A Self-Correcting Active Pixel Sensor Using Hardware and Software Correction

Glenn H. Chapman, Sunjaya Djaja, and Desmond Y.H. Cheung
Simon Fraser University

Yves Audet
Ecole Polytechnique, Montreal

Israel Koren and Zahava Koren
University of Massachusetts, Amherst

Editors' note:
Active pixel sensor (APS) CMOS technology reduces the cost and power consumption of digital imaging applications. This article presents a highly reliable system for the production of high-quality images in harsh environments. The system is based on a fault-tolerant architecture that effectively combines hardware redundancy in the APS cells and software correction techniques.
—Yervant Zorian, Virage Logic; and Dimitris Gizopoulos, University of Piraeus

As system-on-a-chip technology matures, including sensor arrays on the chip itself is increasingly valuable, allowing more system integration, higher operating speeds, and the ability to include many sensor types in a single substrate for hyperspectral image analysis At the same time an important trend in digital cameras has been the move to a larger detector area (currently reaching 35 mm); while shrinking the actual pixel size, both creating enhance resolution. This combination of digital imagers growing ever larger in silicon area and pixel count with shrinking pixel areas, results in increasing defects during fabrication and the number of dead pixels that develop over the device's lifetime. This makes it essential to avoid defects in these megapixel detectors. Furthermore, in remote, dangerous environments such as outer space, high radiation areas, and military battlefields, digital cameras can image the scene at low cost and low risk. However, these environments put more stress on the imager system (from radiation, heat, or pressure), possibly leading to pixel failure, while making the replacement of failed systems difficult. Thus, to increase fabrication yield and extend operational lifetimes for these sensor areas, manufacturers need to develop self-correcting, self-repairing imagers for both cameras and SoC systems.

Another recent trend in digital imager systems is the move from charge-coupled-device (CCD) detectors to CMOS-based active pixel sensors, which are easier to produce, cost less, use less power, and integrate easily with other processors.[1,2] Previously, we proposed an APS cell design that included redundancy, something that is not possible for CCDs, to enhance imager reliability.[3,4] This article extends that work by reporting on our implementation of the redundant-photodiode APS in a CMOS 0.18-micron process and our device testing in normal operating mode and in modes with various forms of defects. In addition to this hardware correction through redundant APS cells, we explore software correction techniques.[5-7] We have combined hardware correction with a new software correction algorithm to create an extremely reliable imaging system. To the best of our knowledge, such a combination has not previously appeared in the literature.

Redundant-pixel circuit

Our hardware correction mechanism consists of dividing a single pixel in half into two active subpixel circuits working in parallel. Figure 1 shows a schematic diagram of the two circuits, connected to achieve a pixel with redundancy. The light detection mechanism of one active pixel circuit works as follows: In normal operation, the incident light increases the reverse bias

0740-7475/04/$20.00 © 2004 IEEE Copublished by the IEEE CS and the IEEE CASS

current of photodiode PD.a. This current discharges the capacitance formed by the photodiode in parallel with the gate capacitance of readout transistor M2.a. The photosite node is precharged through reset transistor M1.a to a voltage level applied at line V (pixel reset) prior to the photocurrent's voltage integration. At the end of an integration period, a row select transistor, M3, is activated so as to deliver a current inversely proportional to the voltage built at the readout transistor's gate at line Column out.

Although the circuit in the diagram shows some duplication, in practice it does not require much increase in area. The photodiode area is much larger than the minimum size (typically 25 to 40% of cell area), and splitting it into two costs only a small percentage of cell area. Scaling the readout (M2.a and M2.b) transistors to half size keeps the circuit working like a full-size device, so the total area increase is low. Much of an APS cell's area is occupied by the row/column/power lines, which are not duplicated. Splitting the reset and row select transistors is optional but increases the pixel's defect tolerance. Figure 2 shows the layout of a redundant split photodiode APS in the CMOS 0.18-micron TSMC technology in which our test chips were manufactured.

The self-correcting scheme built into the APS counters defects affecting the pixel's photosite—the photodiode, output, and reset transistors. Figure 3 shows a cross-section of the important layers involved in the fabrication of one pixel in a CMOS 0.18-micron process. The N+ implant in the P sub (P substrate) forms the photodiode. This same N+ implant also acts as the source of reset transistor M1.

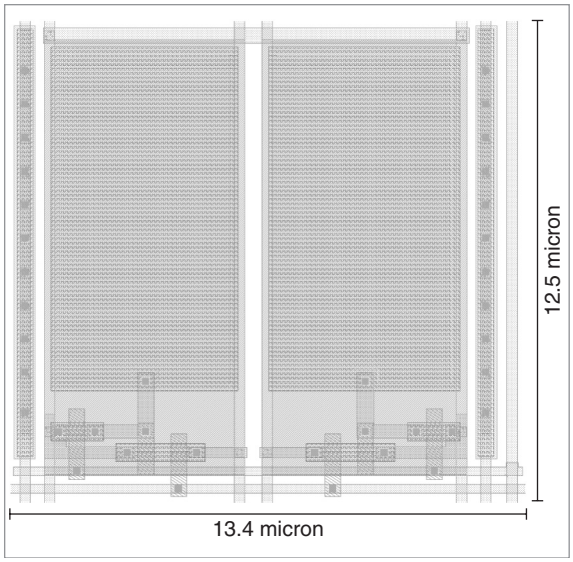


Figure 2. Layout of the split APS.

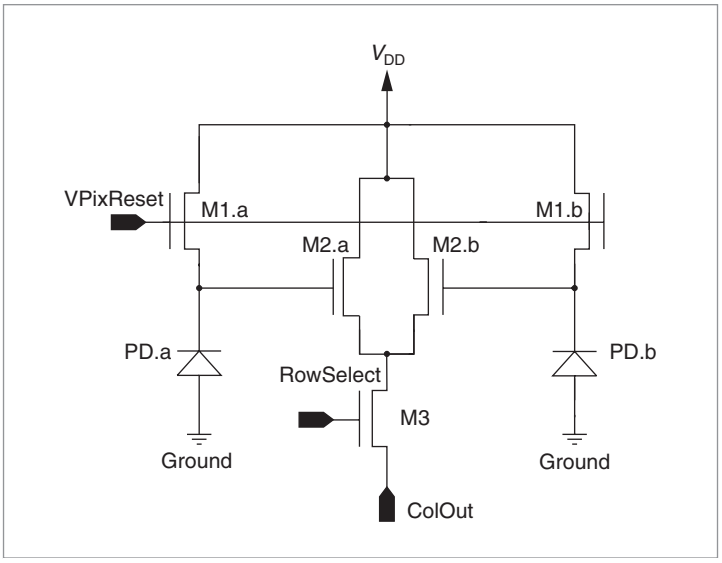


Figure 1. Schematic of redundant APS pixel. Two identical single-pixel circuits work in parallel, providing built-in redundancy for a robust APS array.

Readout transistor M2 is patterned separately and connected to the photodiode through a metal line. In this typical concept, the reset transistor pulls the gate high (near V_{DD}), and during operation, the photocurrent reduces output transistor M2's gate voltage. This keeps the APS in a linear operation region during low-illumination conditions. Hence, the total Col out current ranges from a high value for no illumination (with M2 fully on) to no current for saturated illumination (with M2 off).

Redundant-pixel self-correcting scheme

We categorize APS defects into three main classes on the basis of the final output signal, which we specify as the equivalent illumination the pixel would require to create that measurement:[4]

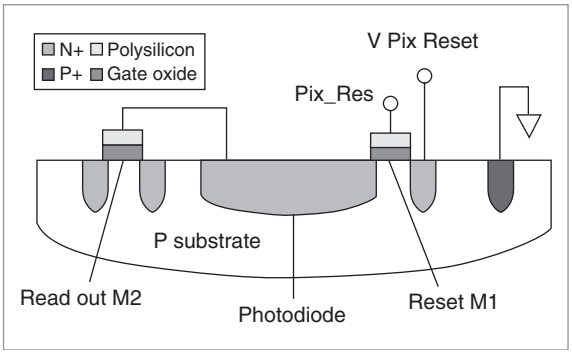


Figure 3. APS pixel fabrication layers.

- *Stuck high*. Optical signals saturate one pixel (for example, readout transistor gate shorted to ground, photodiode malfunction, reset path severed, or row select transistor not operating).
- *Stuck low*. An optical signal is absent on one pixel (for example, readout transistor gate shorted to V_{DD}, photodiode shorted to V_{DD}, photodiode fully covered by particles or other layer defects, or reset always on).
- *Low sensitivity*. An obstruction (a dust particle or layer defects, for example) partially blocks the photo-sensing element.

Stuck high refers to the optical signal's being high under all conditions. This means that output transistor M2's gate is electronically stuck low and always saturated. *Stuck low* refers to the optical signal's being absent under all conditions, causing the photodiode's cathode to be electrically stuck high.

Thus, there are five possible cases of faults affecting the two pixel halves:

1. Both halves of the pixel are active, indicating full-pixel sensitivity (0 to maximum current output range).
2. One half of the pixel is stuck high, leading to half-pixel sensitivity (0.0 to 0.5 output range).
3. One half of the pixel is stuck low, resulting in a half-pixel sensitivity biased by a half level of output (0.5 to 1.0 maximum current output range).
4. Both halves are stuck low—this is a dead pixel (output constantly near 0).
5. Both halves are stuck high, indicating a dead pixel (output constantly near 1); or half stuck high and half stuck low, also indicating a dead pixel (output constantly near 0.5). Pixel output is above zero for these, but it does not respond to illumination changes.

To calibrate the sensor, we can identify all these faults with two simple, standard tests. We take a dark-field image (from an 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 in case 1, half-stuck-low pixels in case 2 and fully dead pixels in case 4. Although data cannot be recovered from a dead pixel, the sensitivity adjustment calculations will take care of the first two cases. In the simplest analysis, a simple multiplication by 2 corrects these half-stuck cases.

We identify the half-stuck-high pixels in case 3 and the fully stuck-high pixels in case 5 from the dark field. In the fully stuck-high 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. Thus, multiplying by 2 results in the full signal.[3,4] These are calibration-related corrections best done after image processing, as is software correction.

Calibration tests can identify the half stuck pixels; a dark-field test shows the half stuck highs as half the maximum output swing plus an offset. A light-field test identifies half stuck lows by their half-maximum output swing. These tests are commonly performed at fabrication time. For calibration in the field, we commonly use the dark-field (no exposure) test; the light-field test might require taking special exposures. This scheme does not correct cases in which the row select or readout transistor is shorted. To achieve the self-correcting scheme, we must use a redundant pixel that sums the output currents with a current-to-voltage column amplifier.

Hardware correction experiments

A key aspect of the hardware correction scheme is that in the event of a failure, one subpixel behaves exactly like a full working pixel but generates half the signal and possibly some offset (in the stuck-high condition). After we remove the offset, this half-response characteristic will be the same whether the other subpixel of the pair has a stuck-high or stuck-low fault.[4] Thus, after analog-to-digital conversion of the pixel signal, a simple 1-bit left serial shift on the register containing the digital result brings the subpixel response to the level of a fully working pixel.

To experimentally test the redundant APS, we designed and manufactured a 1.5 × 1.5 mm test chip in CMOS 0.18-micron TSMC technology. The chip contains several layouts of APS arrays, with column and row decoders to extract the photocurrent from each pixel. To address each pixel, we controlled the decoders via a data acquisition board attached to a computer. We obtained measurements from the pixels with specially developed LabView-based software, which adapts to different acquisition schemes by adjusting the pixel array's reset and exposure times, and its reading sequence. During the measurement, we bring the pixel current off-chip and convert it to a voltage, using an operational amplifier connected as an inverter with a feedback resistor. We tie the Column out1 and Column out2 signals together for the measurement and then visualize the pixel reset and response signals, storing them with a digital oscilloscope.

To measure a pixel's response in the optically stuck-

low and stuck-high situations, we focus an argon laser operating in the visible region at 514 nm through a 50× objective lens, generating a narrow laser spot approximately 2.5 microns in diameter. A microscope mounted with a TV monitor directs the spot precisely on the photodiode surface. This spot enables the entire laser beam to fit within the photodiode surface area of one subpixel. We can then perform the optically stuck-low scenario by keeping one subpixel in the dark while shining the laser spot on the photodiode of the other subpixel and varying the laser beam intensity. From the same setup, we perform the optically stuck-high scenario by submitting one subpixel photodiode to intense laser spot exposure while illuminating both pixels with a uniform light source coming from the microscope that aligns the laser spot. The setup shows very little crosstalk among adjacent subpixels, even when we expose a subpixel to a high-intensity laser spot for the stuck-high scenario.

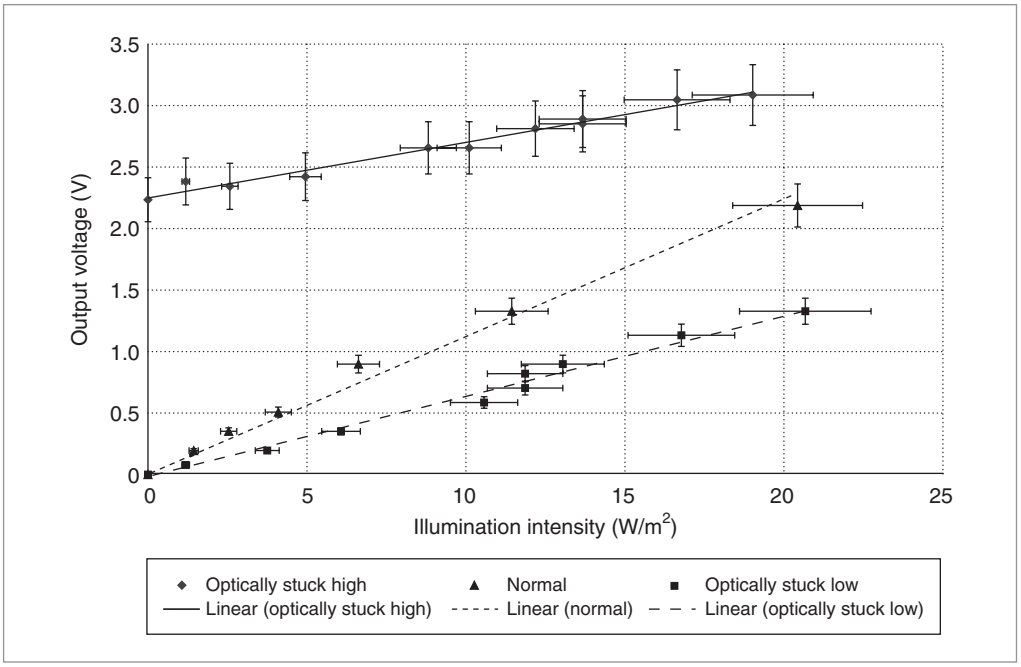


Figure 4. Subpixel response with power per unit area in three scenarios.

Figure 4 plots noninverted-output-voltage results after offset removal as a function of illumination intensity for three possible scenarios: a fully functional pixel (normal), one half stuck low, and one half stuck high. The error bars represent the combined uncertainties of the illumination intensity and the output voltage reading.

We evaluated pixel sensitivity with a linear-regression analysis. Table 1 presents the results. The stuck-low scenario yields a sensitivity of 0.571 ± 0.031 times that of the normal operating pixel, whereas the stuck-high scenario yields a sensitivity of 0.402 ± 0.031 times that of the normal operating pixel. The nonlinearity over the photodiode's full voltage swing is responsible for the deviation of the stuck-low and stuck-high responses from exactly 0.5 times a normal operating pixel's sensitivity.

To further investigate the redundant pixel's behavior, we measured the response of one subpixel working in normal, stuck-low, and stuck-high operations. Again, we extracted the response slopes from a linear-regression fit to measure subpixel sensitivity. Table 2 shows the results. The stuck-low scenario's sensitivity is higher than that of normal operation, agreeing with the results of Table 1 and showing a sensitivity gain of the subpixel in the stuck-low scenario. Conversely, the results show a reduced sensitivity of the subpixel in the stuck-high scenario, in agreement with the 0.402 sensitivity ratio obtained from the analysis shown in Table 1.

Table 1. Pixel sensitivity.

Scenario	Slope (mV-m²/W)	Error (mV-m²/W)
Normal operation	112	± 3.5
Stuck low	64	± 1.5
Stuck high	45	± 2.1

Table 2. Subpixel sensitivity.

Scenario	Slope (V/nW)	Error (V/nW)
Normal operation	1.39	± 0.043
Stuck low	1.59	± 0.004
Stuck high	1.12	± 0.005

Software correction method

When both subpixels are faulty, hardware correction is impossible, so we propose applying software correc-

tion after hardware correction of all pixels with one faulty subpixel. We suggest three software correction methods. The first method, SC[1], replaces a faulty pixel's value with the arithmetic mean of its eight neighbors. The second, SC[2], replaces the missing pixel's value with the arithmetic mean of its four immediate neighbors only. In the third method, SC[3], we fit a quadratic function to the nine pixels in question.

We use the following notation: We denote the faulty pixel's coordinates as (0, 0) and those of its eight neighbors by the pair (i, j), denoting the (row, column) of each neighbor pixel, listed counterclockwise. So these pairs are (0, 1), (1, 1), (1, 0), (1, –1), (0, –1), (–1, –1), (–1, 0), and (–1, 1). We denote the value of the pixel with coordinates (i, j) as $f_{i,j}$, where i and j can assume values of –1, 0, 1. We then denote the estimated value of the faulty pixel obtained by SC^k ($k = 1, 2, 3$) as f^k_{00}

Thus, for method SC[1],

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

and for method SC[2],

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

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

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

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

$$f^3_{00} = [(f_{01} + f_{10,} + f_{0-1} + f_{-10}) / 2] - [(f_{11} + f_{1-1,} + f_{-1-1} + f_{—11}) / 4]$$

All three estimates are linear combinations of the faulty pixel's eight neighbors. In SC[2] and SC[3], the four immediate neighbors get higher weights than the other four. We assume that the faulty pixel's eight neighbors are not faulty or at least hardware-corrected in the first correction step. The probability of two neighbors both having two faulty subpixels is very low (the defect must be very large, typically 10 to 20 microns, and aligned with pixels). In the rare case that this occurs, we omit the faulty neighbor from the average and use simple variations of formulas SC[1], SC[2], and SC[3].

Image quality analysis

Both self-correction methods somewhat decrease the quality of the camera's image. The software correction technique replaces the exact value by a linear combination of the neighboring pixels (which might or might not be close to the correct value). The hardware correction method, which multiplies the reading of half the pixel by 2, reduces the signal resolution by 1 bit. We next compare image quality reduction of the original nonredundant-pixel design, which enables only software correction, with that of our proposed modified design, which attempts hardware correction first and software correction second.

We denote the number of pixels corrected by hardware and software as N_{HC} and N_{SC}, and the average number of errors per image caused by these two methods as $\bar{E}_{HC}$ and $\bar{E}_{SC}$. 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. We obtain QR as follows:

$$QR = (N_{SC}\ \bar{E}_{SC} + N_{HC}\ \bar{E}_{HC}) / M^2$$

where M^2 is the number of pixels per image. Because the original design (OD) has only software correction, the equation becomes

$$QR_{OD} = (N_{SC}\ \bar{E}_{SC}) / M^2$$

and for the modified design (MD),

$$QR_{MD} = (N_{SC,MD}\ \bar{E}_{SC} + N_{HC,MD}\ \bar{E}_{HC}) / M^2$$

We must now obtain estimates for parameters N_{SC}, N_{HC}, $\bar{E}_{SC}$, and $\bar{E}_{HC}$ for both designs. (Note that N_{SC} and N_{HC} depend on the design, whereas $\bar{E}_{SC}$ and $\bar{E}_{HC}$ do not.) We denote as $p = e^{-\lambda t}$ the probability of a pixel in the original design (or a half-pixel in the modified design) being fault-free at time t; we denote as $q = 1 - p$ the probability of a pixel (or a half-pixel) failing by time t. We can closely approximate N_{SC} and N_{HC}, (for small values of q) by

$$N_{SC,OD} = p^8 q M^2$$
$$N_{SC,MD} = (1-q^2)^8 q^2 M^2$$
$$N_{HC,MD} = 2pqM^2$$

and thus

$$QR_{OD} = p^8 q\, \bar{E}_{HC}$$

and

$$QR_{MD} = (1 - q^2)^8 q^2 \bar{E}_{SC} + 2pq\,\bar{E}_{HC}$$

For small values of q, q^2 is close to 0 and p is close to 1, and thus $QR_{MD} < QR_{OD}$ if and only if $2\bar{E}_{HC} < \bar{E}_{SC}$. Denoting ratio $\bar{E}_{HC}/\bar{E}_{SC}$ by α, the new design has a better image quality than the original design if and only if $\alpha > 2$ or the average error caused by software correction is at least twice that caused by hardware correction.

We can easily quantify the average error due to hardware correction. We obtain the estimate of the pixel value, denoted f_{00}^{HC}, by multiplying the value of half the pixel by 2, and thus its last bit might be incorrect. Therefore,

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

Denoting the error caused by hardware correction of a single pixel as E_{HC}, we have

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

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

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

Quantifying the errors caused by software correction is more difficult because they depend on correlations among neighboring pixels. In the following analysis, we perform hardware correction first and then software correction on the pixels, with both subpixels faulty. Thus, we assume that all eight neighbors of a faulty pixel are either fault-free or hardware corrected.

We denote the error incurred in a single pixel from using SC^k ($k = 1, 2, 3$) as

$$\bar{E}^k_{SC} = |f_{00} - f^k{}_{00}|$$

Figure 5 illustrates the calculation of QR for the original and modified designs, as a function of ratio α. As the figure shows, the new design has better image quality when $\alpha > 2$.

Because $\bar{E}_{SC}$ is impossible to calculate analytically, we performed several experiments calculating the average software correction error for various pictures (all in gray scale) and the three SC methods. We analyzed two types of pictures: portraits of people and images of earth from space taken by the Jet Propulsion Laboratory (http://www.jpl.nasa.gov/radar/sircxsar).

The portraits had relatively low average software correction errors, which varied in value between 2 and 6 (for a maximum pixel value of 255). The order of the errors was $\bar{E}^3{}_{SC} < \bar{E}^2{}_{SC} < \bar{E}^1{}_{SC}$, indicating that the four immediate neighbors should have a higher weight in determining the center pixel's value. We reach a similar conclusion observing the error size frequency distributions. Figure 6 shows one such distribution for a portrait.

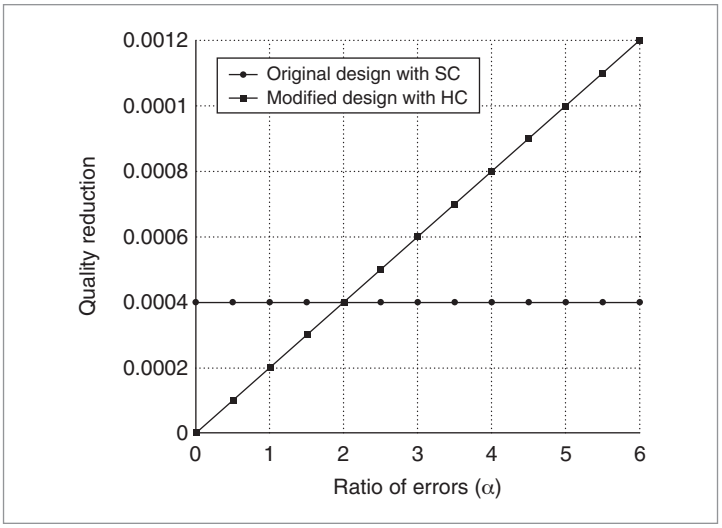


Figure 5. Quality reduction of the two designs as a function of weight coefficient α.

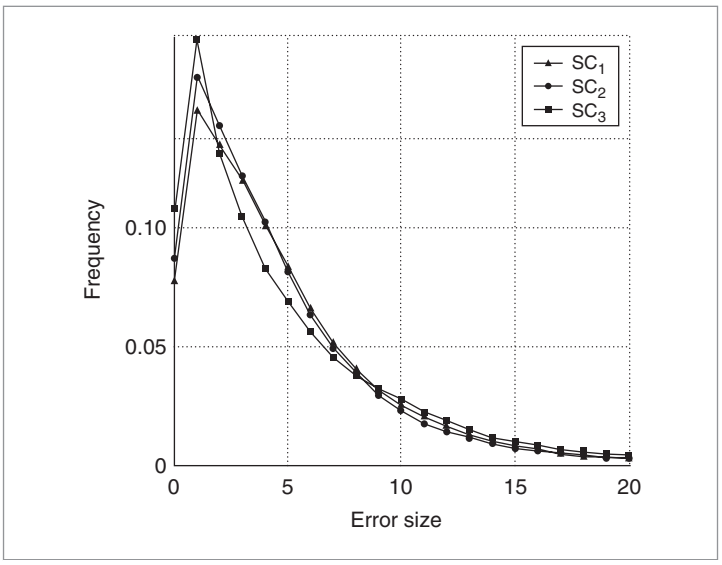


Figure 6. Error size distribution for a portrait.

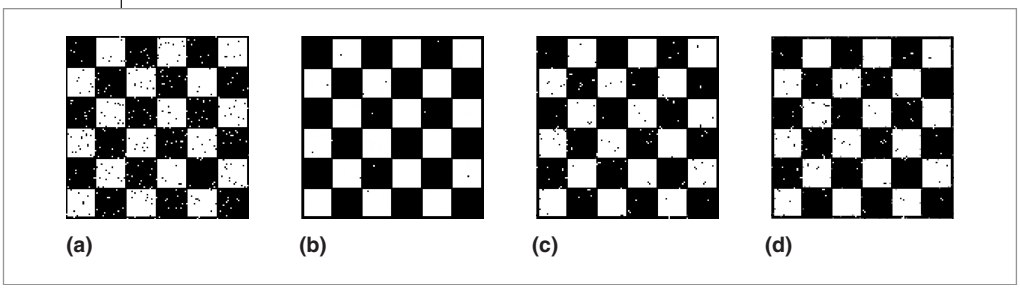


Figure 7. Hardware correction versus software correction: uncorrected image (a), hardware-corrected image (b), image corrected with SC^1 (c), and image corrected with SC^3 (d).

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

The previous results apply to the combination of performing hardware correction first and software correction second. To illustrate the difference in image quality between the methods performed separately, Figure 7a shows a checkerboard image with simulated faulty pixels. Figure 7b shows the image after hardware correction, and Figures 7c and 7d show the same image after SC^1 and SC^3. Clearly, hardware correction results in a much better corrected image than either SC method, correcting a very large percentage of faults. If we apply software correction in addition to the hardware correction of Figure 7b, it significantly reduces even the few remaining errors, and we obtain an almost perfect image (not shown here because to the naked eye it is indistinguishable from the error-free checkerboard).

The calibration using light- and dark-field illuminations discussed earlier identify the pixels with half sensitivities requiring hardware correction and the pixels requiring software correction. We easily make these corrections with the simple multiplication by 2 for hardware correction and the modest-complexity software correction interpolation formulas. Because the number of pixels with errors relative to the number of total pixels is likely to be small, the resulting system overhead for these corrections is small compared with other corrections (such as background subtraction to remove pattern noise) normally used for the entire pixel array.

THE IMAGER HARDWARE CORRECTION METHOD demonstrated in this article shows promising results for improving the yield of megapixel large-area-array APS. The redundant-pixel approach allows for defective-pixel avoidance, which inherently increases the imager yield and thus decreases the number of APS chips rejected after test. The technique employed to correct for defective pixels involves multiplication by a factor of nearly two, a calculation easily performed on chip after analog-to-digital conversion. This simplicity of the correcting scheme also enables the design of self-correcting APS for use in remote or harsh environments.

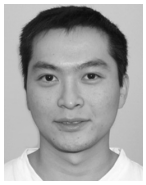
For greater defect density, combining the hardware correction technique with a software correction algorithm has been proven more effective than the hardware or software correction alone. The proposed software methods are also fairly simple and would be easily implementable in the processors typically used in combination with imagers for JPEG image compression. ■

References

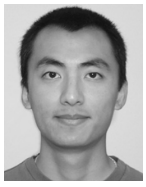
1. S.-Y. Ma and L.-G. Chen, "A Single-Chip CMOS APS Camera with Direct Frame Difference Output," *IEEE J. Solid-State Circuits* (JSSC 99), vol. 34, no. 10, Oct. 1999, pp. 1415-1418.
2. B. Pain et al., "A Low-Power Digital Camera-on-a-Chip Implemented in CMOS Active Pixel Approach," *Proc. 12th Int'l Conf. VLSI Design* (VLSI 99), IEEE Press, 1999, pp. 1-6.
3. G. Chapman and Y. Audet, "Creating 35 mm Camera Active Pixel Sensors," *Proc. 1999 Int'l Symp. Defect and Fault Tolerance in VLSI Systems* (DFT 99), IEEE Press, 1999, pp. 22-30.
4. Y. Audet and G.H. Chapman, "Design of a Self-Correcting Active Pixel Sensor," *Proc. 2001 Int'l Symp. Defect and Fault Tolerance in VLSI Systems* (DFT 01), IEEE Press, 2001, pp. 18-27.
5. I. Koren and Z. Koren, "Incorporating Fault Tolerance into a Digital Camera-on-a-Chip," *Proc. 1999 Microelectronics Reliability and Qualification Workshop*, Jet Propulsion Lab., 1999, pp. 1-3.
6. I. Koren, G. Chapman, and Z. Koren, "A Self-Correcting Active Pixel Camera," *Proc. 2000 Int'l Symp. Defect and Fault Tolerance in VLSI Systems* (DFT 2000), IEEE Press, 2000, pp. 56-64.
7. I. Koren, G. Chapman, and Z. Koren, "Advanced Fault-Tol-

erance Techniques for a Color Digital Camera-on-a-Chip," *Proc. 2001 Int'l Symp. Defect and Fault Tolerance in VLSI Systems* (DFT 01), IEEE Press, 2001, pp. 3-10.

Glenn H. Chapman is a professor in the School of Engineering Science, Simon Fraser University, British Columbia, Canada. His research interests include large-area laser-restructurable silicon systems, microfabrication technology, and micromachined sensors involving lasers. Chapman has a PhD in engineering physics from McMaster University, Ontario. He is a Senior Fellow of the British Columbia Advanced System Institute and a member of the IEEE.



Sunjaya Djaja is pursuing an MAS in electrical engineering at Simon Fraser University. His research interests include integrated image sensors; vision SoCs; digital, mixed-signal, and analog circuits; and algorithms for intelligent image sensing. Djaja has a BS in electrical engineering from Rensselaer Polytechnic Institute and a BSc in computer science from Simon Fraser University.



Desmond Y.H. Cheung is pursuing an MSc in electrical engineering at Simon Fraser University. His research interests include the design and implementation of several novel CMOS image sensors, analog and mixed-signal designs, and SoCs for digital photography. Cheung has a BSc in engineering science (computer option) from Simon Fraser University. He is a student member of the IEEE.

Yves Audet is an assistant professor in the Department of Electrical and Computer Engineering, Ecole Polytechnique, Montreal. His research interests include large-area sensors, imaging sensors, and optical interconnects for VLSI systems. Audet has a PhD in engineering science from Simon Fraser University.

Israel Koren is a professor of electrical and computer engineering at the University of Massachusetts, Amherst. His research interests include yield and reliability enhancement, fault-tolerant architectures, real-time systems, and computer arithmetic. Koren has a DSc in electrical engineering from the Technion—Israel Institute of Technology. He is an IEEE Fellow.

Zahava Koren is a senior research fellow in the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst. Her research interests include stochastic analysis of computer networks, IC yield, and computer system reliability. Koren has a DSc in operations research from the Technion—Israel Institute of Technology.

■ Direct questions and comments about this article to G.H. Chapman, School of Engineering Science, 8888 University Dr., Burnaby, B.C., Canada V5A 1S6; glennc@cs.sfu.ca.