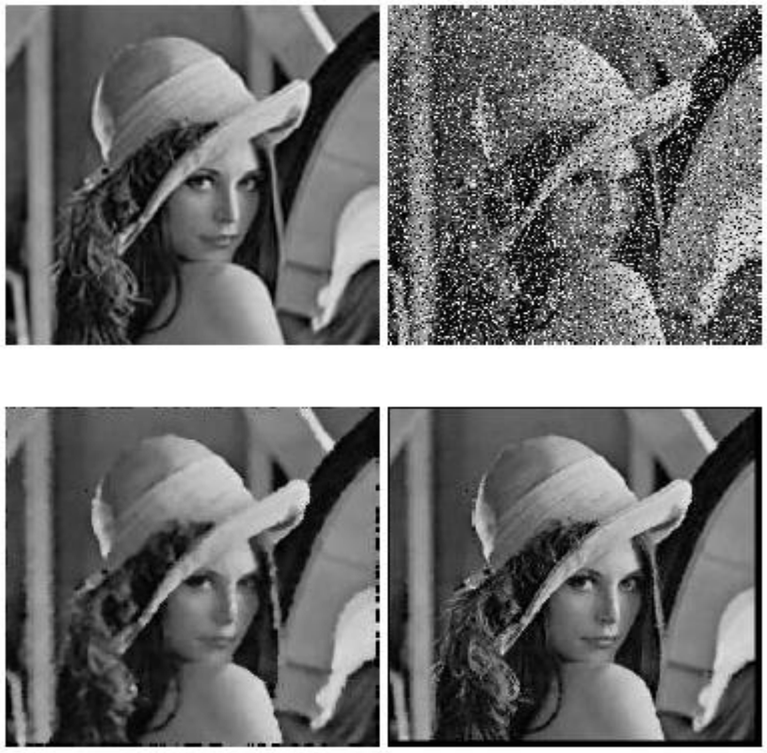


Fig 4 : Results of filtering with a 3*X*3 median and conditional median filter. From left to right, first row: original Image, noisy image; second row: standard median filter, Adaptive median filter.

## VIII. CONCLUSION

The architecture is pipelined which processes one pixel per clock cycle, thus to process an image of size *256* x 256 it requires 0.65 ms when a clock of 100 MHz is used and hence is suitable for real time applications The adaptive median filter successfully removes impulsive noise from images. It does a reasonably good job of smoothening images that contain non-impulsive noise. Overall, the performance is as expected and the successful implementation of the adaptive median filter is presented. Specifically, the project requirements include achieving throughput suitable for real-time video, reducing area as needed for implementation in the given FPGA, and producing a noticeable reduction in the artifacts present in the input frame of video.

## REFERENCES

[1] Zdenek Vasicek, Lukas Sekanina, Novel Hardware Implementation of Adaptive Median Filters 978-1-4244-2277-7/08/ ©2008 IEEE

[2] Olli Vainio, Yrjö Neuvo, Steven E. Butner, *A Signal Processor for Median-Based Algorithms*, IEEE Transactions on Acoustics, Speech, Processing VOL 37. NO. 9, September 1989.

[3] V.V. Bapeswara Rao and K. Sankara Rao, *A New Algorithm for Real-Time Median Filtering*, IEEE Transactions on Acoustics, Speech, Processing VOL ASSP-34. NO. 6, December 1986.

[4] M. O. Ahmad and D. Sundararajan, *Parallel Implementation of a Median Filtering Algorithm*, Int. Symp. on Signals and Systems, 1988.

[5] Dobrowiecki Tadeusz, *Medián Szűrők*, Mérés és Automatika, 37. Évf., 1989. 3.szám

[6] Xilinx Foundation Series Quick Start Guide, 1991-1997. Xilinx. Inc.

## AUTHORS

**First Author** – Dr.Ch. Ravikumar, Head Dept.of ECE, Prakasam Engineering College, Kandukur

**Second Author** – Dr. S.K. Srivatsa, Ph.D (Engg.), Senior Professor, St. Joseph College of Engg, Chennai, India