

PROGRAMMABLE HARDWARE IMPLEMENTATION FOR THE MEDIAN-RATIONAL HYBRID FILTERS

G. Bernacchia (1), L. Khriji (2), M. Gabbouj (3), G. L. Sicuranza (1)

(1) DEEI, University of Trieste, Italy.

(2) Electrical Engineering Department, E.N.I.M., Tunisia.

(3) Signal Processing Laboratory, TUT, P.O. Box 553 FIN-33101 Tampere, Finland.

ABSTRACT

The Median-Rational Hybrid Filter (MRHF) has been recently introduced [1][2] as a new class of nonlinear filters and successfully applied to image filtering problems. The main characteristics of the MRHF are the good noise attenuation property in smooth areas and the preservation of edges and details. In fact, in changing areas the noise attenuation is traded for a good response to the change. Moreover, the median filters effectively remove the impulsive noise while the rational filter performs well in relatively high SNR Gaussian contaminated environments. Another relevant characteristic is that MRHF's usually act on small windows and thus require a reduced number of operations, resulting in simple and fast filtering structures. In this paper we present a programmable hardware implementation for the MRHF's. Exploiting the features of the dynamic logic families it is possible to achieve high speed and compactness, while keeping the power dissipation very low.

Keywords: Median Filters, Rational Filters, Hardware Implementation, Dynamic Logic.

1. INTRODUCTION

Noise reduction is a classical problem in image restoration. Various filtering techniques have been proposed to remove or at least reduce the noise which is usually present in the real-world images due to different causes [3]. In respect to such a task, nonlinear digital techniques have shown their superior capabilities in comparison to linear approaches.

In this paper we present a programmable hardware implementation of the nonlinear Median Rational Hybrid Filters (MRHF's), recently introduced in [1] [2].

One component of the MRHF's is a rational operator, i.e. a ratio of two polynomials. There are several advantages in the use of this operator. In fact, a rational function can approximate any continuous function arbitrarily well as a polynomial function does, but achieving a desired level of accuracy with a lower complexity.

The second component of the MRHF's is a set of median filters which are effective in removing impulsive noises. Therefore, the MRHF uses a two-step approach to remove both impulsive and Gaussian noises from an image. Its structure contains two median filters (MF) and one center weighted median filter (CWMF); the outputs of these filters are the input to a rational filter whose input-output relationship is a simple rational function. Therefore, while the median filters effectively remove the impulsive noise, the rational filter [4] offers good performances in relatively high SNR Gaussian contaminated environments. When both impulsive and Gaussian noises spoil the images the MRHF outperforms other nonlinear filters because it exploits at the same time the features of both median and rational filters.

2. THE MEDIAN-RATIONAL HYBRID FILTER

Definition 2.1 *The output of the MRHF is the result of a rational function that takes into account as inputs three median filters which form an input function set $\{\Phi_1, \Phi_2, \Phi_3\}$. The function Φ_2 is a center weighted median filter whereas the masks of the filters Φ_1 and Φ_3 are chosen so that an acceptable compromise between noise reduction and edge preservation can be achieved.*

The input-output relationship of the MRHF can be written as follows:

$$y(n) = \Phi_2(n) + \frac{\sum_{i=1}^3 \alpha_i \Phi_i(n)}{h + k(\Phi_1(n) - \Phi_3(n))^2} \quad (1)$$

While different choices are possible, here we use a very simple prototype filter with coefficients $\alpha = [1, -2, 1]^T$, satisfying the condition $\sum_{i=1}^3 \alpha_i = 0$. The (positive) parameters h and k are used to control the nonlinear effect of the rational operator. Their values are user-specified and depend on the the applications.

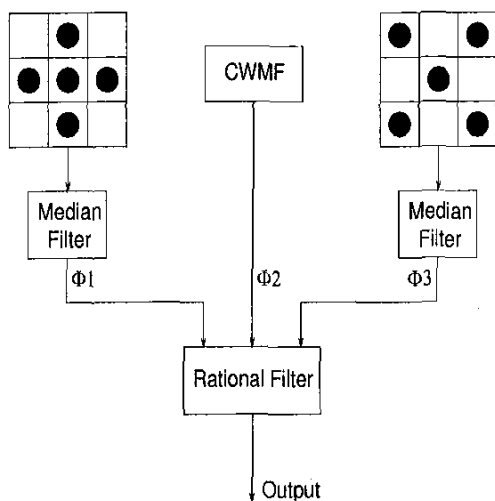


Figure 1: Structure of MRHF using two median filters and a center weighted median filter.

The complete structure used for the MHRF is shown in Figure 1. The functions Φ_1 and Φ_3 are obtained through two median filters acting respectively on a plus-shaped and on a cross-shaped mask. Since median filters have well-known efficient software and hardware implementations, we limit here our discussion to the hardware realization of the rational operator.

3. A PROGRAMMABLE HARDWARE IMPLEMENTATION

In this paper we propose a programmable implementation of the MRHF. In fact, the parameters used in the rational operator are not the results of a previous optimization procedure. Having a programmable structure allows the user to explore different combinations and find the best trade-off between noise attenuation and edge preservation among the tested operators.

In the last few years the reduction of the power dissipated by an integrated circuit has been assuming a very important role as driving constraint for many designs. This is even more true if applied to the fields of communications and multimedia, where power saving is dictated by the need of having portable systems. Moreover high speed is a dominant factor in real time applications, as image and video processing.

A number of new logic families have been introduced (see [5], [6] for an overview) in order to improve the performances of CMOS circuits.

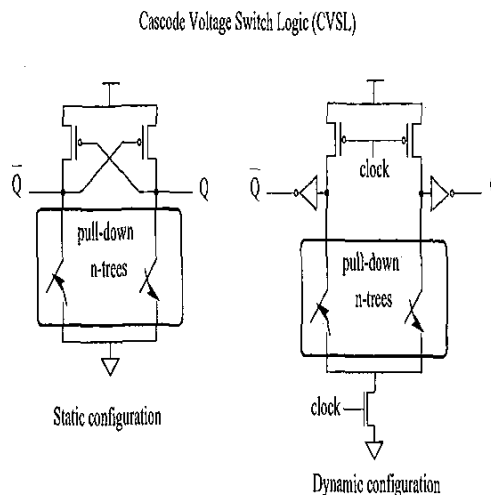


Figure 2: Differential Cascode Voltage Switch Logic: basic configurations.

In this work we decided to use the Differential Cascode Voltage Switch Logic (DCVSL) in the dynamic configuration. In Figure 2 the static and dynamic configurations of this family are shown. The static configuration consists of a push-pull p-MOS load and a pair of inter-related binary decision trees made only of n-type MOS. The two decision trees are complementary, so that when the logic function corresponding to one of them is true the other is false and viceversa. The cross connection of the p-loads introduces a positive feedback, which allows the switching of the outputs. The main advantage of this logic is that it usually requires fewer transistors than a standard NAND/NOR two level implementation. This is accomplished using logic minimization and sharing subfunctions between the two decision trees.

The introduction of the dynamic configuration allows a further improvement in the performances in terms of

power reduction. When the system clock is at logic 0 the decision trees are disconnected from ground and the outputs are pulled up. When the clock is at logic 1 the decision trees are connected to ground, allowing one of the output to switch to 0 if the corresponding function is true, while the p-loads are switched off. Therefore like in the standard CMOS logic there is no direct path from ground to the power supply but for a short time during the switching.

Another advantage of the DCVSL family is that for particular functions multiple outputs can be obtained from the same gate, thus allowing a further gain in size.

A dynamic logic can be exploited very well in pipelined systems, because it is possible to avoid the need of latches or flip-flops like for static logic. On the other side this extreme pipelining means a longer latency time, if we keep constant the clock frequency.

For our implementation we chose to use the standard binary representation for the median filters and the Residue Number System (RNS) for the implementation of the rational function.

An RNS system is defined by a set of *moduli* m_1, m_2, \dots, m_n . An integer a can be represented in this system with an n -uple of integers $a = (a_1, a_2, \dots, a_n)$ where a_i are the rest of the integer division of a by m_i .

The number of positive representable integers M is equal to $\prod_{i=1, \dots, n} m_i$. To represent both negative and positive values, then we have:

$$a_i = \begin{cases} a \bmod m_i & \text{if } a \geq 0 \\ (M + a) \bmod m_i & \text{if } a < 0. \end{cases}$$

As a consequence, the range of representable integers is $[-\frac{M}{2}, \frac{M}{2}]$, if M is even and $[-\frac{M-1}{2}, \frac{M-1}{2}]$, if M is odd.

In our specific application we used three modules: $m_1 = 15, m_2 = 31$ and $m_3 = 32$, and thus we can represent integers in the range $[-7440, 7440]$. We can see that the result of the rational operator in (1) uniquely depends on the sign of the numerator. Therefore we can simply store this sign bit and perform all the operations with unsigned (positive) numbers. This solution allows us to double the dynamic range and then to reduce the need of a scaling.

The advantage of the RNS system is that operations like addition/subtraction and multiplication are split in many blocks (one per module) and evaluated in parallel. Using the given set of moduli each residue a_i can be represented at most with 5 bits. Exploiting the features of the DCVSL logic the modular adders can be designed quite effectively. In fact, for a 5 bit adder the carry generating block can be designed as a single gate with multiple outputs using the equations for a carry lookahead adder. In this way we can partially avoid the drawback of the modular addition, which requires a carry correction whenever the chosen modulo is not a power of two.

Exploiting the short length of the operands the multipliers can be implemented with the shift-and-add algorithm. For each module just three adders are required and the result is exact.

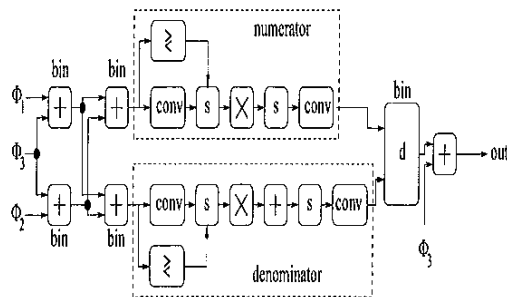


Figure 3: Block diagram of the implementation of the rational function.

In Figure 3 the block diagram of the circuitry for the rational function is shown.

Let us now briefly describe the implemented functions. The fundamental problems using the RNS binary system are scaling, division and conversion.

The maximum absolute value representable with the afore mentioned set of moduli is 14880, therefore it clearly appears that a scaling algorithm is necessary in order to calculate the numerator and denominator in (1). To this purpose we developed a simple scaling algorithm converting the RNS representation into the so called Mixed Radix Representation (MRS). Given the set of modules $\{m_i\}_{i=1, \dots, n}$, an integer a can be represented in the MRS in the following way:

$$a = \alpha_1 + \alpha_2 m_1 + \alpha_3 m_1 m_2 = (\alpha_1, \alpha_2, \alpha_3) \quad (2)$$

where α_i can be directly obtained from the residues m_i . If we scale a by m_1 we can write:

$$a^1 = \frac{a}{m_1} = \frac{\alpha_1}{m_1} + \alpha_2 + \alpha_3 m_2$$

If we consider $\frac{\alpha_1}{m_1}$ as a rounding correction coefficient c , we can write:

$$a^1 \bmod m_1 = (c + \alpha_2 + \alpha_3) \bmod m_1 \quad (3)$$

$$a^1 \bmod m_2 = (c + \alpha_2) \bmod m_2 \quad (4)$$

$$a^1 \bmod m_3 = (c + \alpha_2 - \alpha_3) \bmod m_3 \quad (5)$$

In this design the role of the division is the most critical, since this operation is highly time and size consuming. Therefore we developed a new iterative division algorithm, described in a companion paper [8], which can be implemented both in RNS and in standard binary arithmetic. Since it involves a few comparisons, we choose to implement the algorithm in standard binary arithmetic. In almost all our experiments, after 4 iterations the result obtained differs less than 1% from the theoretical value. The final tests on images showed that 3 iterations are enough to obtain a good result and this allows a good save in terms of size.

One of the main problems of non standard arithmetic systems is the conversion from binary to the required system and vice-versa. In our implementation the conversion from binary to RNS reduces to a simple RNS addition exploiting the features of module's systems of type $(2^k - 1, 2^k, 2^k + 1)$. For $m = 2^k$ the conversion is simply accomplished considering the first k LSB's of the number. For the other two modules the word is split respectively in blocks of 4 and 5 bits and then the blocks are added.

The conversion from RNS to binary is slightly more difficult and usually a conversion to MRS is exploited. In this case the final conversion is obtained as sum of products like in (2) and the output is represented with 13 bits.

4. EXPERIMENTAL RESULTS

The programmable hardware median-rational hybrid filter has been tested using images taken from the TUT database [9] corrupted with i.i.d. noise having the following probability distribution:

$$\nu = (1 - \lambda)\mathcal{N}(0, \sigma_n) + \lambda\mathcal{N}(0, \frac{\sigma_n}{\lambda}) \quad (6)$$

Three values of λ and two different values for the SNR have been chosen: $\lambda = 0.1$ (very impulsive noise), $\lambda = 0.2$ (mixed Gaussian-impulsive noise) and $\lambda = 1$ (purely Gaussian noise), with SNR= 3dB and 9dB.

As an example, Figures 4 (a-d) are respectively the clean Bridge image, the noisy image with $\lambda = 0.2$, SNR = 9dB, the output of the implemented MRHF system and the filtered image by theoretical MRHF.

It can be clearly seen that the images in Figures 4 (c) and (d) are very close to each other, and both show a good noise attenuation and present sharper edges and smoother flat areas.

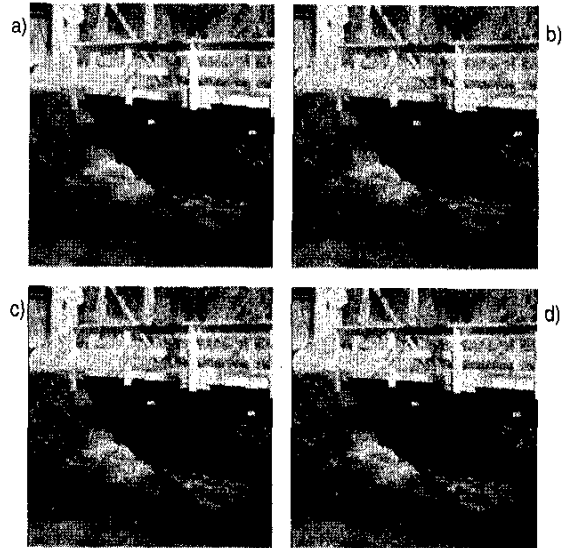


Figure 4: Qualitative comparison between the results of the implemented and theoretical filters: a) original image, b) noisy image, c) image filtered with the implemented filter, d) image filtered with the theoretical filter.

The use of high speed and low power logic families allows us to obtain very good performances. The speed requirements for real-time applications are variable depending on the size of the images and on the frame rate. For images of 1024 by 768 at a frame rate of 50 Hz a frequency of at least 40MHz is required in order to perform all the operations. The results obtained with the DCVS logic show that the rise and fall time for a complex gate are around 1-1.5 ns. For instance the gate performing the calculation of the carry chain has a rise time of 0.938 ns and a fall time of 1.377 ns. Such characteristics make possible to match the real-time processing requirements.

A further promising development would be the implementation of such kind of structure using an asynchronous architecture [10].

5. CONCLUSIONS

In this paper we have presented a hardware implementation of the median-rational hybrid filter, which is able to remove different kinds of additive i.i.d. noises. Our implementation uses the standard binary representation

for the median filters and the Residual Number System for the the rational operator. A dynamic logic based on the Differential Cascode Voltage Switch Logic has been chosen for the actual hardware realization according to the characteristics of low power consumption and high processing speed. Our tests have shown that the implemented system is very robust and that its features make it very suitable for real-time applications.

6. REFERENCES

- [1] L. Khriji and M. Gabbouj, "Median-Rational Hybrid Filters for image restoration", *IEE Electronics Letters*, vol.34, no.10, pp.977-979, May 1998.
- [2] L. Khriji and M. Gabbouj, "Median-Rational Hybrid Filters", *Intern. Conf. on Image Processing ICIP'98*, Chicago, Illinois, USA. October 4-7, 1998.
- [3] I. Pitas and A.N. Venetsanopoulos, "Nonlinear Digital Filters: Principles and Applications", *Kluwer Academic*, Dordrecht, Holland, 1991.
- [4] G. Ramponi, "The Rational Filter for Image Smoothing", *IEEE Signal Processing Letters*, vol.3, no. 3, pp. 63-65, March 1996.
- [5] N.H.E. Weste, K. Eshraghian, "Principles of CMOS VLSI Design", *Addison-Wesley Publishing Company*, 1994.
- [6] Kan M. Chu, David L. Pulfrey, "Design Procedures for Differential Cascode Voltage Switch Circuits", *IEEE Journal of Solid-State Circuits*, vol.21, no.6, Dec. 86.
- [7] Mi Lu, Jen-Shinu Chiang, "A Novel Division Algorithm for the Residue Number System", *IEEE Transactions on Computers*, vol.41, no.8, Aug 92.
- [8] L. Khriji, G. Bernacchia, M. Gabbouj and G. Sicuranza, "A Dedicated Hardware System for a Class of Nonlinear Order Statistics Rational Hybrid Filters with Applications to Image Processing", *Proc. ICIP-99*, Kobe, Japan, Oct. 25-28, 1999.
- [9] M. Gabbouj and I. Tabus, "TUT noisy image database", *Technical Report, no. 13, Tampere University of Technology*, Dec. 1994.
- [10] G. M Jacobs, R. W. Brodersen, "A Fully Asynchronous Digital Signal Processor Using Self-Timed Circuits", *IEEE Journal of Solid-State Circuits*, vol.25, no.6, Dec. 90.