International Journal of Advances in Applied Science and Engineering (IJAEAS)
ISSN (P): 2348-1811; ISSN (E): 2348-182X
Vol. 1, Issue 4, Dec 2014, 198-206

VLSI IMPLEMENTATION OF 2-D FDWT IMAGE SCALING PROCESSOR

RAJESH BALUGURI [1], P.SUDHAKAR RAO [2]

[1]PG Student, Electronics & Comm. Engineering, NOVA College of Engg & Tech., Vijayawada, A.P, India

[2]HOD & Associate Professor, Electronics & Comm. Engineering, NOVA College of Engg & Tech., Vijayawada, A.P, India

Email:-rajesh06501a0432@gmail.com[1], sudhavlsi718@gmail.com [2]

ABSTRACT: ***The block diagram of the proposed scaling algorithm. It consists of a sharpening spatial filter, a clamp filter, and a bilinear interpolation. The sharpening spatial and clamp filters serve as pre filters to reduce blurring and aliasing artifacts produced by the bilinear interpolation. First, the input pixels of the original images are filtered by the sharpening spatial filter to enhance the edges and remove associated noise. Second, the filtered pixels are filtered again by the clamp filter to smooth unwanted discontinuous edges of the boundary regions. Finally, the pixels filtered by both of the sharpening spatial and clamp filters are passed to the bilinear interpolation for up-/ downscaling.***

KEYWORDS—***VLSI,FDWT,Image,Scaling,Processor***

I. INTRODUCTION

The The proposed scaling algorithm consists of two combined prefilters and one simplified bilinear interpolator. For VLSI implementation, the bilinear interpolator can directly obtain two input pixels P(m,n) and P(m,n+1) from two combined prefilters without any additional line-buffer memory. Fig. 3 shows the block diagram of the VLSI architecture for the proposed design. It consists of four main blocks: a register bank, a combined filter, a bilinear interpolator, and a controller. The details of each part will be described in the following sections.

A. Register Bank

In this brief, the combined filter is filtering to produce the target pixels of P(m,n) and P (m,n+1) by using ten source pixels. The register bank is designed with a one-line memory buffer, which is used to provide the ten values for the immediate usage register bank with a structure of ten shift registers. When the shifting control signal is produced from the controller, a new value of P(m+3,n) will be read into Reg41, and each value stored in other registers belonging to row n + 1 will be shifted right into the next register or line-buffer memory. The Reg40 reads a new value of P(m+2,n) from the line-buffer memory, and each value in other registers belonging to row n will be shifted right into the next register

B. Combined Filter

The combined T-model or inversed T-model convolution function of the sharpening spatial and clamp filters had been discussed in Section II, and the equation is represented in (1). Fig. 5 shows the

six-stage pipelined architecture of the combined filter and bilinear interpolator, which shortens the delay path to improve the performance by pipeline technology. The stages 1 and 2 in Fig. 5 show the computational scheduling of a T-model combined and an inversed T-model filter. The T-model or inversed T-model filter consists of three reconfigurable calculation units (RCUs), one multiplier–adder (MA), three adders (+), three subtracters (−), and three shifters (S). The hardware architecture of the T-model combined filter can be directly mapped with the convolution equation shown in (1). The values of the ten source pixels can be obtained from the register bank mentioned earlier. The symmetrical circuit, as shown in stages 1 and 2 of Fig. 5, is the inversed T-model combined filter designed for producing the filtered result of p(m,n+1). Obviously, The T-model and the inversed T-model are used to obtain the values of p (m,n) and p(m,n + 1) simultaneously. The architecture of this symmetrical circuit is a similar symmetrical structure of the T-model combined filter, as shown in stages 1 and 2 of Fig. 5. Both of the combined filter and symmetrical circuit consist of one MA and three RCUs. The MA can be implemented by a multiplier and an adder. The RCU is designed for producing the calculation functions of (S-C) and (S-C-1) times of the source pixel value, which must be implemented with C and S parameters. The C and S parameters can be set by users according to the characteristics of the images. The architecture of the proposed low-cost combined filter can filter the whole image with only a one-line-buffer memory, which successfully decreases the memory requirement from four to one line buffer of the combined filter in our previous work [15]. Table I lists the parameters and computing resource for the RCU. With the selected C and S values listed in Table I, the gain of the clamp or sharp convolution function is {8, 16, 32} or {4, 8, 16}, which can be eliminated by a shifter rather than a divider. Fig. shows the architecture of the RCU. It consists of four shifters, three multiplexers (MUX), three adders, and one sign circuit. By this RCU design, the hardware cost of the combined filters can be efficiently reduced.

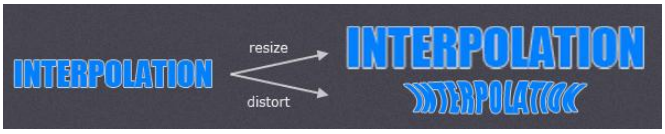
Bilinear Interpolator and Controller

In the previous discussion, the bilinear interpolation is simplified as shown in (6). The stages 3, 4, 5, and 6 in Fig. show the four-stage pipelined architecture, and two-stage pipelined multipliers are used to shorten the delay path of the bilinear interpolator. The input values of P(m,n) and P(m,n+1) are obtained from the combined filter and symmetrical circuit. By the hardware sharing technique, as shown in (6), the temperature result of the function "P (m,n) + dy × (P (m,n+1) − P (m,n))" can be replaced by the previous result of "P (m+1,n) + dy × (P(m+1,n+1) − P (m+1i,n))." It also means that one multiplier and two adders can be successfully reduced by adding only one register. The controller is implemented by a finite-

statemachine circuit. It produces control signals to control the timing and pipeline stages of the register bank, combined filter, and bilinear interpolator.

IMAGE INTERPOLATION

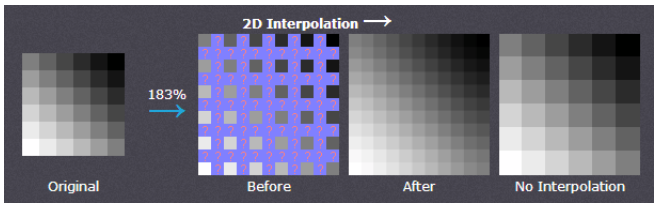
Image interpolation occurs in all digital photos at some stage — whether this be in bayer demosaicing or in photo enlargement. It happens anytime you resize or remap (distort) your image from one pixel grid to another. Image resizing is necessary when you need to increase or decrease the total number of pixels, whereas remapping can occur under a wider variety of scenarios: correcting for lens distortion, changing perspective, and rotating an image.



Even if the same image resize or remap is performed, the results can vary significantly depending on the interpolation algorithm. Itis only an approximation, therefore an image will always lose some quality each time interpolation is performed. This tutorial aims to provide a better understanding of how the results may vary — helping you to minimize any interpolation-induced losses in image quality.

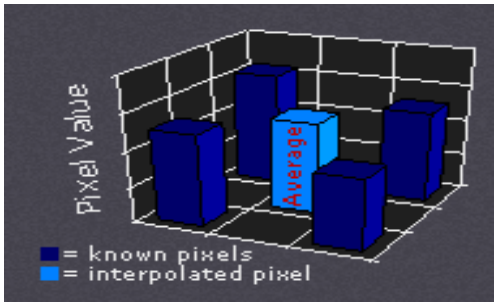
IMAGE RESIZE

mage interpolation works in two directions, and tries to achieve a best approximation of a pixel's color and intensity based on the values at surrounding pixels. The following example illustrates how resizing / enlargement works:



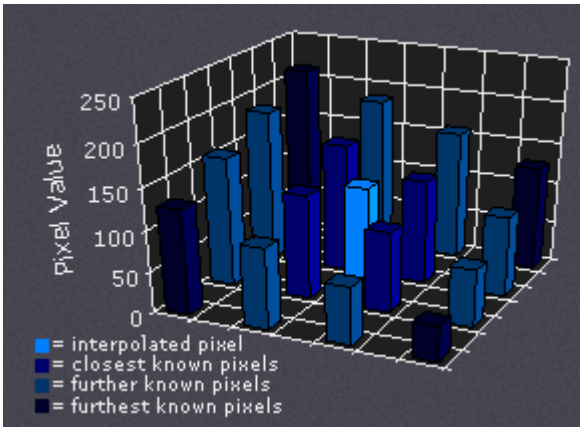
Unlike air temperature fluctuations and the ideal gradient above, pixel values can change far more abruptly from one location to the next. As with the temperature example, the more you know about the surrounding pixels, the better the interpolation will become. Therefore results quickly deteriorate the more you stretch an image, and interpolation can never add detail to your image which is not already present. Nearest neighbor is the most basic and requires the least processing time of all the interpolation algorithms because it only considers one pixel — the closest one to the interpolated point. This has the effect of simply making each pixel bigger. Bilinear interpolation considers the closest 2x2 neighborhood of known pixel values surrounding the unknown pixel. It then takes a weighted average of these 4 pixels to arrive at its final interpolated value. This results in much smoother looking images than nearest neighbor.

The diagram to the left is for a case when all known pixel distances are equal, so the interpolated value is simply their sum divided by four.



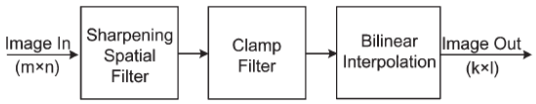
BICUBIC INTERPOLATION

Bicubic goes one step beyond bilinear by considering the closest 4x4 neighborhood of known pixels — for a total of 16 pixels. Since these are at various distances from the unknown pixel, closer pixels are given a higher weighting in the calculation. Bicubic produces noticeably sharper images than the previous two methods, and is perhaps the ideal combination of processing time and output quality. For this reason it is a standard in many image editing programs (including Adobe Photoshop), printer drivers and in-camera interpolation.



II. SCALING ALGORITHM

Initial To conserve computing resource and memory buffer, these two filters are simplified and combined into a combined filter. The details of each part will be described in the following sections



Low-Complexity Sharpening Spatial and

Clamp Filters

The sharpening spatial filter, a kind of high-pass filter, is used to reduce blurring artifacts and defined by a kernel to increase the intensity of a center pixel relative to its neighboring pixels. The clamp filter [6], a kind of low-pass filter, is a 2-D Gaussian spatial domain filter and composed of a convolution kernel array. It usually contains a single positive value at the center and is completely surrounded by ones [15]. The clamp filter is used to reduce aliasing artifacts and smooth the unwanted discontinuous edges of the boundary regions.

The sharpening spatial and clamp filters can be represented by convolution kernels. A larger size of convolution kernel will produce higher quality of images. However, a larger size of convolution filter will also demand more memory and hardware cost. For example, a 6 × 6 convolution filter demands at least a five-line-buffer memory and 36 arithmetic units, which is much more than the two-line-buffer memory and nine arithmetic units of a 3 × 3 convolution filter. In our previous work [15], each of the sharpening spatial and clamp filters was realized by a 2-D 3 ×

3 convolution kernel as shown in Fig. 2(a). It demands at least a four-line-buffer memory for two 3 × 3 convolution filters. For example, if the image width is 1920 pixels, 4 × 1920 × 8 bits of data should be buffered in memory as input for processing.

To reduce the complexity of the 3 × 3 convolution kernel, a cross-model formed is used to replace the 3 × 3 convolution kernel, as shown in Fig. 2(b). It successfully cuts down on four of nine parameters in the 3 × 3 convolution kernel. Furthermore, to decrease more complexity and memory requirement of the cross-model convolution kernel, T-model and inversed T-model convolution kernels are proposed for realizing the sharpening spatial and clamp filters. As shown in Fig. 2(c), the T-model convolution kernel is composed of the lower four parameters of the cross-model, and the inversed T-model convolution kernel is composed of the upper four parameters. In the proposed scaling algorithm, both the T-model and inversed T-model filters are used to improve the quality of the images simultaneously.

The T-model or inversed T-model filter is simplified from the 3 × 3 convolution filter of the previous work [15], which not only efficiently reduces the complexity of the convolution filter but also greatly decreases the memory requirement from two to one line buffer for each convolution filter. The T-model and the inversed T-model provide the low-complexity and lowmemory-requirement convolution kernels for the sharpening spatial and clamp filters to integrate the VLSI chip of the proposed low-cost image scaling processor.

Combined Filter

In proposed scaling algorithm, the input image is filtered by a sharpening spatial filter and then filtered by a clamp spatial filter again. Although the sharpening spatial and clamp filters are simplified by T-models and inversed T-models, it still needs two line buffers to store input data or intermediate values for each T-model or inversed T-model filter. Thus, to be able to reduce more computing resource and memory requirement, sharpening spatial and clamp filters, which are formed by the T-model or inversed T-model, should be combined together into a combined filter as

$$P'_{(m,n)} = \left[P^*_{(m,n)} \begin{bmatrix} -1 & S & -1 \\ & -1 & \end{bmatrix} /(S-3)\right]^* \begin{bmatrix} 1 & C & 1 \\ & 1 & \end{bmatrix} /(C+3)$$

$$= P^*_{(m,n)} \begin{bmatrix} -1 & S-C & SC-2 & S-C & -1 \\ & -2 & S-C & -2 & \\ & & -1 & & \end{bmatrix} / [(S-3)\times(C+3)] \quad (1)$$

$$\approx P^*_{(m,n)} \begin{bmatrix} -1 & S-C & SC-2 & S-C & -1 \\ & -2 & S-C & -2 & \end{bmatrix} / [(S-3)\times(C+3)] \quad (2)$$

where S and C are the sharp and clamp parameters and P(m,n) is the filtered result of the target pixel P(m,n) by the combined filter. A T-model sharpening spatial filter and a T-model clamp filter have been replaced by a combined T-model filter as shown in (1). To reduce the one-line-buffer memory, the only parameter in the third line, parameter −1 of P(m,n−2), is removed, and the weight of parameter −1 is added into the

parameter S-C of P(m,n−1) by S-C-1 as shown in (2). The combined inversed T-model filter can be produced in the same way. In the new architecture of the combined filter, the two T-model or inversed T-model filters are combined into one combined T-model or inversed T-model filter. By this filter-combination technique, the demand of memory can be efficiently decreased from two to one line buffer, which greatly reduces memory access requirements for software systems or hardware memory costs for VLSI implementation.

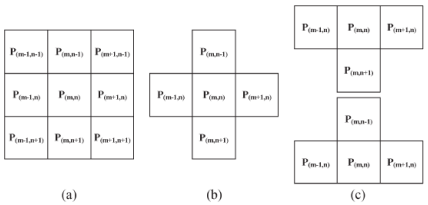


Fig. 2. Weights of the convolution kernels. (a) 3 × 3 convolution kernel. (b) Cross-model convolution kernel. (c) T-model and inversed T-model convolution kernels.

Simplified Bilinear Interpolation

In the proposed scaling algorithm, the bilinear interpolation method is selected because of its characteristics with low complexity and high quality. The bilinear interpolation is an operation that performs a linear interpolation first in one direction and, then again, in the other direction. The output pixel P(k,l) can be calculated by the operations of the linear interpolation in both x- and y-directions with the four nearest neighbor pixels. The target pixel P(k,l) can be calculated by

$$P_{(k,l)} = (1-dx)\times(1-dy)\times P_{(m,n)} + dx\times(1-dy)\times P_{(m+1,n)} + (1-dx)\times dy\times P_{(m,n+1)} + dx\times dy\times P_{(m+1,n+1)} \quad (3)$$

where P(m,n), P(m+1,n), P(m,n+1), and P(m+1,n+1) are the four nearest neighbor pixels of the original image and the dx and dy are scale parameters in the horizontal and vertical directions.

By (2), we can easily find that the computing resources of the bilinear interpolation cost eight multiply, four subtract, and three addition operations. It costs a considerable chip area to implement a bilinear interpolator with eight multipliers and seven adders. Thus, an algebraic manipulation skill has been used to reduce the computing resources of the bilinear interpolation. The original equation of bilinear interpolation is presented in (2), and the simplifying procedures of bilinear interpolation can be described from (4)–(6). Since the function of dy × (P(m,n+1) − P(m,n)) + P(m,n) appears twice in (6), one of the two calculations for this algebraic function can be reduced

$$P_{(k,l)} = \left[(1-dy)\times P_{(m+1,n)} + dy\times P_{(m+1,n+1)}\right]\times dx + \left[(1-dy)\times P_{(m,n)} + dy\times P_{(m,n+1)}\right](1-dx) \quad (4)$$

$$= \left[P_{(m+1,n)} + dy\times(P_{(m+1,n+1)} - P_{(m+1,n)})\right]\times dx + \left[P_{(m,n)} + dy\times(P_{(m,n+1)} - P_{(m,n)})\right](1-dx) \quad (5)$$

$$= \left\{\left[P_{(m+1,n)} + dy\times(P_{(m+1,n+1)} - P_{(m+1,n)})\right] - \left[P_{(m,n)} + dy\times(P_{(m,n+1)} - P_{(m,n)})\right]\right\}\times dx + \left[P_{(m,n)} + dy\times(P_{(m,n+1)} - P_{(m,n)})\right]. \quad (6)$$

By the characteristic of the executing direction in bilinear interpolation [15], the values of dy for all pixels that are selected on the vertical axis of n row equal to n + 1 row, and only the values of dx must be changed with the position of x. The result of the function "[P(m,n) + dy × (P(m,n+1) − P(m,n))]" can be replaced by the previous result of "[P(m+1,n) + dy × (P(m+1,n+1) − P(m+1i,n))]" as shown in (6). The simplifying

procedures successfully reduce the computing resource from eight multiply, four subtract, and three add operations to two multiply, two subtract, and two add operations.

III. WAVELET DECOMPOSITION AND THRESHOLDING

Wavelet is a mathematical function used to divide a given function or continuous-time signal into different scale components. One can assign a frequency range to each scale component. Each scale component can then be studied with a resolution that matches its scale. Thus the Wavelet is a multi resolution representation function. Wavelet transform is the discrete sampling of the wavelets. Based on the recurrence relations property of wavelet, the most common wavelet transforms, such as Daubechies wavelet transform, generate progressively finer discrete samplings of an implicit mother wavelet function; each resolution is twice that of the previous scale down-sampled by 2. Therefore, using the one level wavelet transform, the input signal can be decomposed into two frequency coefficients, the approximation coefficients as the low frequency part and the detail coefficients as the high frequency part. This is the so called wavelet decomposition. With higher level decompositions, multi resolution representation of the signal can be achieved. Figure 1.3 shows the wavelet decomposition of an image. The left picture is the original image and the right one using 1-level wavelet decomposition. We can see from the right image that the top left small picture is the low frequency part which keeps the energy mostly while the others are detail information.

Figure 1.3 Right one shows the wavelet decomposition of the left picture. It uses 1 level Db4 wavelet decomposition in the Matlab. The image size is 512x512

Wavelet thresholding is a denoising method that applies the thresholding shrinkage upon the high frequency components after the wavelet decomposition. There are two basic thresholding methods, the hard thresholding and the soft thresholding, by which the threshold value is computed. Generally speaking, the soft thresholding is used for the wavelet thresholding application. The most common wavelet thresholding methods are Bayes Shrink [10], Visu Shrink[9] and SURE Shrink [8]. Block-based discrete cosine transform (BDCT) is adopted by widely used image/video compression standards, such JPEG, MPEG, and H-263, due to its high energy compaction and low computational complexity. Before BDCT, the image must be split into 8x8 blocks of pixels. If the data does not represent an integer number of blocks then the encoder must fill the remaining area of

the incomplete blocks with some form of dummy data. Filling the edge pixels with a fixed color 8 blocks.×(typically black) creates ringing artifacts along the visible part of the border; repeating the edge pixels is a common technique that reduces the visible border, but it can still create artifacts. DCT is applied on such 8x8 blocks.

Figure 1.4 Illustration of blocking artifacts. Left is the original image with size of512 , while the right one is the compressed output of left with bit-rate=0.18. ×512

After the DCT, quantization is used to reduce the amount of information in the high frequency components. This is done by simply dividing each component in the frequency domain by a constant for that component, and then rounding to the nearest integer, which is the main lossy operation in the whole process. As a result of this, it is typically the case that many of the higher frequency components are rounded to zero, and many of the rest become small positive or negative numbers, which take many fewer bits to store. When using quantization with block-based coding, several types of artifacts can appear, including staircase noise along curving edges, "mosquito noise" around edges, and blocking artifacts. The major problem here is the blocking artifacts, the discontinuities along the block boundaries. The blocking artifacts and other compression artifacts become more severe with increasing compression rates. Figure 1.4 shows the illustration of an image with blocking artifacts after over compressed

IV. RESULTS

The approximation and the detailed coefficients of test image were computed first in Matlab platform. We calculated approximation and detailed Coefficients for all the rows of the 256 × 256 input image referred as the 1-D FDWT transform next, the same Matlab routine was implemented on this coefficients column wise to obtain the 2-D FDWT coefficients. By this consecutive row and column wise operation on the input image data, we get the level-1 decomposition coefficients shown in figure5.7.

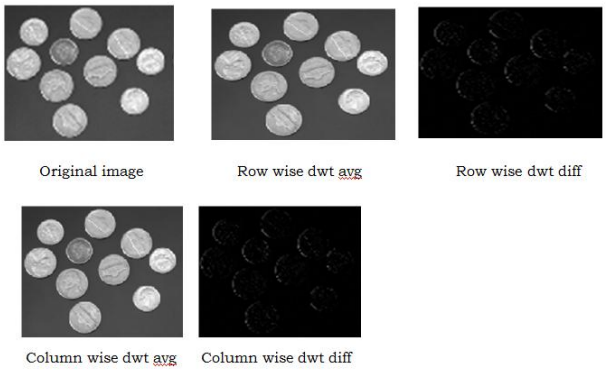


Figure5.7. cell image & its10% reduction using lifting

The RTL schematic for the Forward Piecewise Lifting Scheme generated by the Xilinx Synthesis tool is shown in figure5.8 below.

Figure5.8.RTL Schematic for Forward Lifting Scheme

The RTL schematic for the Forward Piecewise Lifting Scheme generated by the Xilinx Synthesis tool is shown in figure5.9 below.

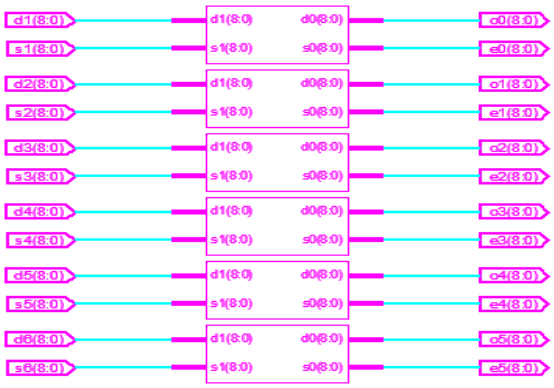


Figure5.9.RTL Schematic for Inverse Lifting Scheme

References

1. *X. Zhang and X.Wu, "Image interpolation by adaptive 2-D autoregressive modeling and soft-decision estimation," IEEE Trans. Image Process., vol. 17, no. 6, pp. 887–896, Jun. 2008.*
2. *F. Cardells-Tormo and J. Arnabat-Benedicto, "Flexible hardware-friendly digital architecture for 2-D separable convolution-based scaling," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 53, no. 7, pp. 522–526, Jul. 2006.*
3. *S. Ridella, S. Rovetta, and R. Zunino, "IAVQ-interval-arithmetic vector quantization for image compression," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 47, no. 12, pp. 1378–1390, Dec. 2000.*
4. *S. Saponara, L. Fanucci, S. Marsi, G. Ramponi, D. Kammler, and E. M. Witte, "Application-specific instruction-set processor for Retinexlink image and video processing," IEEE Trans. Circuits Syst. II, Exp Briefs, vol. 54, no. 7, pp. 596–600, Jul. 2007.*
5. *P. Y. Chen, C. C. Huang, Y. H. Shiau, and Y. T. Chen, "A VLSI implementation of barrel distortion correction for wide-angle camera images," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 56, no. 1, pp. 51–55, Jan. 2009.*
6. *M. Fons, F. Fons, and E. Canto, "Fingerprint image processing acceleration through run-time reconfigurable hardware," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 57, no. 12, pp. 991–995, Dec. 2010.*
7. *C. H. Kim, S. M. Seong, J. A. Lee, and L. S. Kim, "Winscale : An imagescaling algorithm using an area pixel model," IEEE Trans. Circuits Syst.Video Technol., vol. 13, no. 6, pp. 549–553, Jun. 2003.*
8. *C. C. Lin, Z. C. Wu, W. K. Tsai, M. H. Sheu, and H. K. Chiang, "The VLSI design of winscale for digital image scaling," in Proc. IEEE Int. Conf. Intell. Inf. Hiding Multimedia Signal Process., Nov. 2007, pp. 511–514.*

Author's Profile

- **RAJESH BALUGURI** is a PG student persuing his M.Tech in VLSI & ES specialization in NOVA College of Engg & Tech., Vijayawada.
- **P.SUDHAKAR RAO** is working as Associate Professor, Head of the department for ECE department in NOVA College of Engg. & Tech., Vijayawada. He has 10 years of teaching experience and 1 year industrial experience. He has published 04 papers on his research work.