International Journal of Foundations of Computer Science

A New FM Screening Method to Generate Cluster-dot Binary Image using the Local Exhaustive Search with FPGA Acceleration

YASUAKI ITO
Department of Information Engineering
Hiroshima University
Kagamiyama, Higashi-Hirhoshima, 739-8527, JAPAN

and

KOJI NAKANO
Department of Information Engineering
Hiroshima University
Kagamiyama, Higashi-Hirhoshima, 739-8527, JAPAN

Received (received date)
Revised (revised date)
Communicated by Editor's name

ABSTRACT

Screening is an important task to convert a continuous-tone image into a binary image with pure black and white pixels. The main contribution of this paper is to show a new algorithm for cluster-dot screening using a local exhaustive search. Our new algorithm generates 2-cluster, 3-cluster, and 4-cluster binary images, in which all dots have at least 2, 3, and 4 pixels, respectively. The key idea of our new screening method is to repeat a local exhaustive search that finds the best binary pattern in a small window of a binary image. The experimental results show that the local exhaustive search produces high quality and sharp cluster-dot binary images. We also implemented it on an FPGA to accelerate the computation and achieved a speedup factor of up to 229 over the software implementations.

Keywords: Image Processing; Screening for Printing; Local Search; FPGA-based computing.

1. Introduction

Screening is an important task to convert a continuous-tone image into a binary image with pure black and white pixels [1, 2, 8]. This task is necessary when printing a monochrome or color image by a printer with limited number of ink colors. AM (Amplitude Modulated) screening, a commonly used screening method, arranges black dots in a regular grid and reproduces the intensity of an original continuous-tone image by the number of black pixels in a dot. A black dot involves fewer black pixels to reproduce highlight color, and has more black pixels to create a shadow image. FM (Frequency Modulated) screening, on the other hand, keeps dots of a unit size when converting an original continuous-tone image into the binary image for printing. The intensity level of an original continuous-tone image is reproduced by the density of black unit dots (or pix-

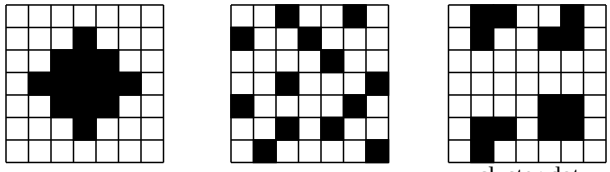

Figure 1: AM screening, FM screening, and cluster-dot screening

els). FM screening pays great attention to generate moiré-free binary images reproducing continuous-tone and fine details of original photographic images. We refer the reader to Figure 1 for illustrations of a dot of AM screening and dots of FM screening. The most well-known FM screening algorithm is Error Diffusion [5] that propagates rounding errors to unprocessed neighboring pixels according to some fixed ratios. Error Diffusion preserves the average intensity level between the original input image and the binary output image. It is also quite fast and often produces good results. However, Error Diffusion may generate worm artifacts, which is a sequence of dots like a worm, especially in the areas of uniform intensity. Several techniques have been developed to prevent artifacts in output binary images [14]. Besides, in Error Diffusion based techniques, the pixel values are propagated to neighbors and the resulting images are defocused.

In applications requiring high fidelity of the printed material (such as printing fine art books, pictorial books, and replicas of paintings), an FM screening method that produces artifact-free higher quality binary images reproducing original work is expected even if the computation takes a lot of time. In our previous paper [7], we have present a new approach for FM screening that we call Local Exhaustive Search (LES for short). Our idea for FM screening is to use the local search technique investigated in the area of combinatorial optimization, which usually takes a lot of computing time. More specifically, the LES produces a binary image whose projected image onto human eyes is very close to the original image. The projected image is computed by applying a Gaussian filter, which approximates the characteristic of the human visual system. We define the total error of the binary image to be the sum of the difference of the intensity levels over all pixels between the original image and the projected image. The LES performs the local exhaustive search for a small square window of size, say, 2×2, 3×3, and 4×4, in the binary image, and finds the best binary image pattern in the window, whose total error is the minimum over all possible binary patterns. After that, a binary subimage in the window is replaced by the best binary image pattern obtained. The local exhaustive search is repeated until no more improvement on the total error is possible. It should be clear that a better binary image can be obtained using a larger window, because search space is larger. As we will show in the experimental results later, we can obtain binary images with fewer total error using a larger window.

Although the LES produces a high quality binary images, they may not produce good printed matter in many practical applications. The generated binary images using the LES contain a lot of isolated black and white dots, which do not exactly appear in printed matter. For example, in laser printers, small isolated black pixels are not printed because toner is

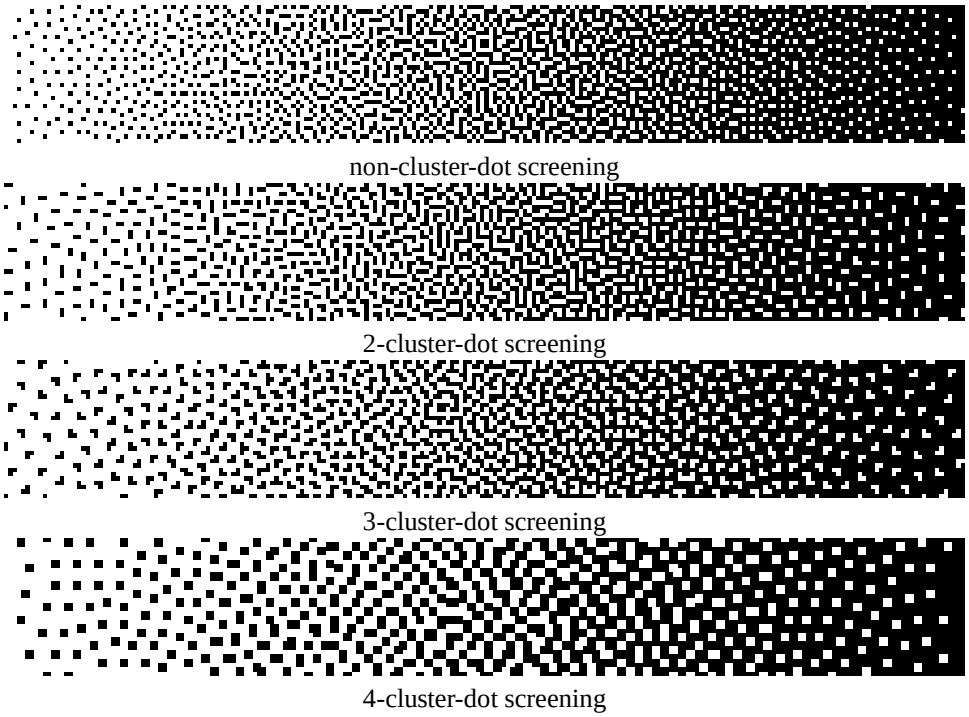

Figure 2: Non-cluster-dot and cluster-dot screening

not transfered for minimum-size dot. In ink-jet printers, a small isolated black dot gains a lot by the ink blur. Also, isolated white dots disappear by the ink blur. Therefore, it is desirable that dots in binary images are *clustered*, that is, all black and white dots have two or more pixels.

The main contribution of this paper is to present an LES-based screening method that generates cluster-dot binary images (Figure 1). In particular, we will show screening methods that generate 2-cluster, 3-cluster, and 4-cluster binary images in which each dot consists of at least 2, 3, and 4 pixels, respectively. Figure 2 shows the resulting binary images of non-cluster-dot screening by the LES [7] and by cluster-dot screening presented in this paper for a ramp image.

The second contribution of this paper is to implement the cluster-dot LES in an FPGA to accelerate the computation. An FPGA (Field Programmable Gate Array) is a programmable VLSI in which a hardware design can be embedded quickly. We have used Nallatech Xtreme DSP kit [13], which is a PCI board with Xilinx VirtexII family FPGA XC2V3000-4 [6], and embedded a circuit to perform the local exhaustive search for a window of size 3×3 and 4×4. To reduce the amount of used FPGA resource and the delay, we use the *instance-specific* approach [3, 4, 11], which embeds a hardware depending on a part of the input instance. The instance-specific approach is applied as follows. One can think that the inputs of LES are an original image and a Gaussian filter. Since a Gaussian filter can be fixed during the computation by LES, we can embed a circuit to perform the local

exhaustive search for a specific Gaussian filter. A round of the LES for a window of size $k \times k$ a Gaussian filter of size $(2w+1) \times (2w+1)$ runs in $O(2^{k^2})$ clock cycles using our FPGA implementation, while it runs $O(2^{k^2} w^2)$ time using the software implementation. Thus, from the theoretical point of view, our FPGA implementation attains a speedup of factor $O(w^2)$ over the software implementation. To show that the actual speed up, we have developed both the FPGA implementation and the software implementation of the LES for cluster-dot FM screening. We have succeeded in accelerating LES by a speedup factor of up to 229 over the software implementation.

This paper is organized as follows. Section 2 formalizes the problem of finding the best binary image of an original gray-scale image as a combinatorial optimization problem. In Section 3, we introduce the cluster-dot screening, as an extension of FM screening and describe it formally. In Section 4, we present the local exhaustive search (LES) method, which is an approximation algorithm for solving the combinatorial optimization problem. Using the LES, we can obtain cluster-dot binary images from continuous-tone images. Section 5 shows an implementation of the LES for cluster-dot screening on an FPGA. In Section 6, we present the experimental results for screening images. Section 7 offers concluding remarks.

2. FM Screening based on the Human Visual System

This section defines the problem of finding the best binary image of an original gray-scale image as a combinatorial optimization problem. The basic idea is shown in our previous paper [7].

Suppose that an original gray-scale image $A = (a_{i,j})$ of size $n \times n$ is given, where $a_{i,j}$ denotes the intensity level at position (i, j) $(1 \leq i, j \leq n)$ taking a real number intensity in the range $[0, 1]$. The goal of screening is to find a binary image $B = (b_{i,j})$ of the same size that reproduces the original image A, where each $b_{i,j}$ is either 0(black) or 1(white). We measure the goodness of the output binary image B using the Gaussian filter that approximates the characteristic of the human visual system. Let $G = (g_{k,l})$ denote a Gaussian filter, i.e. a 2-dimensional symmetric matrix of size $(2w+1) \times (2w+1)$, where each non-negative real number $g_{k,l}$ $(-w \leq k, l \leq w)$ is determined by a 2-dimensional Gaussian distribution such that their sum is 1. In other words,

$$g_{k,l} \quad = \quad c \cdot e^{-\frac{k^2+l^2}{2\sigma^2}} \tag{1}$$

where σ is a parameter of the Gaussian distribution and c is a fixed real number to satisfy $\sum_{-w \leq k,l \leq w} g_{k,l} = 1$. Let $R = (r_{i,j})$ be the projected gray-scale image of a binary image $B = (b_{i,j})$ obtained by applying the Gaussian filter as follows:

$$r_{i,j} \quad = \quad \sum_{-w \leq k,l \leq w} g_{k,l} b_{i+k,j+l} \quad (1 \leq i, j \leq n) \tag{2}$$

Clearly, from $\sum_{-w \leq k,l \leq w} g_{k,l} = 1$ and $g_{k,l}$ is non-negative, each $r_{i,j}$ takes a real number in the range $[0, 1]$ and thus, the projected image R is a gray-scale image. We can say that a binary image B is a good approximation of original image A if the difference between A and R is small enough. Hence, we are going to define the Gaussian error of B as follows.

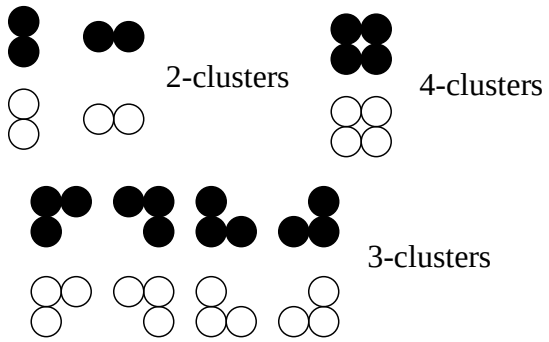

Figure 3: 2-cluster, 3-cluster, and 4-cluster dots

Gaussian error $e_{i,j}$ at each pixel location (i, j) is defined by

$$e_{i,j} \quad = \quad a_{i,j} - r_{i,j}, \tag{3}$$

and the total Gaussian error is defined by

$$Error(A, B) \quad = \quad \sum_{1 \le i,j \le n} |e_{i,j}|. \tag{4}$$

Since the Gaussian filter approximates the characteristics of the human visual system, we can think that image B reproduces original gray-scale image A if $Error(A, B)$ is small enough. The best binary image that reproduces A is a binary image B is given by the following formula:

$$B^* \quad = \quad \arg\min\{Error(A, B) \mid B \text{ is a binary image}\}. \tag{5}$$

The best binary image may have dots with isolated pixels. For example, let A be a binary image of size $n \times n$ with every pixel having intensity $\frac{1}{2}$. Then, the best binary image B^* satisfying (5) is a checkerboard, in which all pixels are isolated.

3. Cluster-dot Screening

This section shows our new idea to generate good cluster-dot binary images.

The idea for generating cluster-dot binary images is to give appropriate restriction when the best binary image is computed. We say that two black pixels form a *2-cluster* if they are adjacent in either vertical or horizontal direction(Figure 3). A binary black pixel is *2-cluster* if it is one of the pixels in a 2-cluster. In other words, if a black pixel has a black neighbor pixel, it is 2-cluster. Similarly, a binary white pixel is *2-cluster* if one of its neighbor is white. A binary image is *2-cluster* if its all pixels are 2-clusters. Clearly, a 2-cluster binary image has no isolated dot, and each dot consists of either more than one white pixels or more than one black pixels.

Similarly, we can define 3-cluster and 4-cluster binary images. We say that 3 black pixels form a *3-cluster* if they are in a 2×2 region(Figure 3). A binary black pixel is

3-cluster if it is one of the pixels in a 3-cluster. We can define *3-cluster* for white pixels in the same way. A binary image is 3-cluster if all the pixels are 3-cluster. We also say that 4 black pixels form a 4-cluster if they are in 2×2 region(Figure 3). A binary black pixel is *4-cluster* if it is one of the pixels in a 4-cluster. *4-cluster* for white pixels can be defined in the same way. A binary image is 4-cluster if all the pixels are 4-cluster.

Note that isolated 3 black pixels in a 3×1 or 1×3 region do not form 3-cluster although they are adjacent. The reason is that, a dot with large diameter gains a lot. For example, 3 black pixels in a 3×1 region has diameter $\sqrt{3^2 + 1^2} = \sqrt{10}$, while each 3-cluster in Figure 3 has diameter of $\sqrt{2^2 + 2^2} = \sqrt{8}$. Hence, we exclude 3 black pixels in a 3×1 or 1×3 region from the 3-cluster. By the same reason, we exclude 4 pixels that do not fit in a 2×2 region from the 4-cluster.

Suppose that a gray scale image A is given. Let c be either 2, 3, or 4. It should be clear that the best c-cluster binary image that reproduces A is a binary image $B^{\mathrm{c-cluster}}$ such that

$$\begin{aligned} B^{\mathrm{c-cluster}} &= \arg\min\{ \mathit{Error}(A, B) \mid \\ &\qquad B \text{ is a c-cluster binary image}\}. \end{aligned} \tag{6}$$

Our goal is to find a c-cluster binary image $B^{\mathrm{c-cluster}}$ for a given gray scale image A.

4. The Local Exhaustive Search for Cluster-dot Screening

The main purpose of this section is to present our new algorithm to find a good binary image B whose total error with respect to original gray-scale image A may not be minimum but small enough. Our approach to obtain c-cluster-dot binary image ($c = 2, 3$, or 4) updates a small square region of a temporal binary image by the best binary pattern, in which the total number of non-c-cluster dots and the total Gaussian error is the minimum over all possible binary patterns.

For a binary image B of size $n \times n$, let $C_{\mathrm{c-cluster}}(B)$ ($c = 2, 3$ or 4) denote the number of non-c-cluster pixels in B. Note that if B is c-cluster then $C_{\mathrm{c-cluster}}(B) = 0$. Hence, the goal of c-cluster-dot screening is to find a good binary image B satisfying $C_{\mathrm{c-cluster}}(B) = 0$. Suppose that an original image A of size $n \times n$ are given. The error $\mathit{Error}_{\mathrm{c-cluster}}$ of a binary image B with respect to A is defined as follows:

$$\mathit{Error}_{\mathrm{c-cluster}}(A, B) \quad = \quad (C_{\mathrm{c-cluster}}(B), \mathit{Error}(A, B)).$$

In other words, the error is a pair of "the number of non-c-cluster pixels in B" and "the difference between A and the projected image of B". We assume that the comparison of any two values of $\mathit{Error}_{\mathrm{c-cluster}}(A, B)$ are based on the lexicographical order, that is, $\mathit{Error}_{\mathrm{c-cluster}}(A, B) < \mathit{Error}_{\mathrm{c-cluster}}(A, B')$ if and only if

- $C_{\mathrm{c-cluster}}(B) < C_{\mathrm{c-cluster}}(B')$ or,
- $C_{\mathrm{c-cluster}}(B) = C_{\mathrm{c-cluster}}(B')$ and $(\mathit{Error}(A, B) < \mathit{Error}(A, B'))$

The idea of our new screening algorithm is to find a binary image B with small error $\mathit{Error}_{\mathrm{c-cluster}}(A, B)$.

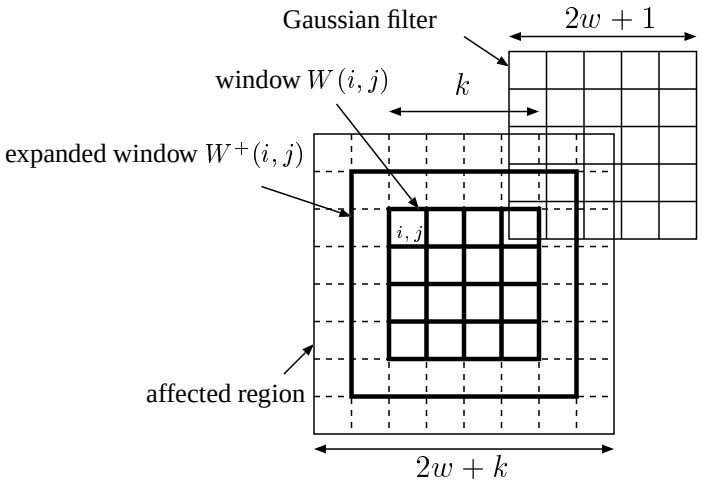

Figure 4: Illustrating a window of size $k \times k$, the expanded window of size $(k+2) \times (k+2)$, a Gaussian filter of size $(2w+1) \times (2w+1)$, the affected region of size $(2w+k) \times (2w+k)$

Suppose that an original image A and a temporary binary image B is given. Further, let $W(i,j)$ be a window of size $k \times k$ in B whose top-left corner is at position (i,j). Our first idea is to compute the error for all 2^{k^2} binary patterns in $W(i,j)$ and replace the current binary subimage in the window by the best binary pattern that minimizes the total error. In other words, we find a binary image B' such that

$$B' \quad = \quad \arg\min\{\mathit{Error}_{c-cluster}(A,B) \mid B \text{ and } B' \text{ differ only in } W(i,j)\}. \qquad (7)$$

Next, let us see the details on how B' satisfying formula (7) above is computed. Since we use a Gaussian filter of size $(2w+1) \times (2w+1)$, the change of the binary pattern affects the errors in a square region of size $(2w+k) \times (2w+k)$, which we call the *affected region* (Figure 4). Also, let $W^+(i,j)$ be a region of size $(k+2) \times (k+2)$ which can be obtained by expanding the window $W(i,j)$ by one pixel. We call $W^+(i,j)$ the expanded window of $W(i,j)$. It should be clear that the best binary pattern can be selected by computing

- the total Gaussian errors of the affected region of size $(2w+k) \times (2w+k)$, because the change of the binary pattern does not affect errors at pixels outside the affected region, and
- the number of non-c-cluster pixels in $W^+(i,j)$, because the change of the binary pattern does not affect the number of non-c-cluster pixels outside $W^+(i,j)$.

Let us evaluate the computing time necessary to find the best binary pattern in the window. The Gaussian error of a particular pixel in an affected region can be computed in $O(k^2)$ time by evaluating formulas (2) and (3). Hence all the Gaussian errors in the affected region can be computed in $O(k^2(2w+k)^2)$ time. After that, their sum can be computed in $O((2w+k)^2)$ time. Also, it can be determined in $O(1)$ time if a particular pixel is c-cluster easily. Hence the total number of non-c-cluster pixels in the expanded

window of size $(k+2) \times (k+2)$ can be computed in $O(k^2)$ time. Thus, the total error in the affected region, which is a pair of the total number of non-c-cluster pixels in the expanded window and the total Gaussian error in the affected region can be computed in $O(k^2(2w+k)^2) + O(k^2) = O(k^2(w+k)^2)$ time.

Since we need to check all the possible 2^{k^2} binary patterns, the best binary pattern can be obtained in $O(2^{k^2}k^2(w+k)^2)$ time. We can improve the computing time by flipping a pixel in the order of the gray code of binary numbers. Recall that the gray code represents a list of all m-bit binary numbers such that any two adjacent numbers differ only one position. Thus, by flipping an appropriate bit using the gray code, we can list all the 2^m binary numbers with m bits. Using the gray code with k^2 bits, we can evaluate the errors for all binary patterns in $O(2^{k^2}w^2)$ time as follows. Starting with the current pixel pattern in the window, we repeat flipping an appropriate pixel according to the gray code. In each flipping operation, we compute the total Gaussian error in the affected region for the current binary pattern in the window. Since the flipping operation for a single bit affects the Gaussian error of $(2w+1) \times (2w+1)$ pixels, the total Gaussian error can be computed in $O(w^2)$ time in an obvious way. Also, we can compute the change of the number of non-k-cluster pixels by checking the number of non-k-cluster pixels in 3×3 pixels whose center is the flipped pixel. This computation takes $O(1)$ time. Thus, the best binary pattern can be computed in $O(w^2) \times 2^{k^2} = O(2^{k^2}w^2)$ time by the local exhaustive search.

We are now in position to show our new screening method. Let $A = (a_{i,j})$ be an original gray-scale image and $B_0 = (b^0_{i,j})$ be an appropriate initial binary image. Although we can initialize the binary image B_0 using any screening method, we assume that B_0 is initialized by the *random dither method*. In the random dither method, a binary pixel takes value 1 with probability p if the pixel value of the corresponding pixel of an original image is p ($\in [0,1]$). Thus, $b^0_{i,j} = 1$ with probability $a_{i,j}$ for every i and j. We repeat sliding a window of size $k \times k$ and improving the binary pattern in the window by replacing the pixel values in it by the best binary pattern. We perform window sliding in the raster scan order as illustrated in Figure 5, to obtain a better quality binary image B_1. The same procedure is repeated, that is, the window sliding operation is applied to B_{t-1} and obtain a better binary image B_t ($t \geq 1$) until B_{t-1} and B_t are identical and no more improvement is possible. When computing B_t for $t \geq 2$, we do not have to perform the exhaustive search for all the windows. If the projected image of the affected region for the current window did not change, then we can omit the exhaustive search.

The details of our new screening algorithm are spelled out as follows:

Local Exhaustive Search(A)
Set an appropriate initial binary image in B_0;
$B_1 \leftarrow B_0$;
for $i \leftarrow 1$ to $n-w+1$ do
  for $j \leftarrow 1$ to $n-w+1$ do
    Perform the exhaustive search in $W(i,j)$ for B_1
    and update B_1 by the best binary pattern.
$t \leftarrow 1$;
do {
  $t \leftarrow t+1$;

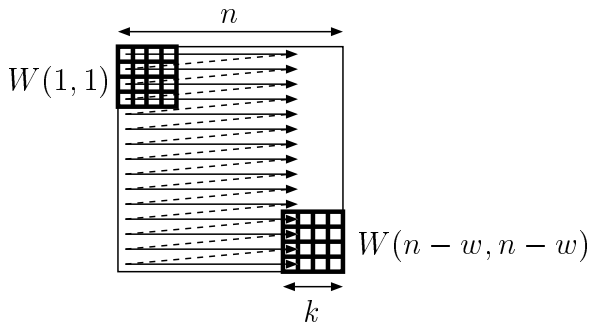

Figure 5: Sliding window in raster scan order

$B_t \leftarrow B_{t-1}$;
for $i \leftarrow 1$ to $n - w + 1$ do
  for $j \leftarrow 1$ to $n - w + 1$ do
    If the projected image in the affected regions of
    $W(i, j)$ for R_t and R_{t-1} are not identical then
    perform the exhaustive search in $W(i, j)$ for B_t
    and update B_t by the best binary pattern.
} until (B_t and B_{t-1} are identical)
output (B_t);

5. Hardware Acceleration for the cluster-dot LES using an FPGA

We have developed a hardware accelerator using the PCI-connected FPGA that performs the local exhaustive search in order to find the best binary pattern in a window. This section is devoted to show the architecture of our FPGA-based hardware accelerator.

Let $A = (a_{i,j})$ be an original gray-scale image of size $n \times n$ and $B = (b_{i,j})$ be the current binary image of A. Before showing the architecture, we first show how our hardware accelerator is used by the host PC. The hardware accelerator is used to compute the best binary pattern in a window $W(i, j)$. For this purpose, the host PC sends necessary information to the hardware accelerator. Let $W^{++}(i, j)$ be the extended region of $W^{+}(i, j)$. Note that, if the original window has $k \times k$ pixels, then $W^{++}(i, j)$ has $(k + 4) \times (k + 4)$ pixels. For the purpose of compute the best binary pattern in $W(i, j)$ the host PC sends, to the hardware accelerator,

- the current values of binary pixels in $W^{++}(i, j)$, and
- the current Gaussian errors of all pixels in the affected region of $W(i, j)$.

The hardware accelerator computes the errors for all possible 2^{k^2} binary patterns in $W(i, j)$ and returns the best binary pattern whose error is the minimum. The host PC receives the best binary pattern and updates the values of binary pixels by the received binary pattern.

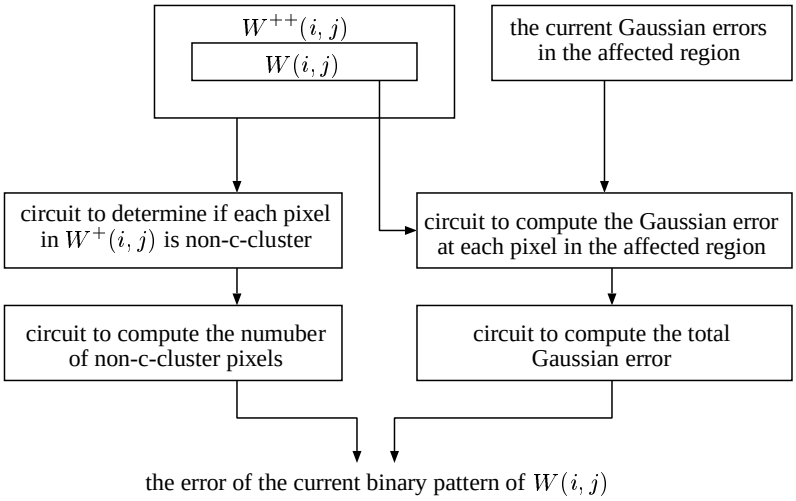

Figure 6: Illustrating a part of the hardware accelerator.

We are going to show the architecture of our hardware accelerator. Figure 6 illustrates a part of the FPGA-based hardware accelerator, which outputs the error for every binary pattern. To list all possible 2^{k^2} binary patterns in $W(i,j)$, we simply use k^2-bit binary counter. We also use a circuit that checks if each pixel in $W^{+}(i,j)$ is a non-c-cluster from the current binary pixel values in $W^{+} + (i,j)$. After that, the number of non-c-cluster is computed by a binary summing circuit, which can be implemented efficiently on the FPGA [9, 10, 12]. Next, we need to compute the total Gaussian errors. For this purpose, we use a circuit that to compute the Gaussian error at each pixel in the affected region, which can be implemented using integer addition/subtraction circuits. After that the sum of the Gaussian errors are computed by an integer summing circuit, which can also be implemented efficiently on the FPGA [12]. In this way, the error of the current binary pixel values of $W(i,j)$ which is a pair of the number of non-c-cluster pixels and the total Gaussian errors is computed. By using the k^2-bit binary counter for $W(i,j)$, we can compute, in 2^{k^2} clock cycles, the best binary pattern whose error takes the minimum all possible 2^{k^2} binary pattern.

6. Experimental Results

This section shows the experimental results. We have developed a software that performs our cluster-dot screening based on the LES for window of size 1×1, 2×2, 3×3, and 4×4, which call *LES1*, *LES4*, *LES9*, and *LES16*, respectively. We have used a Pentium4-based PC (Xeon 4.0GHz) with Linux operating system (Kernel 2.6) for software implementation. The source program is compiled by gcc 3.4.6 with -O2 and -m64 options. We also developed FPGA-based implementation for window of size 3×3, and 4×4, that is, LES9 and LES16. In the FPGA implementation, we have used a PCI board with Xilinx VirtexII family FPGA XC2V3000-4 [6], and embedded a circuit to perform the cluster-dot

local exhaustive. The source program is developed using Verilog HDL, and Xilinx ISE 8.2i for logic synthesis, mapping, and implementation. The timing analysis by Xilinx ISE 8.2i reported all of our implementations run in 101.70 MHz and 102.23 MHz for LES9 and LES16, respectively. Thus, we set the clock frequency the programmable oscillator on the FPGA board to 100MHz.

We have used an 8-bit gray-scale images "Lena" of size 256×256 for the experiment. Also, we use a window of size 1×1, 2×2, 3×3, and 4×4 and a Gaussian filter of size 11×11 and parameter $\sigma = 1.2$. Table 1 shows the computing time for non-cluster, 2-cluster, 3-cluster, and 4-cluster screening concerning LES1, LES4, LES9, and LES16, respectively. In software implementations for the same LES, non-cluster screening runs faster because it is not necessary to check if a pixel is c-cluster. On the other hand, in the FPGA implementation, the computing time of LES9 is almost the same for non-cluster, 2-cluster, 3-cluster, and 4-cluster. This is because the local exhaustive search for a window takes 2^9 clock cycles in the same clock frequency 100MHz, regardless the size of clusters. By the same reason, the computing time of the FPGA implementation of LES16 is the same for all cluster sizes. Also, the speed up by the FPGA implementation for LES16 is much larger than that for LES9, because the LES9 has larger overhead between the communication the host PC and the PCI-connected FPGA board than LES16. More specifically, the FPGA implementation for LES9 performs communication between the host PC and the FPGA board in every 2^9 clock cycles, while that for that for LES16 performs it in every 2^{16} clock cycles. Hence, the communication overhead of LES9 is much larger than that of LES19.

Figure 7 illustrates the error defined by (4). Note that a binary image with smaller total error reproduces better its original image. Also, the errors of 4-cluster-dot are the largest because dots are coarse. However, in practical printing environment, 4-cluster-dot images may reproduce the original image better than the others. From figure 7 LES16 produces binary images with higher quality, because it repeats finding the best binary image with larger windows. The readers should refer to Figure 8 for the resulting binary images using LES16. Clearly, every pixel in the binary images of "Lena" by 2-cluster-dot, 3-cluster-dot, and 4-cluster-dot screening is 2-cluster, 3-cluster, and 4-cluster, respectively. Also, they reproduces the original gray scale image "Lena" very well.

7. Conclusions

We have presented the local exhaustive search based cluster-dot screening for finding a high quality binary image that reproduce the original gray-scale image. Since this screening process requires huge amount of computation, it would be impractical if we would implement it naively. Although the processing time is still much larger than that of the currently used screening algorithms such as Error Diffusion and also much larger hardware resources are required, our algorithm would be useful in applications requiring high fidelity binary images.

References

1. M. Analoui and J.P. Allebach. Model-based halftoning by direct binary search. In *Proc. SPIE/IS&T Symposium on Electronic Imaging Science and Technology*, volume 1666, pages

Table 1: The computing time for screening “Lena” using the software and FPGA implementation

		non-cluster	2-cluster	3-cluster	4-cluster
LES1	Software	1.08×10^0sec	1.14×10^0	1.13×10^0	0.64×10^0
LES4	Software	0.84×10^1sec	0.96×10^1sec	1.05×10^1sec	0.98×10^1sec
LES9	Software	2.36×10^2sec	3.09×10^2sec	3.23×10^2sec	3.45×10^2sec
	FPGA (slices)	1.52×10^1sec (4381)	1.53×10^1sec (4489)	1.56×10^1sec (4790)	1.55×10^1sec (4511)
	Speedup	15.5	20.2	20.7	22.3
LES16	Software	5.40×10^4sec	6.08×10^4sec	6.71×10^4sec	6.74×10^4sec
	FPGA (slices)	3.14×10^2sec (5545)	2.97×10^2sec (5658)	2.97×10^2sec (5752)	2.94×10^2sec (5726)
	Speedup	172	205	226	229

96–108, 1992.

2. T. Asano. Digital halftoning: Algorithm engineering challenges. *IEICE Transactions on Information and Systems*, February 2003.
3. J. L. Bordim, Y. Ito, and K. Nakano. Accelerating the CKY parsing using FPGAs. *IEICE Transactions on Information and Systems*, E86-D(5):803–810, May 2003.
4. J. L. Bordim, Y. Ito, and K. Nakano. Instance-specific solutions to accelerate the cky parsing for large contex-free grammars. *International Journal on Foundations of Computer Science*, pages 403–416, 2004.
5. R. W. Floyd and L. Steinberg. An adaptive algorithm for spatial gray scale. *SID 75 Digest,Society for Information Display*, pages 36–37, 1975.
6. Xilinx Inc. *Virtex-II Platform FPGAs: Complete Data Sheet*, 2003.
7. Y. Ito and K. Nakano. Fm screening by the local exhaustive search, with hardware acceleration. In *Proc. of Workshop on Advances in Parallel and Distributed Computatinal Models (CD-ROM of International Parallel and Distributed Processing Symposium)*, 2004.
8. D. L. Lau and G. R. Arce. *Modern Digital Halftoning*. Marcel Dekker, 2001.
9. R. Lin, K. Nakano, S. Olariu, M. C. Pinotti, J. L. Schwing, and A. Y. Zomaya. Scalable hardware-algorithms for binary prefix sums. *IEEE Transactions on Parallel and Distributed Systems*, 11(8):838–850, 8 2000.
10. D. E. Muller and F. P. Preparata. Bounds to complexityies of network for sorting and for switching. *J. ACM*, 22:195–201, 1975.
11. K. Nakano and E. Takamichi. An image retrieval system using FPGAs. *IEICE Transactions on Information and Systems*, E86-D(5):811–818, May 2003.
12. K. Nakano and K. Wada. Integer summing algorithms on reconfigurable meshes. *Theoretical Computer Science*, pages 57–77, January 1995.
13. Nallatech. *Xtreme DSP Development Kit User Guide*, 2002.
14. V. Ostromoukhov. A simple and efficient error-diffusion algorithm. In *Proc.of the 28th SIGGRAPH*, pages 567 – 572, 2001.

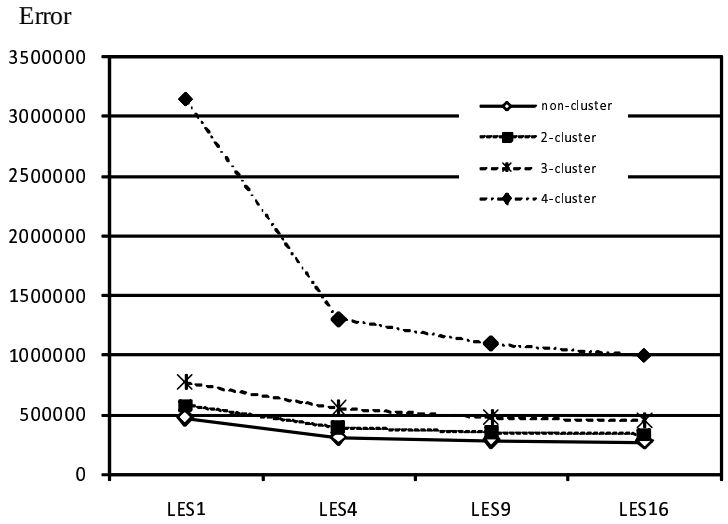

Figure 7: The error of the resulting binary image with respect to the original image “Lena”

Figure 8: The resulting binary images of non-cluster-dot and cluster-dot screening for "Lena" using LES16