Parallel Algorithm for Hardware Implementation of Inverse Halftoning

Umair F. Siddiqi, Sadiq M. Sait
Department of Computer Engineering
King Fahd University of Petroleum & Minerals
Dhahran, Saudi Arabia
{umair,sadiq}@ccse.kfupm.edu.sa

Aamir A. Farooqui
Synopsys Inc., Synopsys Module Compiler
700 Middlefield Road
Mountain View CA 94034, USA
aamirf@synopsys.com

Abstract— **A Parallel algorithm and its hardware implementation of Inverse Halftone operation is proposed in this paper. The algorithm is based on Lookup Tables from which the inverse halftone value of a pixel is directly determined using a pattern of pixels. A method has been developed that allows accessing more than one value from the lookup table at any time. The lookup table is divided into smaller lookup tables, such that each pattern selected at any time goes to a separate smaller lookup table. The 15-pixel parallel version of the algorithm was tested on sample images and a simple and effective method has been used to overcome quality degradation due to pixel loss in the proposed algorithm. It can provide at least 4 times decrease in lookup table size when compared with serial lookup table method implemented multiple times for same number of pixels.**

I. INTRODUCTION

The process of rendition of continuous-tone pictures on a media on which only two levels can be displayed is defined as halftoning. [1]. The problem has gained importance since the time of printing press, when attempts were made to print images on paper by adjusting the size of the dots according to the local image intensity. This process is termed as analog halftoning. The digital halftoning has also become important with the availability and adoption of bi-level devices such as fax machines and plasma displays [2]. The input to a digital halftoning system is an image whose pixels have more than two levels, (e.g., 256), and the result of the halftoning process is an image that has only two levels. Inverse halftoning, on the other hand, is the reconstruction of a continuous tone image from its halftoned version. Inverse halftoning finds applications in diverse areas of image compression, printed image processing, scaling, enhancement, etc. In these applications, image processing operations cannot be directly performed on the image and inverse halftoning is essential [1]. Inverse halftone operation is generally considered to be an iterative process involving complex iterative computations [3]-[6]. The Look up Table (LUT) based method proposed by Murat Mese and Vaidayanathan is a non iterative and computation free method of inverse halftoning that also yields images of good quality. They also suggested this method to be fastest among all previously known ones [7]. To determine the inverse halftone value at a point, the algorithm looks at the pixel's neighborhood. Depending upon the distribution of pixels in the neighborhood, a contone (gray level) value is assigned from a pre-computed lookup table. This algorithm is suitable for parallel hardware implementation since each pixel's contone value is determined in isolation and is not dependent on other pixels. However, the drawback of this method is that it can only process one-pixel at a time. In this paper (a) the algorithm proposed by Mese and Vaidyanathan in [7] has been modified so that it can fetch more than one value from the lookup table at any time, and, (b) a high throughput FPGA implementation has been developed that shows that the proposed algorithm can be implemented easily in hardware.

The paper is organized as follows: Section 2 describes the new algorithms for parallel access of lookup table. Section 3 presents the experimental results of the proposed algorithm, followed by VLSI implementation results in Section 4. Conclusion is given in Section 5.

II. ALGORITHM FOR PARALLEL TABLE ACCESS

A. *Algorithm for distribution of "19pels"*

As mentioned in the previous section that LUT method looks at the distribution of pixels in the neighborhood of the pixel that is going to be inverse halftoned to determine a gray level value. The authors in [7] suggested a 19-bit template named "19pels", to select pixels from the neighborhood and this "19pels" template is shown in Figure 1.

1	2	3	4	5
6	7	8	9	10
11	12	**0**	13	14
	15	16	17	
		18		

Figure 1: "19pels" template.

This "19pels" is associated with each pixel that is to be inverse halftoned. Thus for an n-pixels parallel system, we have n "19pels" in parallel and the Lookup Table contains inverse halftone values for all "19pels". In the proposed algorithm, the Lookup Table is divided into several smaller lookup tables (SLUTs) such that every "19pels" selected in a cycle will go to a separate SLUT.

1. Pattern(n)= "19pels" with pixel 0 at position (row, col);

2. Pattern(n+1)= "19pels" with pixel 0 at position (row,col+1);

3. inner= XOR(pattern(n), pattern(n+1));

4. Magnitude of RXC= |RXC|= Number of Ones(inner);

5. Sign of RXC= sgn(RXC)= $\begin{cases} +; \mid \text{Pattern}(n+1) \mid > \mid \text{Pattern}(n) \mid \\ -; \mid \text{Pattern}(n+1) \mid < \mid \text{Pattern}(n) \mid \end{cases}$

Figure 2: Illustration of Relative XOR Change (RXC).

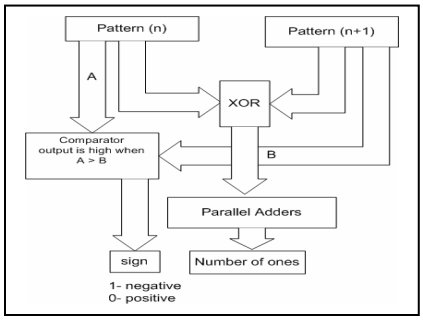

Figure 3: Block diagram for "Relative XOR Change".

The algorithm can be explained using a parameter Relative XOR Change (RXC) as shown in Figure 2 and Figure 3. Pattern(n) and pattern(n+1) are "19pels" of two column-wise consecutive pixels in any row and signal 'inner' holds the bit to bit exclusive-OR of pattern(n) with pattern(n+1). The magnitude of RXC is the number of ones present in the signal 'inner' and sign of RXC is positive when pattern(n+1) is greater than pattern(n) and is negative otherwise. The values of RXC can lie between -19 to +19. RXC is easy to implement in parallel hardware.

To demonstrate how RXC can help develop efficient table distribution algorithm the RXC have been calculated on halftones of gray-levels obtained through (a) Error Diffusion (ED) algorithm proposed by Floyd and Steinberg [8]; (b) Green Noise ED algorithm proposed by D.L. Lau, G.R. Arce and N.C. Gallagher [9] and (c) the Edge Enhancement ED algorithm proposed by R. Eschbach and K. Knoxin [10] (and further modified for adaptive threshold modulation by N. Damera Venkata and Evans [11]). In Figure 4 the RXC has been applied in form of a raster scan through halftones and the results obtained are shown in Table 1. The halftones were generated using MATLAB tool box available at [13].

Our task is to separate all input "19pels" from each other so that they can go to separate SLUTs. It has been observed that RXC can give unique numbers across consecutive "19pels" in the image except for the regions of periodic vibratory or zero response. Therefore, all "19pels" selected in parallel can be represented by unique RXC values and this property of RXC has been used to develop an efficient table distribution algorithm.

Procedure:
input= g_i (Any gray-level between 0 to 255)
1. x1[1:m, 1:n]= $g_i \times$ ones(m, n);
2. x2[1:m, 1:n]= **Halftone Algorithm**(x1);
3. for i= 1: m
4. for j= 1: n
5. y(j)= RXC(x2(i,j), x2(i,j+1));
6. end;
7. output= concatenate[output, y];
8. end;

Results Obtained:

output $\cong \begin{cases} \text{Periodic Vibratory function} & ; g1 < g_i < g2 \\ \text{Zero} & ; g_i < g3 \text{ and } g_i > g4 \\ \text{NOT Periodic Vibratory function} & ; \text{otherwise} \end{cases}$

Figure 4: Procedure and results obtained from RXC calculated over gray-levels 0 to 255.

TABLE 1: PARAMETER VALUES FOR RESULTS IN FIGURE 4.

Halftone Algorithm	g1	g2	g3	g4
Floyd & Steinberg ED	120	140	15	245
Green Noise ED (h=0)	136	141	10	245
Edge Enhancement ED	112 190	146 210	20	245

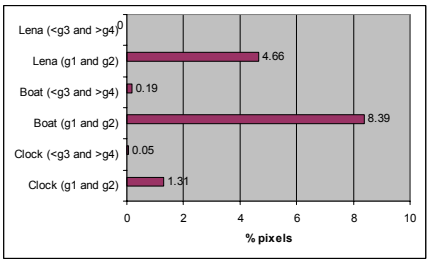

Figure 5: Percentage of pixels present in the regions of periodic vibratory and zero RXC response for Green Noise ED

Looking at the distribution of gray levels in different standard images, it has been observed that the range of gray levels over which RXC have Periodic vibratory or zero response are quite limited in all images, as shown in Figure 5. It has been further observed that only gray levels of aperiodic halftone patterns can be divided into separate regions of RXC, as shown in Table 1. Therefore, all halftones included in this paper are aperiodic and a discussion on aperiodic and periodic halftones can be found in [12].

Figure 6, illustrates the algorithm that can accept 'N' parallel "19pels" patterns and can distribute them among 't' SLUTs, from where each "19pels" can fetch its inverse halftone value independent of others.

B. *Algorithm for SLUT generation*

The previous part has shown the algorithm for input "19pels" that enables them to go to their separate SLUTs. Here we will present the algorithm to develop SLUTs. The algorithm is presented in Figure 7. It can generate SLUTs for any training set comprising of any number of images. A discussion on training sets and their suitable sizes can be found in [7]. In hardware, the SLUTs tables can be stored in terms of decoders that have hard-wired contone values for corresponding "19pels."

Input: Halftone Image **Image[1:m, 1:n];**

OutPut: Smaller Lookup Table Number **y(0)…y(N)**

1. for i=1:1:m
2. P(0)= zeros(1, 19);
3. for j=1:N:n
4. P(1)=Image(i, j), P(2)=Image(i,j+1),
…, P(N)=Image(i, j+N);
5. R1=RXC(P(0),P(1)), R2=RXC(P(1),P(2)),
…, RN=RXC(P(N-1),P(N));
6. P(0)= P(N);
7. **y(0)**= mod(R1, t); **y(1)**=mod(R2, t);
…; **y(N)**=mod(RN, t);
11. end for;
12. end for;

Notes: (a) P(0),P(1),…,P(N) are "19pels" of total N pixels that were taken parallel at a time. (b) t is the total number of SLUTs.

Figure 6: Proposed Algorithm to obtain SLUTS values for "19pels".

Input: All "19pels" present in the training set P[1, ……, n]
Output: Nineteen SLUTs SLUT[1, 2, ……,19]
1. for i= 1:n
2. s1 = RXC(zeros(1,19), P[i]);
3. SLUT[s1] = P[i];
4. end
5. k[1, 2, ….. 19]= number_of_rows(SLUT[1, 2, ……, 19]);
6. for i1=1:19
7. for i2=1:k[1, 2, ….., 19]
8. s2= GenRXC(SLUT[i2], i1);
9. SLUT[k]= concatenate(SLUT[k], s2);
10. end
11. end
12. SLUT[1, 2, …., 19]= erase_duplicate(SLUT[1, 2, ……, 19]);
13. SLUT[1, 2, ….., 19]= training_set(SLUT[1, 2, ……, 19]
note: GenRXC( ) can generate patterns for any given RXC values and training_set( ) can return only those patterns that are present in the training set.

Figure 7: Method for generation of SLUTs.

III. SIMULATION RESULTS

The proposed algorithm has been implemented in MATLAB with parameters (N=15, t=19) and tested on several halftone images including Boat, Clock and Lena. The results obtained along with the image quality are shown in Table 2. It can be noticed by comparing Table 2 and Figure 5 that images having lesser pixels in gray-levels of periodic vibratory and zero RXC response have good pixel coverage. It has been observed that the algorithm can distribute 15 "19pels" in parallel among 19 SLUTs with approximately 30% pixel drop. To overcome this loss of pixels we have replicated gray level values from the neighbors, this simple technique is shown in Figure 6.

In calculating PSNR in Table 2, it has been assumed that all "19pels" that can reach their SLUTs can also find their exact inverse halftone values in their tables. Hence, the PSNR only shows the quality of inverse halftones obtained as a result of loss of pixels in the proposed algorithm. A discussion on the quality of inverse halftones obtained through Lookup Table based inverse halftoning can be found in [7].

Technique for pixel compensation:
Img=Inverse Halftoned image after passing through algorithms in Figure 6 and [8]
1. [m,n]=size(Img);
2. for i=1:m
3. for j=1:n
4. if Img(i,j)== missed_pixel
5. Img(i,j)=Img(i,j-1);
6. end

Figure 8: Illustration of technique used for pixel compensation.

Figure 9 shows an image that has been obtained from our algorithm with pixel replication. Lookup table inverse halftoning is assumed to be ideal as before. The original image was halftoned using Floyd and Steinberg ED method. Electronic versions of many images obtained from our algorithm are available at [14].

TABLE 2: SIMULATION RESULTS OF THE PROPOSED ALGORITHM

Image	Halftone Algorithm	% pixel coverage w/o pixel compensation	PSNR Original & Inv. Halftoned with pixel compensation
Boat	FS ED	65.0864	30.3749
Clock	FS ED	70.6667	30.1671
Lena	FS ED	70.9629	28.7139
Boat	GN ED	63.7531	31.2554
Clock	GN ED	69.8765	31.7895
Lena	GN ED	69.7284	29.5628
Boat	EG ED	67.3086	32.1370
Clock	EG ED	68.5926	29.9289
Lena	EG ED	71.0617	28.2293

Note: FS ED: Floyd & Steinberg ED, GN ED: Green Noise ED and EG ED: Edge Enhancement ED (image size 50x50 pixels).

Figure 9: Inverse Halftoned Image (PSNR: 32.5685, 250x250 pixels)

IV. HARDWARE IMPLEMENTATION

The proposed algorithm has been implemented in hardware. Figure 10 shows the block diagram of a system that can implement the proposed algorithm on hardware. The system is pipelined and the arrows in Figure 10 have implicit registers to hold previous block outputs. The SLUTs developed contains gray level values for all "19pels" that occur in images Boat, Clock and Lena The design was captured in VHDL, simulated in Modelsim and then synthesized with Leonardo Spectrum. The results obtained after synthesizes are shown in Figure 11.

For comparison, serial LUT inverse halftoning algorithm of Murat and Vaidyanathan [7] was also implemented. The implementation contains input registers whose outputs goes into five "19pels" to gray-level decoders that together store the contents of a single LUT and the outputs from decoders feed bit to bit OR gates. The OR gates have been decomposed into several clock cycles to reduce the critical path. The maximum frequency for serial design has come to be equal to the implementation of the proposed parallel algorithm. The results of comparison are shown in Table 3.

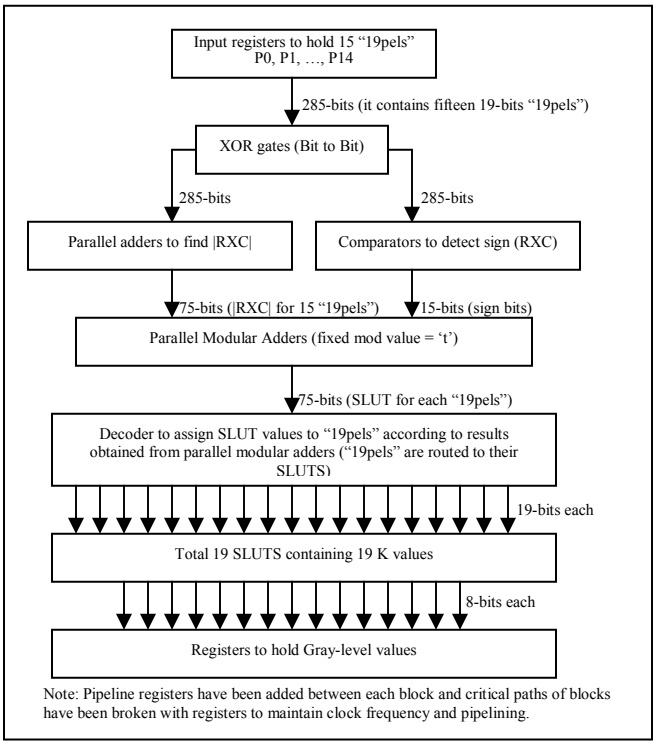

Figure 10: System Block Diagram illustrating our FPGA implementation

Synthesis Results:

Tools used: VHDL and Leonardo Spectrum (with lowest effort and auto optimization):
Clock Frequency: 94.8 MHz Latency: 18 Clock Cycles
Device utilization (device used: 2V8000ff1517):

IOs	48.10%	Global Buffers	06.25%
Function Generators	56.06%	CLB Slices	56.06%
Dffs or Latches	16.80%	Block RAMs	00.00%
Block Multipliers	00.00%		

Figure 11: Synthesis results of the system in Figure 10.

CONCLUDING REMARKS

A new parallel method for Inverse Halftone operation has been successfully designed and implemented. The proposed algorithm is found to be very effective on halftone algorithms that have aperiodic halftone patterns and can be implemented easily on a single FPGA.

ACKNOWLEDGMENT

The authors like to acknowledge King Fahd University of Petroleum & Minerals for all support. We also like to thank Prof. Todd Reed, University of Hawaii at Manoa for his useful advice in improving this paper, and VLSI Design Research Center, Sir Syed University of Engineering & Technology, Pakistan

TABLE 3: COMPARISON OF OUR DESIGN WITH SERIAL LUT METHOD.

	Algorithm in [7]	Proposed Algorithm
Cycles/pixel	1	0.066
LUT size	5.1 K entries	19 K entries
Latency	4 clock cycles	17 clock cycles
Time taken to inverse halftone a 256x256 image	691.3502 ms	45.6389 ms

REFERENCES

[1] Murat Mese and P. P. Vaidyanathan, "Recent Advances in Digital Halftoning and Inverse Halftoning Methods," IEEE Trans. Circuits and Systems I, June 2002.

[2] Ping Wong and Nasir D. Memon, "Image Processing for Halftoning," IEEE Signal Processing Magazine, Vol. 20, July 2003.

[3] S. Hein and A. Zakhor, "Halftone to continuous tone conversion of error diffusion coded image," IEEE Trans. Image Processing, vol. 4, pp. 208-216, Feb. 1995.

[4] Z. Fan, "Retrieval of images from digital halftones," ISCAS, pp. 313-316, May 1992

[5] P. W. Wong, "Inverse halftoning and kernel estimation for error diffusion," IEEE Trans. Image Processing, vol. 4, pp. 486-498, Apr. 95

[6] Z. Xiong, K. Ramchandran and M. Orchard, "Inverse halftoning using wavelets," in Proc. Int. Conf. Image Processing, vol. 1, Lausanne, Switzerland, 1996, pp. 569-572.

[7] M. Mese and Vaidyanathan, "Look up Table (LUT) Method for Inverse Halftoning," IEEE Trans. Image Processing, vol. 10, October 2001.

[8] R. Floyd and L. Steinberg, "An adaptive algorithm for spatial greyscale," Proc. SID, pp. 75-77, 1976

[9] RD.L. Lau, G.R. Arce and N.C. Gallagher, "Green Noise Digital Halftoning," Proceedings of the IEEE, Vol. 86, pp 2424-2442, December 1998.

[10] R. Eschbach and K. Knox, "Error diffusion algorithm with edge enhancement," Journal of Optical Society Am. A, Vol. 8, No. 12, pp. 1844-1850, December 1991.

[11] N. Damera Venkata and Evans, "Adaptive Threshold Modulation for Error Diffusion Halftoning," IEEE Transactions on Image Processing, Vol. 10, No. 1, January 2001

[12] Daniel L. Lau, Robert Ulichney and Gonzalo R. Arce, "Fundamental Characteristics of Halftone Textures: Blue Noise and Green Noise," IEEE Signal Processing Magazine, Vol. 20, July 2003

[13] http://www.ece.utexas.edu/~bevans/projects/halftoning/index.html

[14] http://www.ccse.kfupm.edu.sa/~umair