RNS Assisted Image Filtering and Edge Detection

Evangelos Vassalos, Dimitris Bakalis
Electronics Laboratory, Department of Physics
University of Patras
Patras, Greece
e-mail: vassalos@upatras.gr, bakalis@physics.upatras.gr

Haridimos T. Vergos
Computer Engineering and Informatics Department
University of Patras
Patras, Greece
e-mail: vergos@ceid.upatras.gr

Abstract— **Image edge detection plays a fundamental role in image processing as well as in computer and machine vision while when applied in medical images can improve medical diagnosis. Thus, efficient hardware implementation of an edge detection system is highly appreciated. In this work we propose a novel architecture for image filtering and edge detection using the Residue Number System (RNS). A VLSI implementation of the proposed architecture dictates that the employment of RNS for the arithmetic processing leads to small hardware complexity and high operation frequency.**

Keywords—Residue Number System; Image filtering; Image smoothing; Edge detection; VLSI implementation;

I. Introduction

Edge detection [1] is very frequently employed as a first step in machine and computer vision applications in order to produce input data for processes such as pattern recognition and matching [2], motion estimation [3], feature extraction [4], object tracking [5] etc. It is also important in medical image processing since it can be used in detecting pathological deformations or in object recognition of human organs [6-8].

Most classical edge detection methods are based on calculating the derivative of the pixel's intensity values across the image. Edges are usually met in areas where the derivative's value is peaked. The first order derivative of choice is the gradient. Typical examples of edge detection methods, based on two-dimensional gradient calculations on the spatial domain, are the Robert's, Prewitt's and Sobel's with the latter considered the one offering the best performance among them. Another edge detection technique is the LoG edge detection (or Laplacian of Gaussians) which is defined as the second order derivative of an image and yields in a double-edge image [9]. The most efficient edge detection technique is the one proposed by Canny [10]. It comprises of four main steps: (i) convolving the image with a Gaussian function in order to smooth the image, (ii) calculating the luminosity and the direction of the gradient with the Sobel vertical and horizontal edge detectors, (iii) applying non-maximum suppression, that is rejection of the values of the pixels whose magnitude is not maximum on the orientation of the gradient, and (iv) applying hysteresis thresholding in order to classify the remaining edges as definite or possible edges. This technique however is much more complex, more time consuming and also requires high hardware resources compared to Sobel's. In order to keep the complexity low without downgrading the edge detection quality, only the first two steps of the Canny's technique are considered in this work, that is Gaussian smoothing and edge detection based on Sobel's operators.

Systems that implement image processing algorithms require the realization of a large number of arithmetic operations. Residue Number System (RNS) [11] can be considered for these systems since it is a carry-limited number system commonly adopted for speeding-up computations in digital signal and image processing [12-18] in cryptography and communication units. An RNS is characterized by a set $\{m_1,\ldots, m_p\}$ of p moduli that are pair-wise relatively prime. An integer A, with $0 \le A < DR$, where $DR = m_1 \times \ldots \times m_p$, has a unique representation in the RNS given by the set of residues $\{a_1,\ldots, a_p\}$, where $a_i = |A|_{mi}$ is the least non-negative remainder of the division of A by m_i, with i=1,…,p. An operation $\otimes$ (typically, addition, subtraction or multiplication) over an RNS is defined as $(z_1,\ldots, z_L) = (a_1,\ldots, a_L) \otimes (b_1,\ldots, b_L)$, where $z_i = |a_i \otimes b_i|_{m_i}$. An RNS-based system consists of three main blocks: (a) binary-to-residue conversion, according to the specified moduli set, (b) arithmetic processing in each modulo channel, and (c) residue-to-binary conversion.

RNS channels with moduli of the 2^n, 2^n-1 or 2^n+1 forms have received significant attention mainly because efficient arithmetic circuits have been proposed for them [19-26]. Furthermore, efficient converters exist for various moduli sets [22] [27-29]. In such moduli sets, the 2^n+1 channel deals with operands one bit wider than the corresponding 2^n-1 or 2^n channels, leading to a performance bottleneck. To avoid this, [30] introduced the diminished-one representation according to which each operand is represented decreased by one compared to its normal representation and hence only n bits are used in the arithmetic processing leading to faster implementations [23-26]. A separate bit is utilized to indicate a zero operand or result [21]. To this end, the diminished-one representation for the modulo 2^n+1 channel is considered in this work.

Although some research efforts have been done on image filtering and edge detection using RNS [16-18], they present only software simulation results without any hardware architectures or implementations. In this work we focus on the hardware implementation side and present a novel RNS-based architecture for implementing Sobel's edge detection method with a Gaussian smoothing filter applied prior to it, in order to suppress the noise that exist in the image and improve the final edge detection result. A bit-accurate MATLAB model has been developed in order to provide visual results of the proposed architecture while implementation results in a CMOS 32nm technology have been derived.

The remaining of the paper is organized as follows. The next section provides preliminary information whereas section III introduces and details the proposed RNS-based filtering and edge detection architecture. Section IV presents experimental results and evaluation and the last section concludes the paper.

II. Preliminaries

Consider an image I composed of R rows and C columns of grayscale pixels. Let $I(x,y)$ denote the image pixel at position (x,y), with $0 \le x < C$ and $0 \le y < R$. The output, I_f, of a linear filter in position (x,y) in the spatial domain is given by:

$$I_f(x,y) = \sum_{i=0}^{d-1}\sum_{j=0}^{d-1} W_{i,j} I(x+i, y+j) \quad (1)$$

where $W_{i,j} = \begin{bmatrix} w_{0,0} & \cdots & w_{0,d-1} \\ \vdots & \vdots & \vdots \\ w_{d-1,0} & \cdots & w_{d-1,d-1} \end{bmatrix}$ is a $d \times d$ square matrix with $w_{i,j}$ being the kernel's weights [31]. In order to acquire the whole filtered image, I_f, the window W should slide throughout the input image (see Fig. 1 for d=3). As a result there is a need to cache the input pixels in order to synchronize them for the streaming process. For a $d \times d$ window a straightforward solution for caching the input pixels is to employ d-1 row buffers in parallel with the window, as shown in Fig. 2 for d=3. It is evident that in the boundaries of the image, part of the window will fall outside the image, thus a different approach is required. In this work, we adopt a common practice which is to simply ignore the boundary pixels to keep the complexity low. This results in a slightly cropped filtered image.

Noise filtering or smoothing in image processing is a linear operation. If the noise has a uniform distribution and a zero mean value, then the most common filter for eliminating, at some point, this noise is a filter, G, with Gaussian weights $g_{i,j} = \frac{1}{2\pi\sigma^2} e^{(i^2+j^2)/2\sigma^2}$, with σ being the standard deviation of the filter. The window dimension is usually set equal to 4σ.

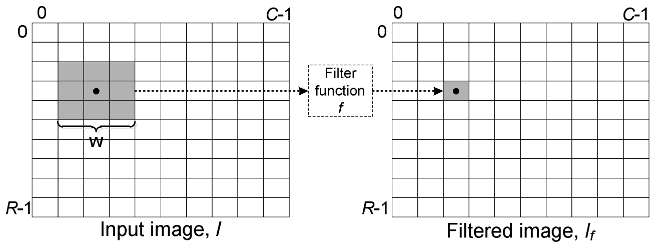

Fig. 1. Filtering an image with a window function f.

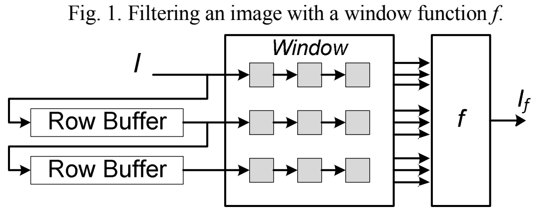

Fig. 2. Caching pixels with row buffers for synchronization.

Edge detection is also a linear filtering operation. Most of the edge detection methods assume that an edge occurs where there is a discontinuity in the intensity function of the image or in other words a steep intensity gradient. The gradient of the image is given by:

$$\nabla I = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right] \quad M = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} \quad \vartheta = \tan^{-1}\left(\frac{\partial I}{\partial x} \Big/ \frac{\partial I}{\partial y}\right) \quad (2)$$

with M and ϑ being the gradient's magnitude and direction respectively. The Sobel edge detection technique is simple and considered to give very reliable results. It approximates $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$ with the $S_v = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$ and $S_h = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$ vertical and horizontal gradient kernels. Its advantages are fast calculation without using excessive resources, small noise suppression and smooth and connective edges [2].

The implementation of (1) at every pixel location, where $W_{i,j}$ is now S_v and S_h, is relatively easy. However realizing M in (2) in hardware involves a square root calculation that can be a challenging task and quite inefficient in terms of both area and delay. This is why it is very often approximated by:

$$M = abs\left(\frac{\partial I}{\partial x}\right) + abs\left(\frac{\partial I}{\partial y}\right) \approx abs(S_v) + abs(S_h) \quad (3)$$

where $abs(x)$ denotes the absolute value of x.

III. The Proposed RNS-Based Edge Detector

We assume grayscale images with 8-bit pixel intensities. We consider an RNS with the 3-moduli set $\{m_1=2^n-1, m_2=2^n, m_3=2^n+1\}$ that has a $2^{3n}-2^n$ dynamic range. Specifically, n=5 is chosen which is adequate for correctly representing all the intermediate results inside the RNS channels. For the modulo 2^n+1 channel the diminished-one representation is used. An operand, $A=A_n...A_0 \in [0,2^n]$, is represented by (A_{d_z}, A_d), where A_{d_z} is a bit that indicates a zero operand when set and $A_d=A-1=A_{d_{n-1}}...A_{d_0}$. Subscripts d and d_z (e.g. A_d, A_{d_z}) will be used hereafter to denote the diminished-one encoding of a modulo 2^n+1 operand.

Fig. 3 presents a generic block diagram of the proposed architecture. Each 8-bit input image pixel is converted to its RNS equivalent $\{|I|_{m_1}, |I|_{m_2}, |I|_{m_3}\}$. After the Binary-to-Residue (BtR) conversion, all the image processing phases are realized in the RNS domain. Firstly, the image smoothing is performed in each modulo channel with the Gaussian kernel $|G|_m$, $m \in \{m_1, m_2, m_3\}$. Then the filtered RNS result is fetched to the next processing elements each one of which implements the edge detection according to the Sobel's kernels $|S_v|_m$ and $|S_h|_m$ and (3). The results of the three modulo channels are combined in order to produce the binary equivalent of the filtered pixel. This is achieved by a Residue-to-Binary converter (RtB). Finally, proper saturation is performed in the attained binary pixel values in order to be represented in the range [0, 255].

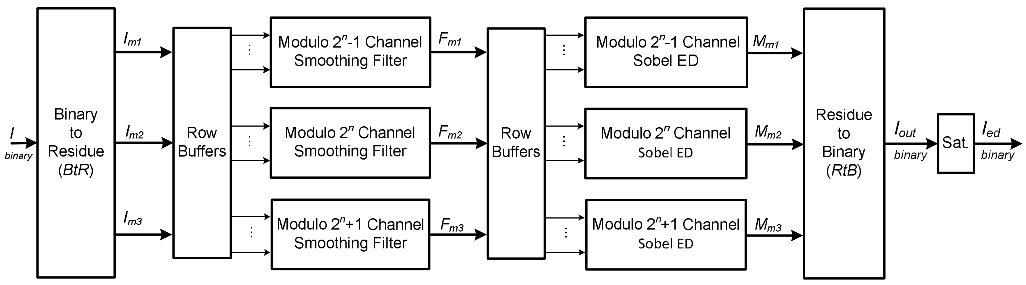
I
binary
Binary to Residue (BtR)
Im1
Im2
Im3
Row Buffers
Modulo 2^n-1 Channel Smoothing Filter
Modulo 2^n Channel Smoothing Filter
Modulo 2^n+1 Channel Smoothing Filter
Fm1
Fm2
Fm3
Row Buffers
Modulo 2^n-1 Channel Sobel ED
Modulo 2^n Channel Sobel ED
Modulo 2^n+1 Channel Sobel ED
Mm1
Mm2
Mm3
Residue to Binary (RtB)
Iout
binary
Sat.
Ied
binary

Fig. 3. The block diagram of the proposed architecture

Both smoothing and edge detection lie on the implementation of (1) throughout the input image's pixels $I(x,y)$ in the chosen RNS, that is:

$$\left|I_f(x,y)\right|_m = \left|\sum_{i=0}^{d-1}\sum_{j=0}^{d-1}\left|W_{i,j}\right|_m \left|I(x+i,y+j)\right|_m\right|_m \qquad (4)$$

where $W_{i,j}$ is either equal to G for the smoothing filter or S_v and S_h for the edge detection process. A generic architecture of the modulo m processing element that implements (4) is shown in Fig. 4, where **D** stands for one standard delay element, RB for the necessary row buffers and $w'_{i,j}(x,y) = \left|w_{i,j}(x,y)\right|_m$ is the representation of the coefficients $w_{i,j}$ in modulo m arithmetic. Since the coefficients are known a priori, there is no need to use 9 generic modulo multipliers in every modulo channel. Instead, single-constant modulo multipliers [25] can be used, which are shown to be very efficient.

The various components of the proposed RNS-based edge detector are analyzed in the following.

A. *Binary-to-Residue Converter (RtB)*

Let I=I_7...I_0, correspond to the intensity of a grayscale 8-bit image pixel in the [0, 255] range. The *RtB* derives the residues of I with respect to $\{m_1, m_2, m_3\}$. Let us denote these residues as $I_{m1} = \left|I\right|_{2^n-1}$, $I_{m2} = \left|I\right|_{2^n}$ and $I_{m3} = \left|I\right|_{2^n+1}$ respectively. Since n=5, I_{m1} can be easily derived by driving the two 5-bit vectors I_4...I_0 and $00I_7I_6I_5$ to a modulo 2^n-1 adder [19]. Obviously, I_{m2} is equal to I_4...I_0. Since the diminished-one representation is adopted for m_3, I_{m3} is formed by the addition of the two 5-bit vectors I_4...I_0 and $00\bar{I}_7\bar{I}_6\bar{I}_5$ along with a correction vector equal to 11000 [22].

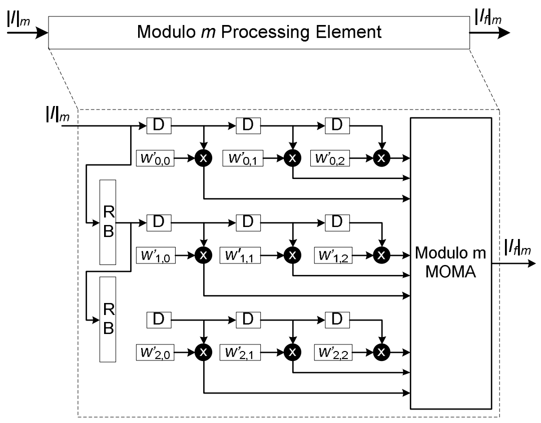

Fig. 4. Architecture of each channel pixel processor

The last two vectors can be merged, hence I_{m3} is the output of an enhanced diminished-one modulo adder driven by the two vectors I_4...I_0 and $11\bar{I}_7\bar{I}_6\bar{I}_5$ [22]. The adder derives the 5-bit diminished-one representation of I_{m3}, denoted as I_{m3d}, along with a zero indication bit (I_{m3dz}) for the case when I_{m3} is equal to 0. In order to be synchronized for the filtering operation, the outputs of the *RtB* are cached using row buffers for every modulo channel (see Fig. 3).

B. *RNS-Based Smoothing Filter*

The first computational stage to be implemented is a smoothing filter that reduces the presence of white noise in the image. To this end, a two-dimensional Gaussian filter is chosen with mean value equal to zero and standard deviation, σ = 0.8. The 3×3 Gaussian kernel, G, used is shown below:

$$G = \begin{bmatrix} g_{0,0} & g_{0,1} & g_{0,2} \\ g_{1,0} & g_{1,1} & g_{1,2} \\ g_{2,0} & g_{2,1} & g_{2,2} \end{bmatrix} = \frac{1}{14}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 2 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad (5)$$

The RNS filter utilizes the integer coefficients while the averaging (1/14) is performed on the saturation block (see Fig. 3). We consider each modulo channel separately. In the modulo 2^n-1 channel, the coefficients $g'_{i,j}$ are the same as in (5). Evidently, when $g'_{i,j}$=1 no multiplication is required, while when $g'_{i,j}$=2 the multiplication can be hardwired since for a modulo 2^n-1 operand A=a_{n-1}...a_0, it holds that $\left|A\times 2\right|_{2^n-1} = a_{n-2}...a_0a_{n-1}$ [25] and thus $\left|I_{m1}\times 2\right|_{2^n-1} = I_{m13}...I_{m10}I_{m14}$.

Hence, 9 product vectors are formed and the filter output, F_{m1}, is implemented by a 9-input modulo 2^n-1 adder (MOMA) composed of End-Around-Carry (EAC) Carry Save Adders (CSAs) and a final two-input modulo 2^n-1 adder. Similarly, in the modulo 2^n channel, $\left|A\times 2\right|_{2^n} = a_{n-2}...a_0 0$ and thus $\left|I_{m2}\times 2\right|_{2^n-1} = I_{m23}...I_{m20}0$. The 9 product vectors that are derived are summed by a modulo 2^n MOMA to produce the smoothed filtered output, F_{m2}, in this modulo channel. Several simplifications are possible due to constant zero bits in the product vectors.

In the modulo 2^n+1 channel, the input pixel values, the coefficients as well as the final result must be in their diminished-one representation. Let $g_{d}'_{i,j}$ denote the diminished-one representation of $g'_{i,j}$. Since all the coefficient values are non-zero, it holds that $g_{d}'_{i,j}$ = $g'_{i,j}$-1. The constant multipliers are straightforward to derive in this case too. Suppose that A_d=a_{n-1}…a_0 is a diminished-one modulo 2^n+1 operand (ignore the zero indication bit A_{d_z} at this point). According to [25] it holds that $\left|A_d \times 2\right|_{2^n} = a_{n-2}...a_0\overline{a}_{n-1}$. Hence, when $g_d{}'_{i,j}$=0, its product with I_{m3} is equal to $\left|P_d\right|_{2^n+1} = \left|I_{m3d}\right|_{2^n+1}$, while when $g_d{}'_{i,j}$=1 the product is equal to $\left|P_d\right|_{2^n+1} = I_{m3d\,3}...I_{m3d\,0}\overline{I}_{m3d\,4}$ the value of which is arithmetically equal to $\left|2I_{m3d}+1\right|_{2^n+1}$. These vectors are justified by the fact that in the diminished-one representation the product of the constant multiplication is written as:

$$\begin{aligned}\left|P_{d_{i,j}}\right|_{2^n+1} &= \left|P_{i,j}-1\right|_{2^n+1} = \left|g'_{i,j}I_{m3}-1\right|_{2^n+1} = \left|(g'_{d\,i,j}+1)(I_{m3d}+1)-1\right|_{2^n+1} \\ &= \left|g'_{d\,i,j}I_{m3d}+g'_{d\,i,j}+I_{m3d}\right|_{2^n+1} = \begin{cases}\left|I_{m3d}\right|_{2^n+1} & if\ g'_{d\,i,j}=0 \\ \left|2I_{m3d}+1\right|_{2^n+1} & if\ g'_{d\,i,j}=1\end{cases}\end{aligned} \tag{6}$$

In the case that I_{m3} is equal to 0, the zero indication bit of the multiplication's result must be set to 1 while the rest of the bits must be set to zero. The only modification needed for this purpose is a logic AND operation (denoted with ∧) between $\overline{I}_{m3d\,z}$ and the bits of I_{m3d} . The result of the multiplier is finally expressed as:

$$\left|P_{d_{i,j}}\right|_{2^n+1} = \begin{cases}(I_{m3d\,z}, I_{m3d\,4}...I_{m3d\,0}) & , if\ g'_{d\,i,j}=0 \\ (I_{m3d\,z}, (\overline{I}_{m3d\,z}\wedge I_{m3d\,3})...(\overline{I}_{m3d\,z}\wedge I_{m3d\,0})(\overline{I}_{m3d\,z}\wedge \overline{I}_{m3d\,4})), & if\ g'_{d\,i,j}=1\end{cases}$$

The result of the smoothing filter in the modulo 2^n+1 channel is acquired, by adding all 9 (diminished-1) product vectors, that is:

$$\begin{aligned}F_{m3d} &= \sum_{i=0}^{d-1}\sum_{j=0}^{d-1}P(i,j)-1 = \left(\sum_{i=0}^{d-1}\sum_{j=0}^{d-1}(P_d(i,j)+1)\right)-1 \\ &= \left(\sum_{i=0}^{d-1}\sum_{j=0}^{d-1}P_d(i,j)\right)+8\end{aligned} \tag{7}$$

This is achieved by a Dadda adder tree consisting of 7 n-bit wide inverted EAC CSAs and a final diminished-one adder. The inversion of the EAC is justified by the fact that the most significant bit of the carry vector output of a CSA, suppose q, has a weight equal to 2^n and since $\left|2^n q\right|_{2^n+1} = \left|-q\right|_{2^n+1} = \left|\overline{q}-1\right|_{2^n+1}$ it should be inverted and placed to the least significant bit position of the carry vector output. This means that an inverted EAC CSA actually increases the result by 1 and a correction equal to -1 must be taken into account so as to acquire the correct sum. Equation (7) also dictates that apart from the 9 product vectors, a correction term is also required to be added that is equal to +8. However, if $\sum_{i=0}^{d-1}\sum_{j=0}^{d-1}I_{m3d\,z}(i,j)=0$, that is, if none of the 9 input pixels has a zero weighted value, then this correction term is introduced by the use of the inverted EAC CSAs and the final diminished-one adder. In the opposite case, a correction term must be explicitly added and its value is equal to:

$$CT_f = \left(-\sum_{i=0}^{d-1}\sum_{j=0}^{d-1}I_{m3d\,z}(i,j)\right)-1 = \left(\sum_{i=0}^{d-1}\sum_{j=0}^{d-1}\overline{I}_{m3d\,z}(i,j)\right)+1$$

since the zero valued products should not be accounted and should be subtracted. Thus, an n-bit 4→2 multiplexer is required in order to choose between the 4 addends, the two produced by the inverted EAC CSA prior to the addition of CT_f and the two after its addition. The select signal of the multiplexer is just a logic OR operation on all 9 zero indication bits. The zero indication bit of the result, $F_{m3d\,z}$, is set in two cases; (a) if the result of the final adder is equal to 0 and (b) if all $I_{m3d\,z}(i,j)$ =1. Thus a two input OR gate can provide $F_{m3d\,z}$ with the one input driven by the most significant bit of the result of the diminished-one adder (case (a)) and the other one by the output of an AND gate that accepts as inputs all 9 $I_{m3d\,z}(i,j)$ (case (b)).

The 3 filters' outputs F_{m1}, F_{m2} and ($F_{m3d\,z}$, F_{m3d}) are then driven to row buffers in order to be synchronized for the next streaming process, which is the edge detection.

C. *RNS-Based Sobel Edge Detector*

In order to detect the edges of the filtered image, (4) must be implemented twice, once with $W_{i,j}$=S_v and once with $W_{i,j}$=S_h and since the one implementation is totally independent of the other they can take place in parallel, inside each modulo m channel. After $\left|S_v\right|_m$ and $\left|S_h\right|_m$ are calculated they are added with a fast two-input modulo m adder in order to form $\left|M\right|_m$.

Again, multiplications are easily performed, since all coefficients are positive or negative powers of 2. The multiplications that correspond to the positive coefficients of the vertical (horizontal) gradient, are implemented exactly as the ones of the previous section for all three modulo channels, while the multiplications for the negative coefficients, $s'_{v_{0,2}}$, $s'_{v_{1,2}}$, $s'_{v_{2,2}}$ ($s'_{h_{2,0}}$, $s'_{h_{2,1}}$, $s'_{h_{2,2}}$) can be also easily implemented. Specifically, for a modulo 2^n-1 operand A it holds that $\left|A\times(-2^t)\right|_{2^n-1} = \overline{a}_{n-t-1}...\overline{a}_0\overline{a}_{n-1}...\overline{a}_{n-t}$ [25]. It therefore holds that $\left|F_{m1}\times(-2^0)\right|_{2^n-1} = \overline{F}_{m1_4}...\overline{F}_{m1_0}$ and $\left|F_{m1}\times(-2)\right|_{2^n-1} = \overline{F}_{m1_3}...\overline{F}_{m1_0}\overline{F}_{m1_4}$. The outputs of the vertical and horizontal edge detection filters in modulo m_1, S_{v1} and S_{h1}, are implemented by 6-input modulo 2^n-1 MOMAs.

If A is a modulo 2^n operand, then $\left|A\times(-2^t)\right|_{2^n} = (\overline{a}_{n-t-1}...\overline{a}_0 1...1)+1$, implying that a constant correction +1 must also be accounted for every negation. Thus, for the edge detection case $\left|F_{m2}\times(-2^0)\right|_{2^n-1} = \left(\overline{F}_{m2_4}...\overline{F}_{m2_0}\right)+1$ and $\left|F_{m2}\times(-2)\right|_{2^n-1} = \left(\overline{F}_{m2_3}...\overline{F}_{m2_0}1\right)+1 = \left(\overline{F}_{m2_3}...\overline{F}_{m2_0}0\right)+2$. Hence, 6 product vectors are formed plus a constant correction equal to +4 that will be driven in two 7-input modulo 2^n MOMAs, one for each gradient component, in order to derive S_{v2} and S_{h2}.

For the third modulo channel, the gradient's negative vertical coefficients are expressed in diminished-one. It holds that $s'_{vd_{0,2}}=-2$, $s'_{vd_{1,2}}=-3$, $s'_{vd_{2,2}}=-2$ (the same apply for the gradients' negative horizontal coefficients $s'_{hd_{2,0}}$, $s'_{hd_{1,1}}$, $s'_{hd_{2,2}}$) and (6) takes the form:

$$\left|P_{d_{i,j}}\right|_{2^n+1} = \begin{cases}\left|-F_{m3d}-2\right|_{2^n+1} & if\ s'_{vd\,i,j}=-2 \\ \left|-2F_{m3d}-3\right|_{2^n+1} & if\ s'_{vd\,i,j}=-3\end{cases} \tag{8}$$

Since $\left|-A_d\right|_{2^n+1} = \left|\overline{A}_d+2\right|_{2^n+1}$ [24], (8) is equal to:

$$\left|P_{d_{i,j}}\right|_{2^n+1} = \begin{cases} \left|\overline{F}_{m3d}\right|_{2^n+1} & \text{if } s'_{vd\,i,j} = -2 \\ \left|2\overline{F}_{m3d}+1\right|_{2^n+1} & \text{if } s'_{vd\,i,j} = -3 \end{cases} \tag{9}$$

with $\left|\overline{F}_{m3d}\right|_{2^n+1} = \overline{F}_{m3d_4} \ldots \overline{F}_{m3d_0}$ and

$\left|2\overline{F}_{m3d}+1\right|_{2^n+1} = \overline{F}_{m3d_3} \ldots \overline{F}_{m3d_0} F_{m3d_4}$ [25]

The zero indication bits must also be incorporated in (9). Similarly to section III.B, some logic AND operations are required between $\overline{F}_{m3d_z}$ and the bits of F_{m3d}. The final result of each one of the 3 negative vertical multipliers is expressed as:

$$\left|P_{d_{i,j}}\right|_{2^n+1} = \begin{cases} \left(F_{m3d_z}, \overline{F}_{m3d_4} \ldots \overline{F}_{m3d_0}\right) & \text{, if } g'_{d\,i,j} = -2 \\ \left(F_{m3d_z}, (\overline{F}_{m3d_z} \wedge \overline{F}_{m3d3}) \ldots (\overline{F}_{m3d_z} \wedge \overline{F}_{m3d0})(\overline{F}_{m3d_z} \wedge F_{m3d_4})\right) & \text{, if } g'_{d\,i,j} = -3 \end{cases}$$

It is evident that the same apply for the corresponding horizontal multipliers too. S_{v3} (and equivalently S_{h3}) is acquired by employing an adder tree for adding the 6 product vectors, an extra CSA level for adding the required correction term that in this case is equal to $CT_s = \left(-\sum_{i=0}^{d-1}\sum_{j=0}^{d-1} F_{m3d_z}\right) - 1 = \left(\sum_{i=0}^{d-1}\sum_{j=0}^{d-1} \overline{F}_{m3d_z}\right) + 1$, an n-bit wide 4→2 multiplexer to choose between four outputs of the adder tree; the two prior to the addition of CT_s and the two after its addition and finally a fast two-input diminished-one modulo 2^n+1 adder. The zero indication bit of the result, that is S_{v3d_z} (S_{h3d_z}) is acquired by an 2-input OR gate with the one input connected to the most significant output of the diminished-one adder and the other one to the output of an AND gate that accepts as inputs the zero indication bits $F_{m3d_z}(i,j)$ of the respective filtered image's pixels.

Finally, the overall vertical and horizontal gradient magnitudes for each modulo value are merged by the respective modulo m adders as: $M_{m1} = \left|S_{v1}+S_{h1}\right|_{2^n-1}$, $M_{m2} = \left|S_{v2}+S_{h2}\right|_{2^n}$ $M_{m3d} = \left|\left(S_{v3d_z}, S_{v3d}\right) + \left(S_{h3d_z}, S_{h3d}\right)\right|_{2^n+1}$. These vectors are the inputs of the Residue-to-Binary Converter.

D. *The Residue-to-Binary Converter (RtB)*

The final stage is the one that converts the result from its residue representation to its binary equivalent, in order to be used by successive non-RNS components. This conversion process is usually based on the Chinese Remainder Theorem (CRT) or revised versions of it. An efficient *RtB* has been presented for {2^n-1, 2^n, 2^n+1} in [27] that computes the binary equivalent X from its residues $x_1 = |X|_{2^n}$, $x_2 = |X|_{2^n-1}$ and $x_3 = |X|_{2^n+1}$, by utilizing the CRT as:

$$X = x_1 + 2^n\left|-2^n x_1 + (2^{2n-1}+2^{n-1})x_2 + (-2^{2n-1}+2^{n-1})x_3\right|_{2^{2n}-1} \tag{10}$$

X is reconstructed by concatenating the $2n$ bits derived from the modulo 2^{2n}-1 summation of 3 vectors, with the n bits of x_1.

When x_3 is encoded in diminished-one, as in our case, the following 3 $2n$-bit vectors [29] should be driven to a 3-input modulo 2^{2n}-1 adder to derive the $2n$ most significant bits of X:

$\overline{x}_{1_{n-1}} \ldots \overline{x}_{1_0} 1 \ldots 1$

$x_{2_0} x_{2_{n-1}} \ldots x_{2_0} x_{2_{n-1}} \ldots x_{2_1}$

$\overline{x}_{3d_0} \hat{x}_{3d_{n-1}} \ldots \hat{x}_{3d_0} \overline{x}_{3d_{n-1}} \ldots \overline{x}_{3d_1}$

where $x_{3d_{n-1}} \ldots x_{3d_0}$, $x_{2_{n-1}} \ldots x_{2_0}$ and $x_{1_{n-1}} \ldots x_{1_0}$ denote the bits of the n-bit vectors x_{3d}, x_2 and x_1, respectively, and $\hat{x}_{3d_i} = x_{3d_i} \vee x_{3d_z}$, $0 \le i < n$, and $\vee$ denotes a logic OR.

Based on these vectors, the output image from the RNS, I_{out} (see Fig. 3), is acquired by adding the following three $2n$-bit wide vectors $\overline{M}_{m1_4} \ldots \overline{M}_{m1_0} 1 \ldots 1$, $M_{m2_0} M_{m2_4} \ldots M_{m2_0} M_{m2_4} \ldots M_{m2_1}$ and, $\overline{M}_{m3d_0} \hat{M}_{m3_4} \ldots \hat{M}_{m3_0} \overline{M}_{m3d4} \ldots \overline{M}_{m3d_1}$ with an $2n$-bit wide EAC CSA and a final modulo 2^{2n}-1 adder.

E. *Saturation*

After the binary equivalent of the RNS processed image pixel, I_{out}, is acquired, correction and saturation is applied on it. The correction is actually a binary comparison and subtraction. If the output pixel's value $I_{out}(x,y) \ge DR/2$, then it is assigned with the value $I_{ed} = ((DR/2) - I_{out}(x,y))$ from which only the 8 least significant bits are kept (saturation). The other bits are truncated since the maximum representable value is 255. If on the other hand $I_{out}(x,y) < DR/2$, then only truncation is applied.

IV. Evaluation

In order to evaluate the proposed edge detection system we at first developed a bit-accurate model of the RNS-based edge detection system using Mathworks's MATLAB software which also provided the coefficients of the Gaussian kernel. Fifty random images were selected from COREL image database, converted to grayscale and cropped at 256×256 resolution. A Gaussian noise of zero mean value and σ=0.01 were introduced to them afterwards. All stages of the proposed edge detection scheme took place in the RNS domain and the acquired results were validated by direct comparison with the respective functions provided by MATLAB. Indicative results from two such images are shown in Fig. 5 and Fig. 6.

The proposed architecture, assuming a 256×256 image, was then described in HDL. The correct operation of the HDL descriptions was validated via simulation. The description was then synthesized in a standard cell 32nm CMOS technology with typical operating conditions (V_{DD}=1.05 *V*, T=25^o *C*) [32] and estimates for area, delay and power dissipation were derived. The delay of the synthesized circuit was less than 2 *ns* (1.948 *ns*) leading to high operating frequencies. The total area required was equal to 112332 μm^2. Most of the implementation area, namely 96.6%, is occupied by the sequential logic (the synchronization row buffers). The RNS-based implementation of the smoothing filter and the Sobel edge detector occupy the 0.9% and 1.8% of the total area, respectively, while both the *BtR* and the *RtB* conversions account for only the 0.5% of the circuit's area. Finally, the saturation block is 0.2% of the total area. Regarding, the power dissipation, the dynamic power dissipation, assuming an operating frequency of 250 *MHz*, was estimated at 30.23 *mW* while the leakage power was equal to 18.57 *mW*.

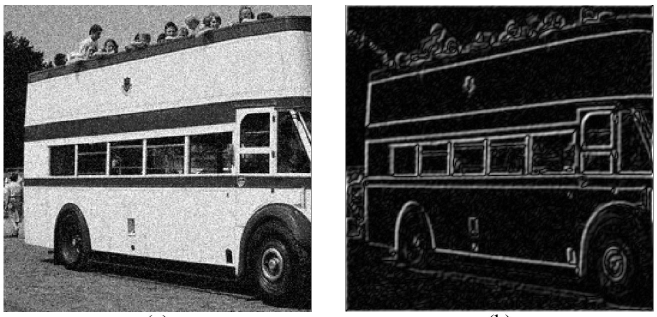
(a) (b)

Fig. 5. a) Original noisy image, b) After smoothing and edge detection

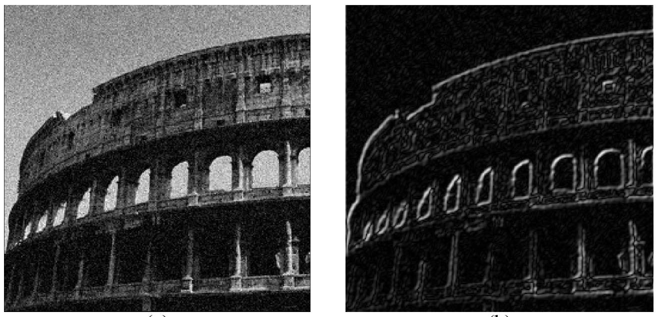
(a) (b)

Fig. 6. a) Original noisy image, b) After smoothing and edge detection

V. CONCLUSIONS

A novel hardware architecture for image smoothing and Sobel edge detection has been proposed in this work. All processing is performed over the 3-moduli set RNS for attaining high operation frequency while row buffers have been utilized to cache and synchronize the input image pixels. Correct operation of the proposed architecture was validated both in software with a bit-accurate model in MATLAB and in hardware by synthesizing it in CMOS 32nm technology.

ACKNOWLEDGMENT

This work was supported by the Caratheodory Programme of the University of Patras (D.178).

REFERENCES

[1] D. Marr, and E. Hildreth, "Theory of edge detection," Proc. R. Soc., London, vol. B207, pp. 187–217, Feb. 1980.

[2] A.R. Khalid, R. Paily, "FPGA implementation of high speed and low power architectures for image segmentation using Sobel Operators," J. Circuit Syst. Comp., vol. 21, pp. 1250050 (1–14), Nov. 2012.

[3] A. Paul, and K. Bharanitharan, "Adaptive edge detection for motion estimation in H.264/AVC," *In Proc. of IEEE Int. Conf. Imaging Syst. Tech.*, pp. 144–147, Jul. 2010.

[4] Y. Li, et al. "Fast SIFT algorithm based on Sobel edge detector," *Proc.of 2nd Int. Conf. Consum. Electron, Commun. Netw.*, pp. 1820–1823, Apr. 2012.

[5] M. Mukherjee, Y. U. Potdar, and A. U. Potdar, "Object tracking using edge detection," *In Proc.of Int. Conf. Work. Emerg. Trends in Tech.*, pp.686–689, Feb. 2010.

[6] Y. Jun, and L. Xiao-bo, "Boundary detection using mathematical morphology," Patter. Recog. Lett., vol. 16, pp. 1277–1286, Dec. 1995

[7] S.V. Thangam, K.S. Deepak, G.N.H. Rai, and P.P. Mirajkar, "An effective edge detection methodology for medical images based on texture discrimination," *In Proc. 7th Int. Conf. Adv. Patter. Recog.*, pp. 227–231, Feb. 2009.

[8] S. Ye, S. Zheng, and W. Hao "Medical image edge detection method based on adaptive facet model," *In Proc. Int. Conf. Comput. Appl. Sys. Model.*, vol. 3, pp. 574–578, Oct. 2010.

[9] P.P. Acharjya, R. Das, and D. Ghoshal, "A study on edge detection using the gradients," Int. J. Sci. Res. Pu., vol. 2, pp. 1–5, Dec. 2012.

[10] J. Canny, "A computational approach to edge detection," IEEE Trans. Patter. Anal. Mach. Intell., vol. 8, pp. 679–698, 1986.

[11] A. Omondi, and B. Premkumar, Residue Number Systems: Theory and Implementation, 1st ed. London: Imperial College Press, 2007.

[12] A. Safari, and Y. Kong, "Four tap Daubechies filter banks based on RNS," *In Proc. Int. Symp. Commun. Inform. Tech.*, pp. 952–955, Oct. 2012.

[13] S. Yun, and Z. Hu, "Method and dedicated processor for image coding based on Residue Number System," *In Proc. Int. Conf. Mod. Prob. Radio Eng. Telecommun. and Comput. Scie.*, pp. 406–407, Feb. 2012.

[14] A.U. Rahman, M.T. Naseem, I.M. Qureshi, and M.Z. Muzaffar, "Reversible watermarking using Residue Number System," *In Proc. 7th Int. Conf. Inf. Assur. Secur.*, pp. 162–166, Dec. 2011.

[15] W. Wang, M.N.S. Swamy, and M.O. Ahmad, "RNS application for digital image processing," I*n Proc. 4th Int. Work. System-on-Chip Real-Time Appl.*, pp. 77–80, Jul. 2004.

[16] D.K. Taleshmekaeil, and A. Mousavi, "The use of Residue Number System in improving the digital image processing," *In Proc. 10th Int. Conf. Signal Process.*, pp. 775–780, Oct. 2010.

[17] D.K. Taleshmekaeil, H. Mohamamdzadeh, and A. Mousavi, "Using Residue Number System for edge detection in digital images processing," *In Proc. 3rd IEEE Int. Conf. Commun. Softw. Netw.*, pp.249–253, May 2011.

[18] S. Moharrami, and D.K. Taleshmekaeil, "The application of the Residue Number System in digital image processing: Propose a scheme of filtering in spatial domain," Res. J. Appl. Sci., vol. 7, pp. 786–292, Aug. 2012.

[19] L. Kalampoukas, D. Nikolos, C. Efstathiou, H. T. Vergos, and J. Kalamatianos, "High-speed parallel-prefix modulo 2^n-1 adders," IEEE Trans. Comput., vol. 49, pp. 673–680, Jul. 2000.

[20] H.T. Vergos, C. Efstathiou, and D. Nikolos, "Diminished-one modulo 2^n+1 adder design," IEEE Trans. Comput., vol. 51, pp. 1389–1399, Dec. 2002.

[21] C. Efstathiou, H.T. Vergos, and D. Nikolos, "Handling zero in diminished-one modulo 2^n+1 adders," Int. J. Electron., pp. 133–144, Feb. 2003.

[22] H.T. Vergos, and D. Bakalis, "On implementing efficient modulo 2^n+1 arithmetic components," J. Circuits Syst. Comput., vol. 19, pp. 911–930, Aug. 2010.

[23] D. Bakalis, H. T. Vergos, and A. Spyrou, "Efficient modulo 2^n±1 squarers," Integration, vol. 44, pp. 163–174, Jun. 2011.

[24] E. Vassalos, D. Bakalis, and H.T. Vergos, "On the design of modulo 2^n±1 subtractors and adders/subtractors," Circuits Syst. Signal Process., vol. 30, pp. 1445–1461, Dec. 2011.

[25] E. Vassalos, and D. Bakalis, "CSD-RNS-based single constant multipliers," J. Sign. Process. Syst., vol.67, pp. 255–268, Jun. 2012.

[26] H.T. Vergos, and G. Dimitrakopoulos, "On modulo 2^n+1 adder design," IEEE Trans. Comput., vol. 61, pp. 173–186, Feb. 2012.

[27] Z. Wang, G.A. Jullien, and W.C. Miller, "An improved residue-to-binary converter," IEEE Trans. Circuits Syst. I, vol. 47, pp. 1437–1440, Sep. 2000.

[28] K. Navi, A. Molahosseini, and M. Esmaeildoust, "How to teach Residue Number System to computer scientists and engineers", IEEE Trans. Educ., vol. 54, pp. 156–163, Feb. 2011.

[29] E. Vassalos, D. Bakalis, and H. T. Vergos, "Reverse converters for RNSs with diminished-one encoded channels," *In Proc. of IEEE EUROCON Conference*, Jul. 2013.

[30] L. Leibowitz, "A Simplified binary arithmetic for the Fermat number transform," IEEE Trans. Acoustics Speech Signal Process., vol. 24, pp. 356–359, Oct. 1976.

[31] D.G. Bailey, Design for embedded image processing on FPGAs, 1st ed., Asia: John Wiley & Sons, 2011.

[32] Synopsys Inc., "SAED 32/28nm Digital Standard Cell Library", 2012.