

# MEMORY ERROR COMPENSATION TECHNIQUES FOR JPEG2000

*Yunus Emre and Chaitali Chakrabarti*

School of Electrical, Computer and Energy Engineering  
Arizona State University, Tempe, AZ 85287  
{yemre,chaitali}@asu.edu

## ABSTRACT

This paper presents novel algorithm-specific techniques to mitigate the effects of failures in SRAM memory caused by voltage scaling and random dopant fluctuations in scaled technologies. We focus on JPEG2000 as a case study. We propose three techniques that exploit the fact that the high frequency subband outputs of the discrete wavelet transform (DWT) have small dynamic range and so errors in the most significant bits can be identified and corrected easily. These schemes do not require any additional memory and have low circuit overhead. We also study several error control coding schemes that are effective in combating errors when the memory failure rates are low. We compare the PSNR versus compression rate performance of the proposed schemes for different memory failure rates. Simulation results show that for high bit error rates ( $10^{-2}$ ), the error control coding techniques are not effective and that the algorithm-specific techniques can improve the PSNR quality of up to 10dB higher compared to that of the no-correction case.

*Index Terms* - JPEG2000, SRAM, voltage scaling, error tolerance

## 1. INTRODUCTION

JPEG2000 is an image coding standard that is widely used in digital photography, high definition video transmission, video surveillance, medical imagery, etc [1]. It has better compression performance, higher computational complexity and larger memory requirement compared to JPEG. Although large sized memories can be accommodated on-chip, today these memories suffer from reliability related problems in scaled technologies. Thus data that is stored in the SRAM memory is not identical to the data that is read out for further processing.

SRAM failure rate is affected by several factors including process variations such as random dopant fluctuation (RDF), length, width and oxide thickness [2-3], soft errors (alpha particles) [4], temperature and aging [5]. RDF is the most dominant, and in scaled technologies, the standard deviation ( $\sigma$ ) of RDF is fairly large. For example, in 32 nm technology, standard deviation of threshold voltage variation is estimated to be around 40 mV to 60mV which is approximately 20% of the threshold voltage [6]. This results in error rates increasing from  $10^{-6}$  at nominal voltage of 0.9V to  $10^{-2}$  when voltage is dropped to 0.6V. Fortunately, image and video codecs have some degree of error tolerance and full correction is not needed to provide good quality.

Several circuit, architecture and system level techniques have been proposed to mitigate and/or compensate for memory failures. At the circuit level, different SRAM structures such as 7T, 8T and 10T have been designed to reduce failure rate [6]. Architecture level techniques have been proposed to avoid faulty bits using defect maps [3]. Row and column redundancy have been effectively used for low error rates [7]. In [8], memory capacity is traded-off for cache reliability. Many techniques make use of error control coding (ECC) [4], [9]. In [4], orthogonal latin square codes which achieve adaptive coding capability is used to trade-off cache size with correction capability. In [9], two dimensional coding (product code) is used to mitigate soft errors. ECC does not make any distinction between soft and hard errors and is effective for both types of failures.

More recently, algorithm-specific techniques have been developed for JPEG2000 [10] and MPEG [11]. These techniques mitigate system degradation due to memory failures using additional features that are intrinsic to the algorithm. In [10], binarization and second derivative of the image is used to detect error locations in different DWT sub-bands in JPEG2000. These are then corrected in an iterative fashion by flipping one bit at a time starting from the most significant bit (MSB). The overall procedure has fairly high latency and power overhead. The method in [11] stores the MSBs in a memory bank with 8T SRAM cells and the least significant bits (LSB) in memory banks with 6T cells. The 8T structure increases the area by almost 30% which also increases the leakage power consumption. The method in [12] operates the memory banks that store MSBs at a different voltage level than the ones that store LSBs. This is shown to achieve 45% power reduction with slight degradation in image quality.

In this work, we propose several techniques for combating memory errors in JPEG2000. Specifically, we describe an adaptive ECC scheme and three algorithm-specific techniques to compensate for the errors. The proposed ECC schemes are based on single error correction double error detection (SECDED) codes where the stronger codes are derived from the weaker codes to minimize the circuit and power overhead. These techniques are generic and can be used effectively for low error rates if the increase in SRAM size is not an issue. For instance, by using the (72, 64) code we can achieve high quality image if bit failure rate is above  $10^{-3}$  at the expense of 12.5% increase in SRAM size. Next, we propose three algorithm-specific methods with different levels of complexity that do not need additional SRAM memory. These methods exploit the fact that the DWT outputs have larger values for low frequency components and smaller values for high frequency components. Thus errors in MSBs of high frequency components can easily be identified and removed. The proposed methods allow us to operate at very high compression ratio (below 0.5 bpp) for memory bit error rates as high as  $10^{-2}$  with less than 2

---

This work was supported in part by a grant from CSR 0910699

dB drop in PSNR from the no-error case. The overhead of these methods is very small, making it possible for JPEG2000 implementations to be operated at very low voltages.

The rest of the paper is organized as follows. In Section 2, SRAM analysis under voltage scaling and other variations is presented. Section 3 gives a brief summary of JPEG2000. In Section 4, we describe the adaptive ECC methods followed by three algorithm specific methods in Section 5. Simulation and power estimation results for well-known test images are given in Section 6. The paper is concluded in Section 7.

## 2. SRAM FAILURE AND POWER ANALYSIS

The errors in SRAM are categorized into two groups: hard errors which are mainly due to RDF and soft errors which are primarily due to alpha particles. Soft error rate stays almost constant or slightly reduces as transistor sizes scale down; however the number of errors due this effect increases since the probability of hitting multiple transistors by an alpha particle gets larger [4]. Although the number of soft errors increases exponentially as voltage is scaled down, the overall failure rate in memories is still dominated by RDF [6]. Inter-die variation due to RDF is typically modeled using additive iid Gaussian distributed threshold voltage with zero mean and standard deviation ( $\sigma_{VT}$ ) which greatly depends on manufacturing and transistor sizes [6].

There are three main factors that contribute to overall SRAM failure rate: read upset, write failure and access failure.  $P(\text{overall}) = U\{P(\text{read upset}), P(\text{read access}), P(\text{write})\}$ . Read upset occurs during read cycles because of unbalanced voltage sharing at the read node. Write and read access failures are due to high drop or increase in the read and write current. In [11], read failure rate of 6T SRAM structure at nominal (typical) corner for 65nm technology is shown to differ from  $10^{-8}$  to  $10^{-3}$  when voltage is dropped from nominal value of 1V to 0.6V. Moreover, at scaled voltage levels, write failure rate is evaluated to be  $10^{-3}$  for the same technology.

We studied the three types of failure using RDF model for 32nm technology using Hspice with high performance predictive technology models (PTM) from [13]. We did Monte Carlo simulations to calculate the overall bit error rate for different levels of RDF. Figure 2 plots the BER as a function of operating voltage for three different values of  $\sigma_{VT}$ . The overall error probability is dominated by one of the failures. For instance, at nominal voltage, read upset dominates the failure whereas read access failure is more effective at medium voltage levels and write access failure dominates at very low voltage levels. In 32nm technology, the BER is around  $10^{-6}$  at nominal voltage of 0.9V and increases to  $10^{-4}$  at 0.75V to  $10^{-3}$  at 0.68V and to  $10^{-2}$  at around 0.6V. Although the results presented here show only hard failure rates, the error compensation methods described in the following sections are applicable for both hard and soft errors.

Figure 3 shows normalized power consumption as a function of the operating voltage for 32 nm SRAM core (without read-write circuitry) obtained using Hspice and PTM models. We see that dynamic power consumption can be lowered by more than 2 times when the supply voltage is reduced only by 0.15V. Under aggressive voltage scaling (operating voltage=0.6V), the power saving in write cycle is 91% and that of read cycle is 79% at the cost of higher error rates (BER= $10^{-2}$ ). Leakage power also drops by more than 80% with aggressive voltage scaling. In Section 6, we show that most of the significant errors can be compensated and satisfactory image quality can be achieved using algorithmic

techniques even for such high failure rates. This enables all components of the JPEG2000 accelerator (datapath and memory) to be operated at low voltages, thereby reducing the overall system power consumption significantly.

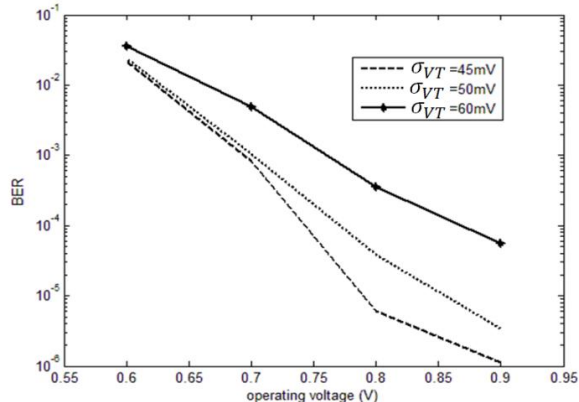


Figure 2: SRAM failure probability in 32nm technology for different voltage and RDF levels

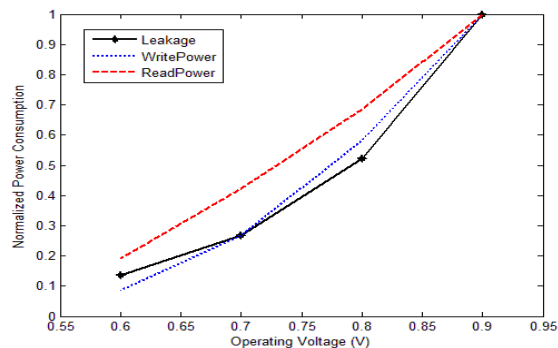


Figure 3: Leakage and dynamic power consumption (normalized) of SRAM cells in 32 nm technology for different voltage levels

## 3. JPEG2000 SUMMARY

JPEG2000 is one of the widely used still image compression techniques. It supports progressive image and region of interest coding and has better visual quality and compression performance compared to JPEG. In fact, it achieves more than 2 dB improvement in PSNR for the same compression rate compared to JPEG [1].

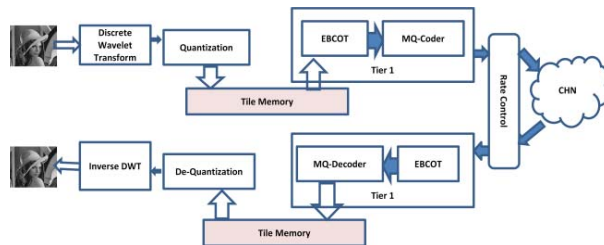


Figure 4: Block Diagram of JPEG2000

The general block diagram of the JPEG2000 encoder/decoder is illustrated in Figure 4. The original image (in pixel domain) is

transformed into frequency sub-bands using the DWT engine followed by quantization and stored into tile memory. Compressed image is obtained after EBCOT processing followed by rate control. While rate control can be achieved by adjusting quantization level and by EBCOT, EBCOT provides gradual improvement (unlike quantization) and is more favorable.

The 2 level sub-band representation of the DWT output is shown in Figure 5-a. The input image of size  $N \times N$  is processed by high-pass (H) and low-pass (L) filters along rows and columns followed by subsampling to generate the first level outputs of size  $N/2 \times N/2$ : HL1 (high-pass along row followed by low-pass along columns), HH1 (high-pass along rows followed by high-pass along columns), LH1 (low-pass along rows followed by high-pass along columns) and LL1 (low-pass along rows followed by low pass along columns). The LL1 outputs are further processed by high-pass and low-pass filters along the rows and columns to generate the second level outputs, HH2, HL2, LH2, and LL2, each of which is of size  $N/4 \times N/4$ . For a natural image, the low frequency sub-bands are likely to contain coefficients with larger values while higher frequency sub-bands contain coefficients with small values. Therefore the most significant bits of high frequency sub-bands (such as HL1, LH1, HH1) typically consist of all zeros. We exploit this fact in developing the memory error compensation techniques.

The DWT outputs are processed by Tier-1 coding which exploits the redundancy of these outputs in different bitplanes. Tier-1 coding in JPEG2000 takes  $32 \times 32$  or  $64 \times 64$  bits from each bitplane and encodes them independently. This stage has two main sections: EBCOT and MQ-Coder. EBCOT uses context based coding for each bit in a given bit plane, MQ-Coder then calculates its probability and determines the final compressed bit stream. For all-zero bit planes, Tier-1 coding skips processing those bit plane and only adds one bit information to the header of the file. Figure 5-b and 5-c illustrates the bit plane representation and independent code blocks ( $32 \times 32$  or  $64 \times 64$ ) of an image.

In most implementations, DWT coefficients are stored in a memory as shown in Figure 4 [14]. In general, the tile memory is typically a SRAM because of its low latency. In scaled technologies, there may be errors in these memories as described in Section 2. As a result the DWT output that is stored in the tile memory is different from the data that is read out of the memory for EBCOT processing. In Section 4 and 5, we describe techniques to compensate for these errors.

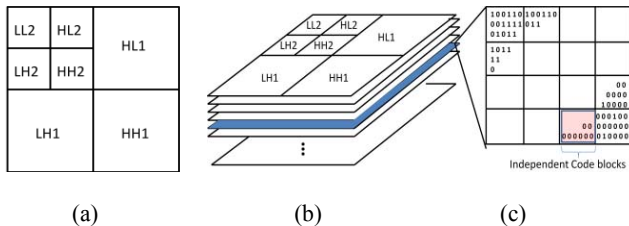


Figure 5: a) Sub-band representation of DWT output b) Bit plane representation of Sub-band c) Code block representation of bit planes

#### 4. ADAPTIVE ECC

In this section, we study the use of ECC in combating errors in memories. We propose to use adaptive SECDED schemes where the stronger codes can be derived from weaker but longer codes. We study 3 different SECDED codes: (72, 64), (39, 32), and (22, 16). (22,16) is the strongest code with an area increase of 37.5%

followed by (39, 32) with area increase of 21.9%, and (72, 64) with area increase of 12.5%. We show the BER performance of these codes in Section 6.

An important characteristic of these codes is that the parity generator matrix for shorter code (stronger) can be derived from the parity generator matrix of the longer code (weaker). This can be utilized to design hardware that can be shared for multiple codes. Consider the parity generator matrix of the (72, 64) code illustrated in Figure 6-a. It consists of 8 rows (equal to number of parity bits). The first half of this code (column 1 to 32) except the seventh row can be used to generate the parity matrix of (39, 32) code since the seventh row consists of all zeros. Similarly, the parity matrix of (22, 16) can be derived from the matrix of (39, 32) by taking the first 16 columns and dropping the all zero row (see Figure 6-b). Although we need additional circuitry compared to SECDED implementations which are optimized for a single code, generating codes in this manner allows us to adjust coding strength as required.

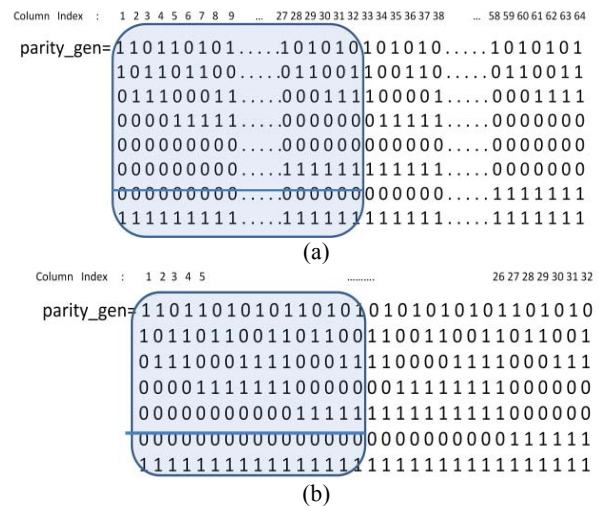


Figure 6: a) Parity generation of (39, 32) code from (72, 64) code, b) Parity generation of (22, 16) code from (39, 32) code

### 5. ALGORITHM-SPECIFIC TECHNIQUES

In this section, we propose JPEG2000-specific techniques to mitigate the impact of memory errors on system quality. Unlike other circuit level and ECC methods, these schemes introduce little circuit overhead and no additional data storage. For high frequency sub-bands such as HL, LH and HH, we propose three methods that exploit the fact that, for a natural image, these outputs typically consist of smaller values and thus contain small number of non-zero bits in MSB planes. Unfortunately this fact cannot be exploited for LL2 sub-band and a low pass filtering technique is used to mitigate errors there.

#### 5.1. Method 1

In this method, we erase all the data (ones) in the bit planes that are higher than a certain level for high sub-bands. Through our simulations, we found that for 16 bit data very little information is lost by discarding 2 to 4 MSB planes. Furthermore, since EBCOT skips coding bit planes that consist of all-zero bits, this technique helps reduce the power consumption due to EBCOT.

## 5.2 Method 2

Although Method 1 is very simple, there can be false erasures in MSB planes resulting in loss of quality. Method-2 addresses this problem by exploiting the image statistics. Here, the number of ones in a given bit plane is counted and if the number is below a certain threshold, all the bits in that plane are erased and the all-zero bit plane information is passed to EBCOT. If the number exceeds the threshold, the counter is disabled and all the bits are coded by EBCOT. The threshold value here is dynamic and is calculated using BER and number of bits in the bit-plane. We set the threshold value to be equal to twice the expected number of errors in a bit plane to reduce performance degradation due to false detection probability. For instance, the threshold value for HH1 sub-band of size  $128 \times 128$  when  $BER=10^{-3}$  is given by  $2 \times 128 \times 128 \times 10^{-3} \approx 32$ , which is also the upper limit of the counter.

The overhead of this method is the counter. Fortunately, it is not triggered often since it operates only on the high bit planes. Furthermore, it is disabled after it identifies the first bit plane that is not erased.

## 5.3. Method 3

Discarding all the bits in a given bit plane when a threshold condition is satisfied may sometimes result in losing valuable MSBs. We propose a third method which looks at data in current bit plane and also data in one upper and two lower bit planes. This is motivated by the fact that bits in a given bit plane are correlated with bits in their neighboring bit planes. First, the counter and dynamic threshold based scheme is used to decide whether to process the current bit plane. Next, the non-zero bit in the current bit plane is processed by considering a  $3 \times 3 \times 4$  neighborhood of bits as illustrated in Figure 7. If a non-zero bit is detected in  $(k, l)$  position of the  $i^{th}$  plane, it checks the  $3 \times 3$  block of bits around the  $(k, l)$  position in the  $(i + 1)^{th}$ ,  $(i - 1)^{th}$  and  $(i - 2)^{th}$  planes in addition to its 8 neighbors in the  $i^{th}$  plane. If it detects another 1 within this search group, it decides that the current bit is a correct 1. Similar to method 2, it stops after identifying the first bit-plane that is not eligible for erasure.

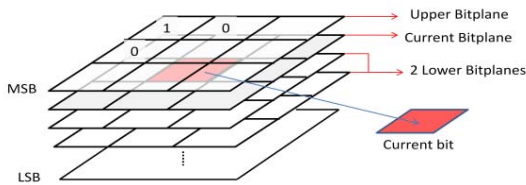


Figure 7: Bit planes used in error detection mechanism

## 5.4. Correction Method for LL2 Sub band

For the LL2 sub-band, the errors can be corrected by simple filtering. For each DWT output, a  $3 \times 3$  block of its neighbors is used to calculate its expected value. If the current value differs from the mean by  $\Delta$ , the original value is kept, otherwise it is updated by the mean value. In order to reduce the amount of computation, we only consider 8 MSB of a 16 bit output to calculate the mean. Through our simulations we found that for this scenario  $\Delta = 16$  gives very good results.

## 6. SIMULATION RESULTS

### 6.1 Quality Performance

In order to evaluate the performance of the different methods, we implemented (9, 7) lossy JPEG2000 with 2 level DWT and  $32 \times 32$  EBCOT block size. We studied the quality vs. compression rate characteristics. Quality is measured with respect to the reconstructed image using peak signal to noise ratio (PSNR) and the compression rate is calculated in terms of bits per pixel (bpp) based on the final compressed image size. Two representative images,  $256 \times 256$  Lena and Cameraman, are used in the simulations. Matlab is used to compute the performance curves. The overall BER rate in the tile memory is changed from  $10^{-4}$  to  $10^{-2}$  which is compatible with the BER results obtained for 6T SRAM under voltage scaling in Section 2.

#### 6.1.1. Case 1: $BER=10^{-4}$ :

Figure 8 illustrates the system performance using Lena image when  $BER=10^{-4}$ . If error-free performance is desired, any of the ECC methods can be used; so we choose the weakest ECC, that is (72, 64), to minimize power consumption. If some amount of image quality degradation is tolerable (below 0.5dB for 0.5bpp), the no-correction method is sufficient and there is no benefit of using any of the algorithm-specific methods.

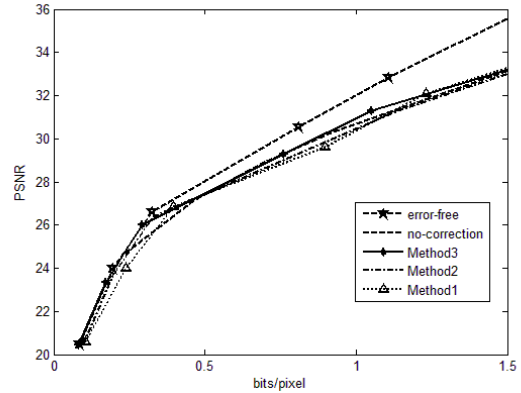


Figure 8 Quality vs. Compression rates when  $BER=10^{-4}$  for Lena

#### 6.1.2. Case 2: $BER=10^{-3}$ :

The performance curves show a different trend when BER increases to  $10^{-3}$ , as shown in Figure 9 for Lena and Cameraman images. Of the ECCs, (22, 16) code provides error-free performance for all compression rates. The (72, 64) code and (39, 32) code provide fairly good results and also are superior to other algorithms especially for high compression rates. Thus for very high quality images, adaptive ECC could be chosen when the increase in memory size and power consumption is acceptable.

Algorithm-specific methods can provide good results for low and medium quality/ rate regions. Method-3 follows the error-free curve very closely in this range (0.2 to 1bpp); it satisfies the desired quality with slight increase in power consumption. For example, it improves quality almost by 3 dB at 0.75bpp rate compared to the no-correction case. If some degradation is tolerable, Methods 1 and 2 are good candidates when compression rate is above 0.8bpp, since they have lower complexity compared

to Method 3. For very low compression rates ( $<0.25$ bpp), the performance of the no-correction case is just as good as the others.

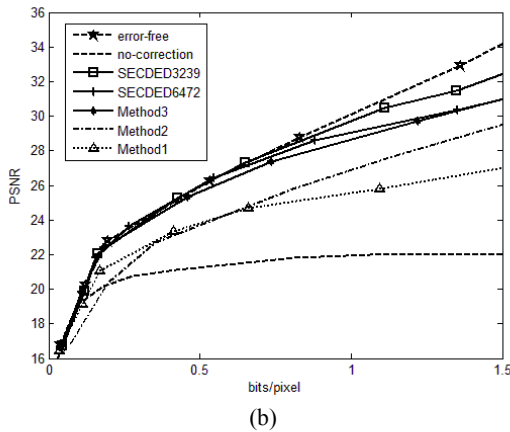
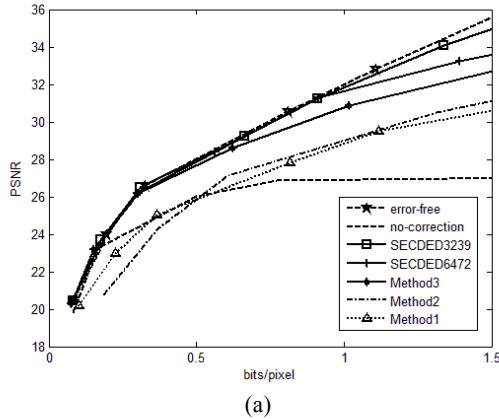


Figure 9 Quality vs. Compression rates when  $BER=10^{-3}$  for a) for Lena b) for Cameraman

### 6.1.3. Case 3: $BER=10^{-2}$ :

Figure 10 illustrates the performance when  $BER=10^{-2}$ . The benefit of ECC is not as high as it was for  $BER=10^{-3}$  or  $10^{-4}$ . For instance, for 0.5 bpp, improvement due to (72, 64) code is around 2 dB compared to the no-correction case. Figure 10-b illustrates the trade-off between coding strength and performance of all ECCs. The performance improvement due to ECC is small and does not compensate for the large overhead here.

Algorithmic methods improve the quality 3 to 8 dB for compression rate around 0.5 bpp as shown in Figure 10a. Although Method-1 and 2 do not improve performance much for low rates, they can provide acceptable performance with much lower overhead. Method 3 follows the no-error curve closely under 0.25 bpp compression rate. For the mid-quality range, it improves the performance noticeably and achieves the best quality among all the ECC and algorithm-specific techniques. For instance, it achieves 1.5dB better quality than (22, 16) code without any memory overhead.

### 6.1.4 Unequal Error Distribution

We also considered unequal error distribution where the low subbands had lower error rates compared to the high subbands. For the case when  $BER=10^{-4}$  for low subbands (LL2) and  $BER=10^{-3}$  for high subbands, we found that when same number of bits is used

to represent all coefficients (16 bits in this case), an error in MSB of high subbands degrades the performance in PSNR at comparable levels to an error in MSB at low subbands, even though their effects on visual quality are different.

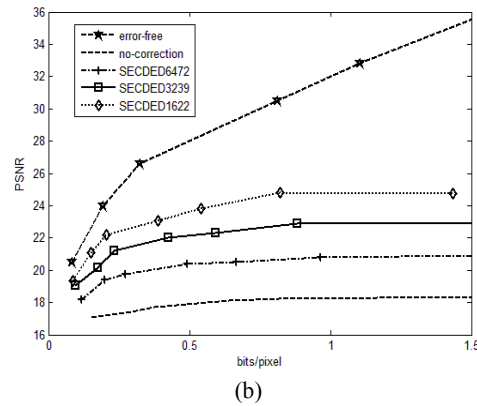
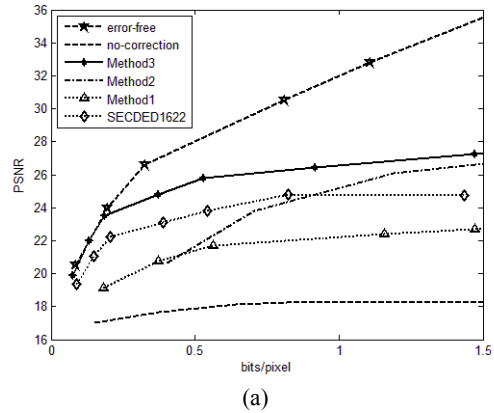


Figure 10 Quality vs. Compression rates for Lena when  $BER=10^{-2}$  a) using algorithmic techniques b) using ECC techniques

## 6.2. Overhead: Area, Delay, Power Consumption:

### 6.2.1. ECC Method

*Circuitry:* We combine the circuitry of multiple ECCs to achieve adaptive coding capability with lesser circuitry. The encoder for (72, 64) and (39, 32) is illustrated in Figure 12. For (72, 64), the input bits  $b_1$  through  $b_{32}$  are sent to one parity generator and bits  $b_{33}$  through  $b_{64}$  are sent to the second parity generator. The combiner combines the two sets of parity bits and generates parity bits for the (72, 64) code. When higher coding capability is required, as in (39, 32), the second parity generator and combiner (shaded blocks in Figure 12) are disabled and the outputs of the first generator are output. The decoder can be implemented using a similar hierarchical structure.

The power consumption and latency of encoder/decoder pair of ECCs are obtained using Design Compiler and 45nm models from [15]. Table 1 lists the area, power consumption, and latency results for the (72, 64) code and (39, 32) code. The clock period is 1.25ns. Note that for large memory, power consumption due to encoding and decoding can be significant.

*Memory Core Power Saving:* In addition to circuitry overhead, ECC techniques require extra memory to store redundant data (parity sequence) which increases the leakage power of the SRAM core. Therefore, leakage power saving of the ECC-based method is lower. For instance, if the SRAM core is operated at 0.7V, the leakage power saving could be as high as 75% (see Fig 3). However if (39, 32) code is used, the additional memory causes the leakage energy reduction to be 69%.

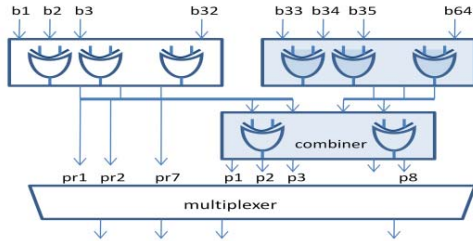


Figure 12: Block diagram of encoder for (72, 64) and (39, 32) codes

	Encoder (72,64)/(39,32)	Decoder (72,64)/(39,32)
Area (# of slices)	313.95	575.49
Worst-case delay (ps)	390 / 270	1142 / 610
Leakage Power (uW)	3.12	5.07
Active Power (uW)	230.30 / 93.38	347.18 / 155.42

Table 1: Power consumption, latency and area overhead of ECC scheme for (72, 64) code and (39, 32) code

### 6.2.2. Algorithm-Specific Methods

*Circuitry:* The simplest scheme, Method-1, requires no additional circuitry; therefore it does not add any overhead. Both Methods-2 and 3 require a counter circuitry. In addition, Method-3 requires an all-zero detector for comparing with the neighboring bit values.

As mentioned in Section 5, the counter circuit is not triggered much. For instance, for images sizes of sizes 256x256 with BER of  $10^{-4}$ ,  $10^{-3}$  and  $10^{-2}$ , dynamic threshold value for the counter ranges between 3 and 320. Therefore, in order to support a 256x256 image with BER= $10^{-2}$ , we need a 9 bit counter.

The power consumption and latency of the 9-bit counter used in Methods 2 and 3 and the all-zero detector used in Method 3 are obtained using Design Compiler and 45nm models from [15]. Table 2 illustrates the power consumption, area and latency results for clock period of 1.25ns.

	9-bit Counter	All-zero detector
Area (# of slices)	90.81	22.39
Worst-case delay (ps)	292	80
Leakage Power (uW)	1.01	0.081
Active Power (uW)	31.77	0.78

Table 2: Power consumption, latency and area overhead of circuitry in Methods 2 and 3

*Memory Core Power Saving:* Unlike ECC based techniques, these methods do not require any additional memory and can

therefore achieve full power reduction due to voltage scaling. For instance, if Method 3 is invoked to compensate BER= $10^{-2}$  obtained by operating at 0.6V, the leakage energy of the SRAM core would drop by 82% drop and the active energy would drop by 85%.

## 7. CONCLUSION

In this paper, we presented use of generic ECC based techniques and algorithm-specific techniques to mitigate memory failures due to both technology and voltage scaling in JPEG2000 implementations. The proposed ECC techniques can be used for low memory error rates but have additional memory overhead. The three algorithm-specific techniques provide good output quality at high failure rates. For instance, for BER= $10^{-3}$  at 0.5 bpp, Method 3 results in 8 dB increase in PSNR compared to no-correction case and 2 dB drop compared to no-error case. Overall, these methods have no memory overhead and cause only a slight increase in system complexity. They enable us to drop the operating voltage of memory resulting in significant reduction in system-level power consumption with little reduction in image quality for low to medium compression rates.

## 8. REFERENCES

- [1] P. Schelkens, A. Skodras, and T. Ebrahimi, *The JPEG 2000 Suite*, Wiley, 2009.
- [2] S. Mukhopadhyay, et. al, "Modeling of Failure Probability and Statistical Design of SRAM Array for Yield Enhancement in Nanoscaled CMOS". IEEE Trans. on CAD of Integrated Circuits and Systems, vol. 24, no 12, pp. 1859-1880, Dec. 2005.
- [3] A. Agarwal, et al., "Process Variation in Embedded Memories: Failure Analysis and Variation Aware Architecture", IEEE Journal of Solid-State Circuits, vol 40, no 9, pp. 1804-1813, Sept. 2005.
- [4] Z. Chishti, et al., "Improving Cache Lifetime Reliability at Ultra-low Voltages", Micro'09, pp. 89-99, Dec. 2009.
- [5] B. P. Wong, et. al., "Nano-CMOS Circuits and Physical Design", First Edition, Wiley-Interscience, 2004.
- [6] K. Zhang, "Embedded Memories for Nanoscale VLSIs", Springer 2009.
- [7] S. E. Schuster, "Multiple word/bit line redundancy for semiconductor memories," *IEEE J. Solid-State Circuits*, vol. SC-13, pp. 698-703, Oct. 1978.
- [8] C. Wilkerson, et al., "Trading off Cache Capacity for Reliability to Enable Low Voltage Operation," Proc. 35<sup>th</sup> International Symposium on Computer Architecture (ISCA-35), pp. 203-214, June 2008.
- [9] J. Kim, et. al, "Multi-bit Tolerant Caches Using Two Dimensional Error Coding", Micro-40, pp 197-209, 2007.
- [10] M. A. Makhzan, et. al, "A low Power JPEG2000 Encoder with Iterative and Fault Tolerant Error Concealment", IEEE Trans. on VLSI Systems, vol 17, no 6, pp. 827-837, June 2009.
- [11] I. J. Chang, et. al., "A Voltage-Scalable & Process Variation Resilient Hybrid SRAM Architecture for MPEG-4 Video Processors", DAC, pp 670-675, 2009.
- [12] M. Cho, et. al, "Accuracy-aware SRAM: a reconfigurable low power SRAM architecture for mobile multimedia applications", Asia and South Pacific Design Automation Conference, pp. 823-828, 2009.
- [13] <http://ptm.asu.edu/>
- [14] Analog Devices Inc., Norwood, MA, "ADV202: JPEG 2000 Video Codec", <http://www.analog.com/>
- [15] Nangate, Sunnyvale, California, "45nm Open Cell Library", <http://www.nangate.com/>