A Self-Correcting Active Pixel Camera *

Israel Koren, Glenn Chapman† and Zahava Koren

Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA 01003
E-mail: koren@ecs.umass.edu
† School of Engineering Science
Simon Fraser University, Burnaby BC, Canada V5A 1S6
E-mail: glennc@cs.sfu.ca

Abstract

Digital cameras on-a-chip are becoming more common (e.g., [4, 5]) and are expected to be used in many industrial and consumer products. With the size of the CMOS active pixel-array implemented in such chips increasing to 512×512 *and beyond [5], the possibility of degradation in the reliability of the chip over time must be a factor in the chip design.*

In digital circuits, a commonly used technique for reliability or yield enhancement is the incorporation of redundancy (e.g., adding redundant rows and columns in large memory ICs). Very limited attempts have been directed towards fault-tolerance in analog circuits, mainly due to the very high level of irregularity in their design. Since active pixel arrays have a regular structure, they are amenable to reliability enhancement through a limited amount of added redundancy. The purpose of this paper is to investigate the advantages of incorporating some fault-tolerance methods, including redundancy, into the design of an active pixel sensor array.

1: Introduction

In remote, hard to reach and dangerous environments like outer space, high radiation areas, mine and military action fields, digital cameras offer an important ability to image the scene at low cost and risk. However, these same locations put much more stress on the camera system (through radiation, heat, pressure) possibly leading to partial failures. At the same time, the location makes it difficult or impossible to replace failed imagers. Hence there is considerable advantage to creating fault-tolerant, self-correcting imagers.

Traditional CCD image chips are very susceptible to failures. Since pixel data is passed down columns through other pixels, this means that single pixel failures can lead to loss of whole columns above some point. As was noted in [1], Active Pixel Sensors (APS) are CMOS type imagers that involve row and column addressed pixels and thus offer more flexibility to apply redundancy. The size of APS is increasing to 512×512 pixels and beyond [4]. Moreover, their area is increasing as several commercial APS are approaching standard 35mm film sizes (35×23.3mm) [2, 6]. Indeed it is notable that all these larger designs are APS rather than CCD devices, showing their better scalability in area. Such large areas and high pixel count devices make the possible degradation of their reliability over time a factor that must be taken into account in the chip design.

* This work was supported in part by JPL, under contract 961294, and by NSF, under contract MIP-9710130.

This paper explores various ways of creating self-correcting imagers, first with a software technique, then with a hardware modification of the APS design and finally with a combination of the two. The APS described in [1] used a pixel split in half with the addition of laser links/cuts to design a large area 40×30 mm device with good yield. The current paper demonstrates that a modification of that design, combined with some off chip signal processing, can make for a robust self-healing imaging chip.

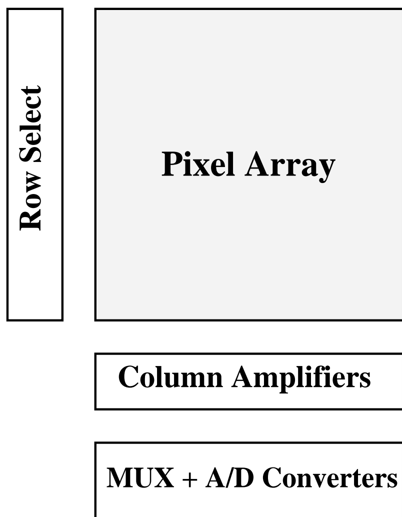

Figure 1. The digital camera-on-a-chip.

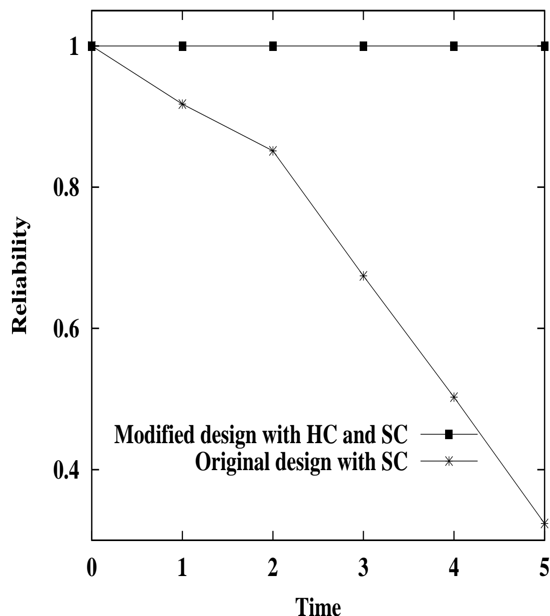

Figure 2. Reliability of a 512×512 array as a function of time ($\lambda = 2 \cdot 10^{-4}$).

The block diagram of a digital camera-on-a-chip including an $M \times M$ pixel array is shown in Figure 1. As the APS area and number of pixels increase, the area of the pixel array becomes large relative to other chip sections (i.e., the Amps, Mux, ADC and row select - see Figure 1). In the 35mm film example [1], the pixels constitute over 90% of the chip area. Hence, as a first approximation the reliability analysis in this paper will concentrate on the pixel array.

Due to the large number of pixels, the device reliability is expected to deteriorate rapidly, and incorporating some fault-tolerance into it is therefore justified. Two fault-tolerance methods, one software based and the other hardware based, are described in Section 2. Sections 3 and 4 provide further details on the hardware and software correction methods, respectively. A detailed analysis of the two methods and some of our numerical results are presented in Section 5, and Section 6 concludes the paper.

2: Fault model and fault-tolerance methods

The fault model we are using here assumes that each pixel has faults arriving according to a Poisson process with a rate of λ faults per unit time, and that the pixels are statistically independent with regard to faults. Hence, the probability of a given pixel to be fault-free at time t is $e^{-\lambda t}$ and the probability of the whole array to be fault-free at time t is $e^{-M^2 \lambda t}$. This last expression represents the array reliability when no fault-tolerance exists, and it deteriorates very fast, even for small failure rates λ, due to the large value of M^2.

A preliminary study was reported in [3], evaluating the benefits of adding redundant rows and columns to the pixel array. Unlike in memory arrays, a spare row (column) of pixels can only replace a boundary row (column) rather than any faulty row (column). Consequently, this technique has limited capabilities and is satisfactory only for long duration missions and high image quality requirements. We suggest here, therefore, two other fault-tolerance techniques. In [3] we called an $M \times M$ array "acceptable" if every one of its $m \times m$ sub-arrays had at most one faulty pixel, where m is a parameter determined by the image quality requirements. The larger the value of m, the smaller the number of faulty pixels allowed, and the higher the quality of the digital image. In this paper, we require that m be equal to 2, which is equivalent to saying that all eight immediate neighbors of a faulty pixel are fault-free. If this is the case, the value of a faulty pixel can be replaced by some combination of the values of the eight pixels surrounding it. This method is especially suitable for digital images, since clearly a pixel value is correlated to that of its immediate neighbors (although the exact nature of the correlation depends on the type of images being photographed). This method is software based and does not require any hardware support. We denote it by *SC (Software Correction)*. The reliability of an $M \times M$ pixel array at time t is, therefore, defined now as the probability that every 2×2 sub-array has at most one faulty pixel, and the whole array can thus be corrected using SC. This probability is unfortunately, very difficult to calculate analytically, and we calculate it therefore, using Monte-Carlo simulation.

The lower curve in Figure 2 shows the reliability of a 512×512 pixel array employing the software-based scheme as a function of time for a failure rate of $\lambda = 2 \cdot 10^{-4}$. This curve shows that even with the use of the SC scheme, the reliability still deteriorates quite rapidly. We therefore suggest in what follows, an additional, hardware based, fault-tolerance technique.

A recent implementation of a novel defect-tolerance technique which was intended for yield improvement [1] can be adapted to provide fault-tolerance for reliability enhancement. The basic approach splits the photodiode in each pixel into two half-size photodiodes (explained in the next section) and duplicates the readout transistors so that one photodiode can be used even if the second one is defective. After testing the entire pixel array under full dark and full light images, the pixels with half intensity can be identified and their value can then be multiplied by two. We denote this method by *HC (Hardware Correction)*.

A pixel in this new design is considered faulty if both its halves are faulty, and non-faulty if at least one half of it is operational. A faulty pixel can still be software-corrected by using a combination of its eight non-faulty neighbors. The combination of both fault-tolerance methods (SC and HC) increases the reliability of the array significantly, as can be seen in the upper curve of Figure 2. In this and the remaining figures, "original design" refers to the design in which the pixel is not divided into two, and thus, only SC is possible. The "modified design" refers to the design in which every pixel is divided into two, and both HC (if only one half of the pixel is faulty) and SC (if both halves are faulty) are used. The reliability values in the upper curve in Figure 2 were calculated using $\lambda = 2 \cdot 10^{-4}$ for a half-pixel (i.e., assuming, conservatively, that the failure rate for a half-pixel is the same as the failure rate for a whole pixel in the original design). The reliability of the new design decreases only when the mission time is extremely long (compared to the original design). The next two sections provide further details regarding the HC and the SC techniques.

3: The hardware correction method

Figure 3 shows the schematic of the a single APS cell, with the split diodes, while Figure 4 shows the resulting layout. In the APS, the photodiodes charge the gates of the readout transistors (M2.1, M2.2). When the row select transistor (M3) is activated, current flows through the readout transistors to the column output. The diodes are then reset to a precharged level. Since both readout transistors (M2.1, M2.2) are in parallel on the same column output line, this redundancy

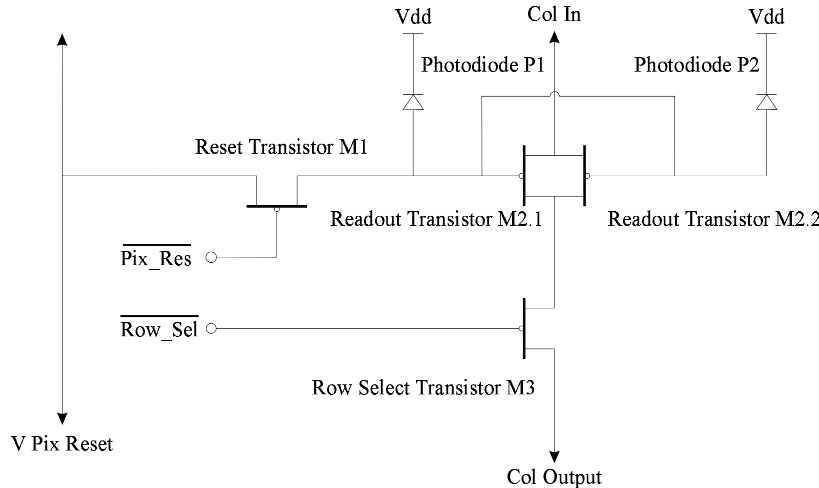

Figure 3. The modified design of the pixel cell.

requires no additional circuitry at the output and less area compared to making two photodiodes that are read separately.

By splitting the photodiodes and readout transistors in an APS cell in half, the cell becomes much more reliable. Yet the cost in devices behavior is small - only a 6% reduction in cell photodiode area which results in a correspondingly lower sensitivity to light.

Since most of the cell area is taken up in the diodes and readout transistors let us examine the most probable failures in these components. Generally there are three possible failure modes for a photodiode/readout: (1) Low sensitivity (something covering part of the photodiode, leakage in the photodiode, poor transfer characteristics of the transistor etc). (2) Stuck low: the readout transistor passes no current (photodiode shorted, gate to photodiode path cut, transistor stuck off). (3) Stuck high: the readout transistor is always on (photodiode always charged, transistor stuck on).

Case (1) (low sensitivity) is actually just a variation on the general problem of pixel sensitivity variation across the device. In a single APS, the stuck cases (2,3) become dead pixels. However, in the split pixel design there is an important advantage. If only one half pixel is faulty, good data can still be extracted from the pixel. There are then five possible cases:

(1) Both pixels active: full pixel sensitivity (0.0-1.0 output range) (2) one half stuck low: half pixel sensitivity (0.0-0.5 output range) (3) one half stuck high: half pixel sensitivity (0.5-1.0 output range) (4) both halves stuck low: dead pixel (output constantly near zero) (5) both halves stuck high: dead pixel (output constantly near 1).

It is important to recognize that all these faults can be identified with the previously mentioned two simple and common tests done to calibrate the sensor. It is standard practice to take a dark field image (imager without light) to identify base noise levels for subtraction and a light field (fully illuminated) image to calibrate sensor sensitivity. The light field image will identify low sensitivity pixels (1), half stuck low (2) and fully dead pixels (5). While the dead pixel data cannot be recovered, the other two would be taken care of in the sensitivity adjustment calculations. Indeed the half stuck is just a simple case of multiplying by 2.

The half stuck high (3) and fully stuck high (4) pixels are identified from the dark field. In the full stuck case the dark field subtraction sets the pixel to 0. In the half stuck case the dark field subtraction reduces the pixel to the half sensitivity case. These really are calibration related corrections that are best done by the post image processing, as with the software correction.

This design results in a cell where with no additional active circuitry but yet fails only under the following conditions. In the cases where both photodiodes or both readout transistors are faulty

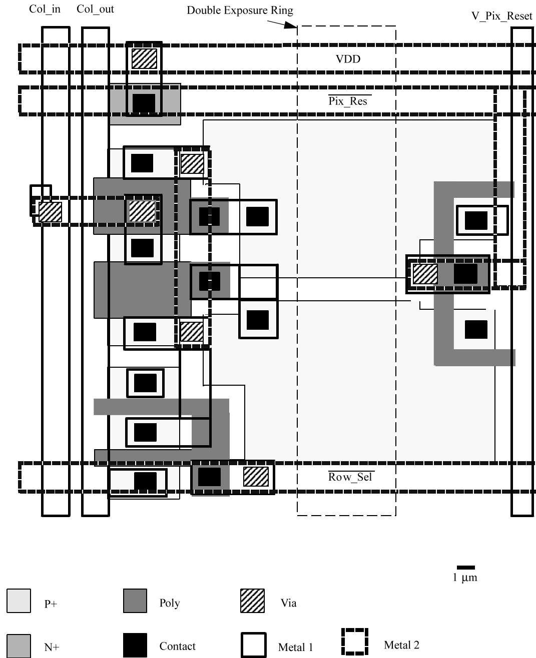

Figure 4. The layout of the modified pixel cell.

then the cell is unreadable as noted. The later is a rare event since on the basis of area, 79% of the device is the photodiode and only 4% is used by the readout transistors. The other 17% of the cell is the remaining circuitry almost all of that is the control lines, which are less likely to fail after production. Only 4% of the cell area can be considered common active control circuitry (the row select and reset transistors) which are thus likely to cause a common failure of the cell (affecting the output of both halves).

In harsh environments it may be desired to remove even this small area of single failure point in the split cell. This can be done by making both reset and row select transistors separate devices for both halves, as shown in Figure 5. For the reset, each diode has a separate transistor (M2.1 and M2.2). The two row select transistors in parallel now form separate current feeds to the col_output line. This takes another 4% of the pixel cell area, and thus reduces the light sensitivity accordingly. This would result in a cell that is very robust against failures.

The modified pixel cell has thus a very small probability of a total failure during operation, and even those failures are easily detectable and correctable with the SC scheme.

4: The software correction method

We consider in this paper three forms of software corrections. In the first, SC^1, the value of the missing pixel is replaced by the arithmetic mean of its eight neighbors. In the second, SC^2, it is replaced by the arithmetic mean of its four immediate neighbors only. The third approximation, SC^3, is based on fitting a quadratic function to the nine pixels in question.

To be more specific, we use the following notations. We denote the coordinates of the faulty pixel by (0,0), and those of its eight neighbors by (0,1), (1,1), (1,0), (1,-1), (0,-1), (-1,-1), (-1,0), (-1,1). We denote by $f_{i,j}$ the value of the pixel whose coordinates are (i,j) $(i,j = -1,0,1)$. We then denote by f_{00}^k the estimated value of the faulty pixel as obtained by SC^k $(k = 1,2,3)$.

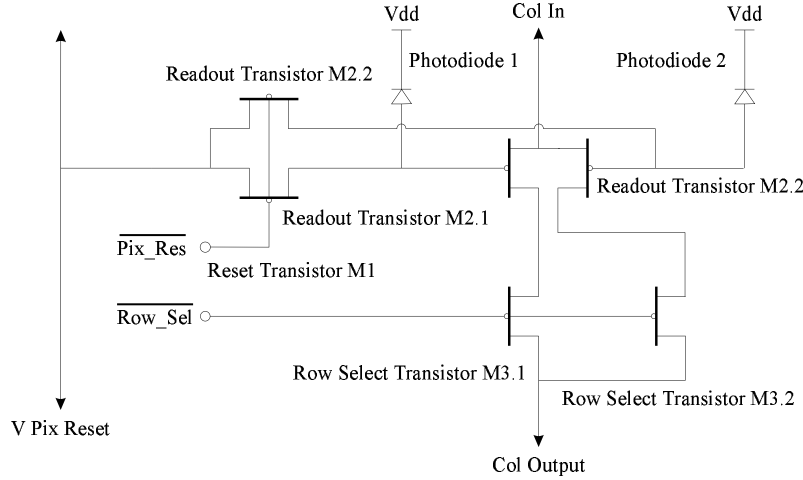

Figure 5. The further modified pixel cell.

Thus,

$$f^1_{00} = \frac{1}{8}\left(f_{01} + f_{11} + f_{10} + f_{1,-1} + f_{0,-1} + f_{-1,-1} + f_{-1,0} + f_{-1,1}\right)$$

and

$$f^2_{00} = \frac{1}{4}\left(f_{01} + f_{10} + f_{0,-1} + f_{-1,0}\right)$$

To obtain f^3_{00}, we assume that the faulty pixel and its eight neighbors obey a quadratic function

$$f^3_{x,y} = a_{00} + a_{10}x + a_{01}y + a_{11}xy + a_{20}x^2 + a_{02}y^2 + a_{21}x^2y + a_{12}xy^2$$

Substituting the given f_{ij}'s for $(i,j) \neq (0,0)$ we have eight linear equations in the eight unknown coefficients a_{kl}. Since $f^3_{00} = a_{00}$, we need to solve only for a_{00}, which results in

$$f^3_{00} = \frac{1}{2}\left(f_{01} + f_{10} + f_{0,-1} + f_{-1,0}\right) - \frac{1}{4}\left(f_{11} + f_{1,-1} + f_{-1,-1} + f_{-1,1}\right)$$

Note that all three estimates are linear combinations of the eight neighbors of the faulty pixel. In SC^2 and SC^3, the four immediate neighbors get higher weights than the other four. The effects of HC and SC on the image quality are analyzed in the next section.

5: Image quality analysis and numerical results

Both self-correction methods decrease to some extent the quality of the image obtained by the camera. The SC technique replaces the exact value by a linear combination of the neighboring pixels (which may or may not be close to the correct value). The HC method which multiplies the reading of half the pixel by 2 reduces the signal resolution by one bit. We next compare the image quality reduction in the original design (which enables only software correction) to that of the modified design (which attempts hardware correction first and software correction second).

We denote by N_{SC} and N_{HC}, the number of pixels that are corrected by SC and HC, respectively, and by $\overline{E}_{SC}$ and $\overline{E}_{HC}$, the average errors caused by these two methods, per image.

Denoting by QR the quality reduction of a corrected image, we define QR as the overall average error in pixel value. Clearly, the lower the value of QR, the better the design. QR can be obtained

by

$$QR = \frac{1}{M^2} \left(N_{SC} \, \overline{E}_{SC} + N_{HC} \, \overline{E}_{HC} \right) \tag{1}$$

where M^2 is the number of pixels per image.

Since the original design ($O.D.$) has only SCs and the modified design ($M.D.$) has mostly HCs and very few SCs, (1) becomes

$$QR(O.D.) = \frac{1}{M^2} N_{SC}(O.D.) \, \overline{E}_{SC} \tag{2}$$

and

$$QR(M.D.) = \frac{1}{M^2} \left(N_{SC}(M.D.) \, \overline{E}_{SC} + N_{HC}(M.D.) \, \overline{E}_{HC} \right) \tag{3}$$

We now need to obtain estimates for the parameters N_{SC}, N_{HC}, $\overline{E}_{SC}$, and $\overline{E}_{HC}$ for both designs. (Note that while N_{SC} and N_{HC} depend on the design, $\overline{E}_{SC}$ and $\overline{E}_{HC}$ do not.)

Denoting by $p = e^{-\lambda t}$ the probability of a pixel in the original design (or a half-pixel in the new design) to be fault-free at time t and by $q = 1 - p$ the probability of a pixel (or a half-pixel) failing by time t, N_{SC} and N_{HC} can be closely approximated (for small values of q) by

$$N_{SC}(O.D.) = p^8 q M^2$$

$$N_{SC}(M.D.) = (1 - q^2)^8 q^2 M^2$$

$$N_{HC}(M.D.) = 2pqM^2$$

and thus

$$QR(O.D.) = p^8 q \, \overline{E}_{SC} \tag{4}$$

and

$$QR(M.D.) = (1 - q^2)^8 q^2 \, \overline{E}_{SC} + 2pq \, \overline{E}_{HC} \tag{5}$$

For small values of q, q^2 is close to 0 and p is close to 1, and thus $QR(M.D.) < QR(O.D.)$ if and only if $2\overline{E}_{HC} < \overline{E}_{SC}$. Denoting by α the ratio $\overline{E}_{SC}/\overline{E}_{HC}$, the modified design has a better image quality than the original design if and only if $\alpha > 2$, or the average error due to software correction is at least twice that incurred by hardware correction.

The average error due to HC can be easily quantified. The estimate, denoted f_{00}^{HC}, is obtained by multiplying the value of half the pixel by two, and thus, its last bit may be incorrect. Therefore,

$$f_{00}^{HC} = \begin{cases} f_{00} & \textit{if the last bit of } f_{00} \textit{ is } 0 \\ f_{00} - 1 & \textit{if the last bit of } f_{00} \textit{ is } 1 \end{cases}$$

Denoting by E_{HC} the error caused by HC when correcting a single pixel, we have

$$E_{HC} = \begin{cases} 0 & \textit{if the last bit of } f_{00} \textit{ is } 0 \\ 1 & \textit{if the last bit of } f_{00} \textit{ is } 1 \end{cases}$$

Assuming that the last bit of the pixel's value is equally likely to be a 0 or a 1

$$\overline{E}_{HC} = 0.5$$

It is more difficult to calculate the errors caused by SC, since they depend on the correlations among neighboring pixels. We denote the error incurred in a single pixel due to using SC^k $(k = 1, 2, 3)$ by

$$E_{SC}^k = |f_{00} - f_{00}^k|$$

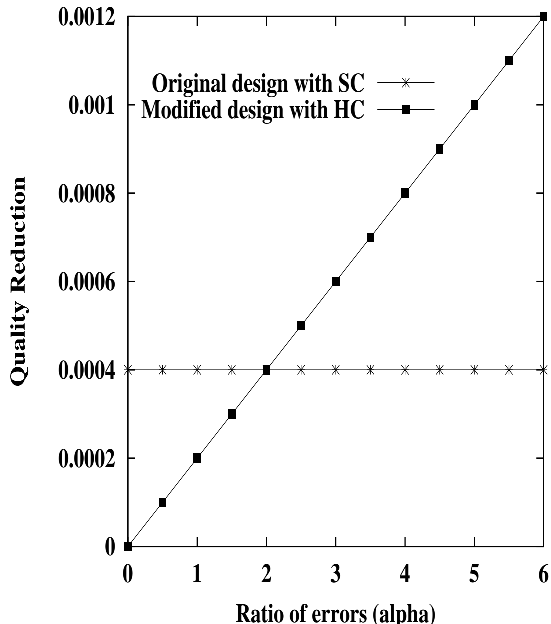

Figure 6. Quality reduction of the two designs as a function of the weight coefficient α ($\lambda = 2 \cdot 10^{-4}$, $t = 2$).

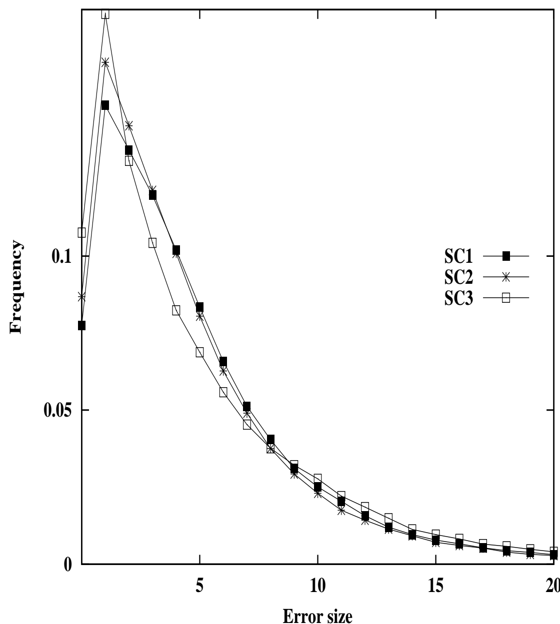

Figure 7. The distribution of the error size.

and the average error over the whole picture is denoted by $\overline{E}^k_{SC}$. $\overline{E}_{SC}$ can, at this point, only be calculated using simulation or by analyzing actual digital images. However, as our numerical experiments (reported next) show, it is very likely that $\overline{E}_{SC} > 1$. Thus, in most cases, $\alpha > 2$ and the modified design has a lower image quality reduction and should be preferred over the original design.

Figure 6 illustrates the calculation of QR for the two designs, as a function of the ratio α. As can be seen, the new design has a better image quality when $\alpha > 2$.

Since $\overline{E}_{SC}$ is very difficult to calculate analytically, we performed several experiments in which the average SC error was calculated for different pictures (all in gray scale) and the three SC methods. The pictures analyzed were divided into two main categories: portraits of people, and images of earth from space as taken by JPL (www.jpl.nasa.gov/radar/sircxsar).

The portraits had relatively low average SC errors, which varied between 2 and 6 (for a maximum pixel value of 255). The order of the errors was: $\overline{E}^3_{SC} < \overline{E}^2_{SC} < \overline{E}^1_{SC}$ which indicates that the four immediate neighbors should have a higher weight when determining the value of the center pixel. A similar conclusion is reached when observing the error size frequency distributions. One such distribution for a portrait, is depicted in Figure 7.

The results for images of landscapes were slightly different. The average SC errors tended to be much larger (between 10 and 20), and although in some cases SC^3 was better, there were some images for which SC^2 was best.

To illustrate the difference in image quality between the HC and SC methods we show in Figure 8 a small checkerboard with simulated faulty pixels (see part (a)). Figure 8(b) shows the image after HC, while Figures 8(c) and 8(d) show the same image after SC^1 and SC^3, respectively. Clearly, HC results in a much better corrected image than both SC methods. If SC is applied in addition to HC, an almost perfect image is obtained.

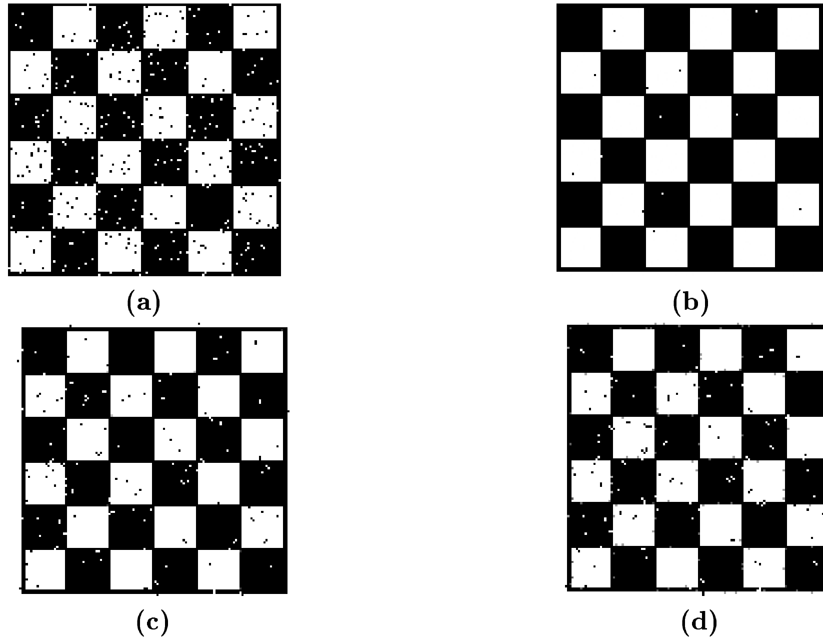

Figure 8. Comparing the HC to two SC schemes: (a) Uncorrected image, (b) Corrected with HC, (c) Corrected with SC^1, (d) Corrected with SC^3.

6: Conclusions

Hardware and software based fault tolerance techniques for an active pixel array have been presented in this paper. The hardware based method involved a redesign of the pixel cell so that the probability of a total pixel fault is greatly reduced. The hardware and software based methods have been analyzed, demonstrating that in most cases the combination of the hardware and software based methods results in a higher image quality than that of the software based technique alone.

References

[1] G. Chapman and Y. Audet, "Creating 35 mm Camera Active Pixel Sensors," *Proc. of the 1999 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 22-30, November 1999.

[2] EOS-30 review, sensor page, at www.dpreview.com, July 2000.

[3] I. Koren and Z. Koren, "Incorporating Fault-Tolerance into a Digital Camera-On-A-Chip," *Proc. of the 1999 Microelectronics Reliability and Qualification Workshop*, pp. 1-3, Pasadena, Oct. 1999.

[4] S-Y. Ma and L-G. Chen, "A single-Chip CMOS APS Camera with Direct Frame Difference Output," *IEEE Journal of Solid-state Circuits,* Vol. 34, Oct. 1999, pp. 1415-1418.

[5] B. Pain *et al.*, "A Low-Power Digital Camera-on-a-Chip Implemented in CMOS Active Pixel Approach," *Proc. of the 12th VLSI Design Conference*, pp. 1-6, January 1999.

[6] Siliconfilm at www.siliconfilm.com, July 2000.