DESIGN OF NOVEL ARCHITECTURE OF FILTER TO REMOVAL OF NOISE IN DIGITAL IMAGE

C Kumar Charle Paul
Indus college of Engg
Coimbatore

K Thirunadana Sikamani
Prof&HeadCSE
St.Peter's University

N Kirubananda sarathy
Asst Prof ECE Dept
St.Peter's University, Chennai

Abstract

Evolvable hardware (EH) is a new field about the use of evolutionary algorithms (EA). It brings together reconfigurable hardware, artificial intelligence, fault tolerance and autonomous systems. Evolvable hardware refers to hardware that can change its architecture and behavior dynamically and autonomously by interacting with its environment This paper represent as a new technique for the design of Adaptive Median Filter within an Evolvable hardware framework, using genetic algorithm (GA), aimed at removing the impulse noise from the image and reducing distortion in the image. This implementation aims at reducing the number of generations required to provide time bound optimal filter configuration and to improve the quality of the filter designed. The GA processing and the Evolvable hardware framework is synthesized on Xilinx . In this paper investigates a high-speed non-linear Adaptive median filter implementation is presented. Then Adaptive Median Filter solves the dual purpose of removing the impulse noise from the image and reducing distortion in the image. Adaptive Median Filtering can achieve the filtering operation of an image corrupted with impulse noise .

1 Introduction

Reconfigurable hardware devices make it possible to change the topology of electronic circuits at runtime. Using reconfigurable devices as a platform for Evolvable hardware (EHW) is well suited for real-time adaptive systems. EHW is a scheme inspired by natural evolution, for automatic design of hardware systems. It refers to hardware that can change its architecture and behavior dynamically and autonomously by interacting with its environment. It is built on top of reconfigurable logic devices, whose architecture can be reconfigured by using Evolutionary Algorithms (EA). The reconfigurable device acts as the design space for the EA, which then

Issn.No:0975-0932

determines the optimum hardware configuration required for a particular design specification. EHW is best suited for cases where the design specification doesn't provide sufficient information to permit using conventional design methods [2]. For example, the specification may only state desired behavior of the target hardware. In other cases an existing circuit must adapt, i.e. modify its configuration, to compensate for faults or perhaps a changing operational environment. For instance, deep-space probes may encounter sudden high radiation environments and alter a circuit's performance; the circuit must self-adapt to restore as much of the original behavior as possible and quickly. In this work, a reconfigurable median filter constitutes the backbone of the Noise redection. A genetic algorithm based implementation of meadin filter to cancel out the interference from the varied noise sources and abstract the original signal is presented in this work. Both the filter as well as the hardware required for evolution is implemented in a single Field programmable gate array (FPGA). The circuit is based on context-switching in FPGA-devices and preliminary results indicate the use of a compact hardware as well as fast adaptation. The proposed evolvable architecture for ANC is very effective, resulting in significant improvement in terms of reproduced signal quality. The entire system is synthesized on a Xilinx Virtex XCV1000 FPGA. As the entire filter, including the FIFO, and the delay elements, are realized on the reconfigurable fabric, a more optimum filter is realized (using fewer resources) for a given frequency response. Median filtering is a powerful instrument used in image processing. The traditional median filtering algorithm, without any modifications gives good results. There are many variations to the classical algorithm, aimed at reducing computational cost or to achieve additional properties. Median filters are used mainly to remove salt-and pepper noise. Doing this, they preserve edges in the image (preserve their location and do not affect their steepness, unlike Gaussian filters), but unfortunately median filtering may destroy small features in the image. A way to avoid it is to apply center-weighted median filtering instead of a plain median, but the drawback of this solution is the detoriation of the filter’s ability to suppress impulse noise. Common drawback of various kinds of the median filtering is their computational cost. Computing a two-dimensional median for an *NxN* window, requires sorting of *NxN* elements for every image pixel and choosing the median value for the

Issn.No:0975-0932

output. After the sorting, each queue element is assigned a value called a rank, specifying its position in the queue as a result of sorting.

Therefore, using median filtering in any real-time vision system requires a significant computational power. One way to speed up the computations is to implement the algorithm in hardware, e.g. with the help of FPGA circuits. The rationale behind this is to use the FPGA's inherent ability to execute operations in parallel. Moreover, programmable logic creates the possibility to tailor the implementation to the user's needs. All this results in a significant speedup over the software implementations by using sequential processors. One drawback of hardware-based algorithm development is the complexity of the design process as implementing algorithmically complex operations is very difficult. Median filtering, like many other low-level image processing algorithms is fairly simple - the main problem in this case is the amount of data to handle.

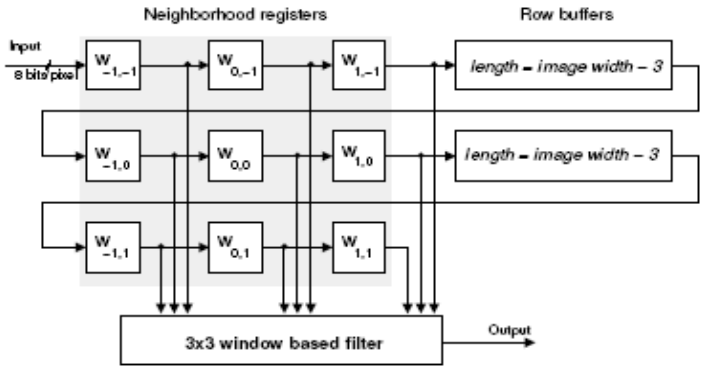

Fig 1 Adaptive Median Filter

2. Adaptive Median Filter

The 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,

Issn.No:0975-0932

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. Thus, the Adaptive Median Filter solves the dual purpose of removing the impulse noise from the image and reducing distortion in the image. Adaptive Median Filtering can handle the filtering operation of an image corrupted with impulse noise of probability greater than 0.2. This filter also smoothens out other types of noise, thus, giving a much better output image than the standard median filter as Shown in Fig1 .

3. Evolvable Hardware

3.1 Evolvable Hardware Concepts

Evolvable hardware is based on the idea of combining reconfigurable devices with evolutionary

algorithms such as Genetic Algorithms [3,4]. The basic concept in EHW is to regard the configuration bits for reconfigurable hardware devices as chromosomes for GA. By choosing an appropriate fitness function for the given task, GA can autonomously find the best hardware configuration in terms of chromosomes i.e. configuration bits. The algorithm for evolving circuits on a reconfigurable fabric is shown in Fig. 2

Issn.No:0975-0932

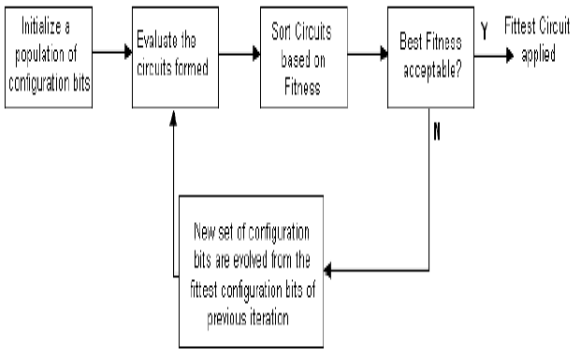

Fig 2 Block Diagram of EHW

Evolvable hardware problems fall into two categories: original design and adaptive systems. Original design uses evolutionary algorithms to design a system that meets a predefined specification. Adaptive systems reconfigure an existing design to counteract faults or changed operational environment. Original design of digital systems is not of much interest because industry already can synthesize enormously complex circuitry. For example, one can buy IP to synthesize USB port circuitry, Ethernet microcontrollers and even entire RISC processors. Some research into original design still yields useful results, for example genetic algorithms have been used to design logic systems with integrated fault detection that out perform hand designed equivalents. Original design of analog circuitry is still a wide-open research area. Indeed, the analog design industry is nowhere near as mature as is the digital design industry. Adaptive systems have been an area of intense interest in the recent past.

The fitness of an evolved circuit is a measure of how well the circuit matches the design specification. Fitness in evolvable hardware problems is determined via two methods:-

1) Extrinsic evolution: All circuits are simulated to see how they perform

2) Intrinsic evolution: Physical tests are run on actual hardware.

Issn.No:0975-0932

In off-line fitness computation (OFL) or Extrinsic evolution, the evolution is simulated in software, and only the elite chromosome is written to the hardware device. In online Fitness Computation (ONL), the hardware device gets configured for each chromosome for each generation (sometimes named intrinsic evolution). GA is the most commonly used evolutionary algorithm and uses biological operators like crossover and mutation .

4. 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 as Shown in Fig 3 & 4 .

Issn.No:0975-0932

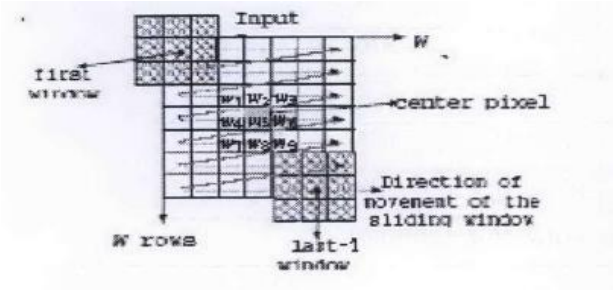

Fig.3 Moving Window Architecture

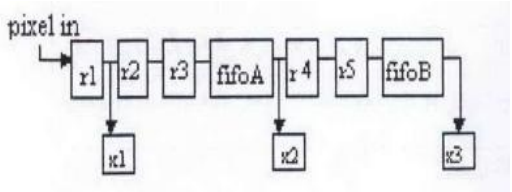

Fig. 4 Reading Pixels from Window.

5. Parallel Sorting strategy:

To make a fair comparison of the parallel sorting strategy against wave sorter strategy in terms of the total number of required steps to sort an array, it is necessary to consider the steps used to read data from memory and the steps required to store the sorted data back to memory. The proposed approach is based on the same structure of the registers array used in the wave sorter strategy. With this kind of array, data can be stored in the array by sending a datum to the first register and later, when the second datum is sent to the first register, the value on the first array is shifted to the second register. Thus, for every datum sent to the array to be stored, values in registers are shifted to their respective adjacent registers. This process requires *n* steps. The same number of steps is required to take data out from the array as Shown Fig 5. This approach allows storing a new set of data in the array while the previous set is being sent back into the memory. As mentioned in section 2, suffix sorting might imply more than one sorting iterations. If *k* sorts are required, then the parallel sorting requires to $((n+n/2) * k + n)$ to sort an array of *n* data. Thus total number of steps required can be obtained by the following equation:

Issn.No:0975-0932

$$f_{steps}^{PS}(n,k) = n\left(\frac{3}{2}k+1\right)$$

The parallel strategy leads to a significant reduction compared to the wave sorter approach. Furthermore, in additional sorts the necessary number of steps for sorting is equal to the number of characters in the biggest group of identical characters divided by 2 (remember that an additional sorting is implied if groups of identical adjacent characters appear in the array). This implies that in practice, it is possible to reduce more than the number of steps to solve the suffix problem.

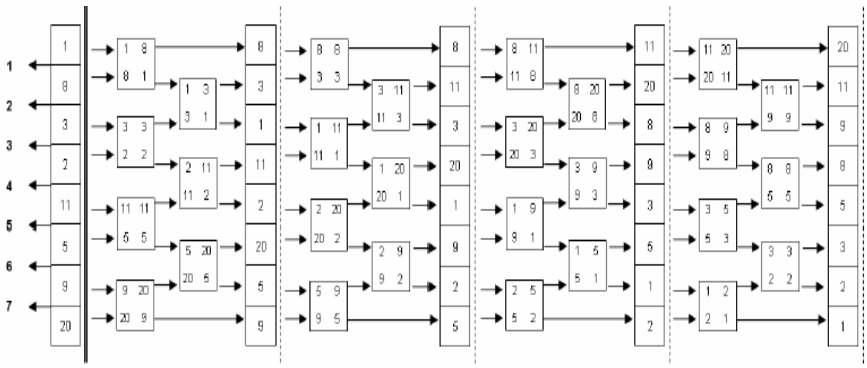

Fig 5.Parallel sorting with two levels of comparators performance

6. Implementation and Testing:

The adaptive filter works on a rectangular region $\mathbf{S_{xy}}$. The adaptive median filter changes the size of $\mathbf{S_{xy}}$ during the filtering operation depending on certain criteria as listed below. The output of the filter is a single value which the replaces the current pixel value at (x, y), the point on which $\mathbf{S_{xy}}$ is centered at the time. The following notation is adapted from the book and is reintroduced here:

$\mathbf{Z_{min}}$ = Minimum gray level value in $\mathbf{S_{xy}}$.

$\mathbf{Z_{max}}$ = Maximum gray level value in $\mathbf{S_{xy}}$

$\mathbf{Z_{med}}$ = Median of gray levels in $\mathbf{S_{xy}}$

Issn.No:0975-0932

$\mathbf{Z_{xy}}$ = gray level at coordinates (x, y)

$\mathbf{S_{max}}$ = Maximum allowed size of $\mathbf{S_{xy}}$

The adaptive median filter works in two levels denoted Level A and Level B as follows:

Level A: A1= $\mathbf{Z_{med} - Z_{min}}$

A2= $\mathbf{Z_{med} - Z_{max}}$

If A1 > 0 AND A2 < 0, Go to level B

Else increase the window size

If window size <= $\mathbf{S_{max}}$ repeat level A

Else output $\mathbf{Z_{xy}}$.

Level B: B1 = $\mathbf{Z_{xy} – Z_{min}}$

B2 = $\mathbf{Z_{xy} – Z_{min}}$

If B1 > 0 And B2 < 0 output Z_{xy}

Else output Z_{med}.

The algorithm has three main purposes:

(a) To remove ‘Salt and Pepper’ noise.
(b) To smoothen any non impulsive noise.
(c) To reduce excessive distortions such as too much thinning or thickening of object boundaries.

7. RESULT:

Two signals were considered for the test. These were subjected to salt and pepper noise of unit amplitude. one form of the evolved architecture of the reconfigurable fabric during operation, give the deviations of the output with respect to the original value of the signals. The

Issn.No:0975-0932

absolute error values of the filter for the 2 signals.The adaptive median filter is designed to remove impulsive noise from images. Therefore, our algorithm's performance was first tested with basic salt and pepper noise. 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 image which was then processed by the algorithm. The Fig 6 show the performance of the adaptive median filter.

Fig 6.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.

8. CONCLUSION

Issn.No:0975-0932

The proposed architecture is designed to reconfigure itself and provide real-time noise reduction. The filter logic is implemented on a novel reconfigurable fabric designed for the specific purpose of implementing an FIR filter using primitive operators, and is synthesized on a Xilinx. The results obtained show the validity of the approach to adaptive noise reduction using Evolvable Digital Filters. The filter was able to track signal variations, and retrieved the signal data with minimal error. The architecture provides the capability of implementing the reconfigurable fabric in a pipelined fashion and Also includes the implementation of the pipelined version of the filter for improved speed, and to optimize the programmable processing element at circuit level for efficient ASIC implementation of the reconfigurable fabric. The propose architecture , which is processed one pixel per clock cycle, thus to process an image of size *256* x 256 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. Overall, the performance is as expected and the successful implementation of the adaptive median filter is presented.

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.
[7] Jim Torresen, "An Evolvable Hardware Tutorial", FPL 2004,821-830
[8] [2]. L. Sekanina, "Evolvable Hardware Tutorial", in GECCO 2007, New York
[9] . Bernard Widrow and Samuel D. Steavns, "Adaptive Signal Processing", Pearson Edition, 2000.

Issn.No:0975-0932

[10]. Redmill, D. W., Bull, D. R., and Dagless, E., "Genetic synthesis of reduced complexity filters and filter banks using primitive operator directed graphs". *IEE Proc. Circuits Devices Syst,* vol.147, pp. 303-310, 2000. Bull, D. R. and Horrocks, D. H., "Primitive operator digital filters", IEE Proc. Circuits, Devices and Systems, pp. 401-412, 1991.

[11]. L. Sekanina and P. Mikusek, "Analysis of Reconfigurable Logic Blocks for Evolvable Digital Architectures", EvoWorkshops 2008, LNCS 4974, pp. 144–153, 2008.

Evolution Hardware (EH'05)

Issn.No:0975-0932