DATA-PATH AND MEMORY ERROR COMPENSATION TECHNIQUE FOR LOW POWER JPEG IMPLEMENTATION

Yunus Emre and Chaitali Chakrabarti

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

ABSTRACT

This paper presents a novel technique to mitigate effects of data-path and memory errors in JPEG implementations. These errors are mainly caused by voltage scaling and process variation in scaled technologies. We characterize the data-path and memory errors and derive a probability distribution of the total number of errors. We propose an algorithm-specific technique that corrects most errors after the quantization step in JPEG encoder. It is based on exploiting the characteristics of the quantized coefficients and achieves high performance with small circuit overhead. Simulation results show that the proposed technique has a PSNR performance degradation of around 1.5 dB compared to the error-free case and 4 dB improvement compared to the no correction case at compression rate of 0.75 bpp when $BER = 10^{-4}$.

***Keywords*:** JPEG, Error Compensation, Voltage Scaling

1. INTRODUCTION

JPEG is one of the most widely used image compression standards today. It has slightly lower compression performance compared to JPEG2000, but because of its simple structure and ease of implementation, it is still very popular. JPEG is part of many embedded devices for multimedia where power consumption is a very important metric. Lowering the supply voltage is an effective way of reducing the power consumption of these devices. However, in scaled technology nodes, this exacerbates process variations which could result in critical path violations leading to failures [1]. Thus JPEG architectures in scaled technologies will have to achieve a balance between voltage scaling to lower power consumption and image quality.

Several JPEG architectures have been proposed that trade-off power consumption and quality. They primarily focus on discrete cosine transform (DCT) which is one of the high power consuming units [2-5]. The DCT architecture in [2] exploits correlation between DCT coefficients in conjunction with standard techniques such as voltage scaling, data parallelism and pipelining. Data bit-width adaptation is exploited in [3] to reduce the number of computations primarily for high frequency components. Process variations effects are considered in [4] which generates more important DCT coefficients first and uses longer delay paths for less important coefficients. Algorithmic noise tolerance and N-modular redundancy techniques are investigated for DCT based image coding system in [5]. Memory, power and image quality trade-offs have been studied in [6] where memory banks that store MSBs are operated at a different voltage level than the ones that store LSBs, thereby achieving power reduction with some degradation in image quality.

In this work, we investigate ways to compensate for errors due to process variation and voltage scaling in JPEG implementations. Specifically, we propose an algorithm-specific technique to mitigate errors that result during the computation of 2D-DCT, quantization as well as errors in memory units. We exploit three features of encoded JPEG data: i) the number of sign extension bits is determined in the quantization step, ii) two adjacent AC coefficients after zig-zag scan have similar values, iii) coefficients corresponding to higher frequencies generally have smaller values. The proposed technique allows us to operate at various compression rates even for very high bit error rates with small drop in performance. For instance, when the BER is 10^{-4} , we achieve approximately 1.5 dB degradation compared to error-free performance at compression of 0.75 bpp and 4 dB improvement compared to no correction case. Such a BER arises when the supply voltage is dropped from 1V to 0.9V in 45 nm technology [7]. Synthesis results show that the hardware overhead of the method is fairly small. Thus this technique will allow JPEG implementations to be operated at low voltages with only a mild degradation in image quality.

The paper is organized as follows. We present the general description of JPEG in Section 2, followed by analysis of failures in computation and memory units in Section 3. The proposed algorithm-specific technique is presented in Section 4. Simulation results illustrating the performance of the techniques and synthesis results of overhead circuitry are described in Section 5.

2. JPEG SUMMARY

The general block diagram of a JPEG encoder/decoder is shown in Figure 1. The original image in pixel domain is divided into 8x8 blocks which are transformed into frequency domain using 2 dimensional (2D) DCT. Since 2D DCT is separable, it is implemented using 1D transform along rows followed by 1D transform along columns. DCT is followed by quantization, where the coefficients are scaled by factors that depend on the desired image quality and/or compression rate. Next, zig-zag scanning is used to order the 8x8 quantized coefficients into a one dimensional vector (1x64 format) where low frequency coefficients are placed before the high frequency coefficients. The entropy coder generates the compressed image using Huffman coding.

The rate and quality of the image is determined at the quantizer. In order to achieve different quality and compression rates, the quantization matrix is multiplied with a quality factor that

* This work was funded in part by NSF CSR0910699.

is determined with the help of quality metric (Q) which ranges from 1 to 100 [9]. A lower Q result in lower image quality and compression rate. Figure 2-a illustrates JPEG luminance quantization table for $Q = 50$. Note that high frequency components which are at the bottom right corner are quantized aggressively while low frequency components which are at the top left corner are mildly quantized. Figure 2-b also shows the zig-zag scanning order. The very first element is the DC coefficient which is encoded in differential order by subtracting the DC coefficient of the previous block and encoding the difference using a Huffman table in baseline JPEG; the rest of the coefficients are AC coefficients, which are encoded using another Huffman table. While this paper focuses on baseline JPEG, the proposed algorithm specific technique is applicable to the other modes as well.

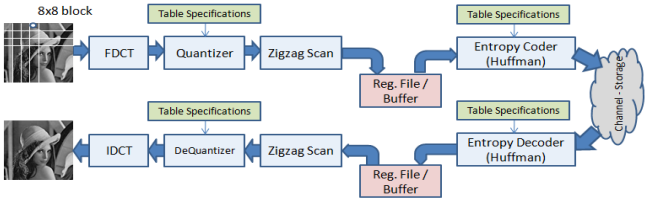
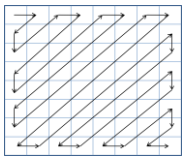


Figure 1: Block Diagram of JPEG

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

(a)



(b)

Figure 2: a) Luminance Quantization Matrix for $Q = 50$; b) Zigzag scan order for a 8x8 block

3. SOURCES OF ERROR

In this section we investigate the source of hardware errors in a JPEG implementation. We focus on failures in the data path (forward DCT, quantizer), and memory (register file, buffer).

Datapath: Failures in the datapath can happen because of critical path violation due to aggressive voltage scaling and/or process variation during computation of i) DCT along rows, ii) DCT along columns and iii) quantization. Assume that a single datapath violation occurs during 1D DCT along rows that result in a single miscalculated coefficient. This failure affects the values of eight 2D-DCT coefficients along a column of 8x8 DCT. Fortunately, after zig-zag scan, the miscalculated coefficients in a column are separated. Also, an error introduced during computation of DCT along columns and/or quantization does not propagate to other coefficients. The main computation blocks in a DCT are adders and multipliers, where the multipliers are typically implemented by hardwired shifters and adders. In the rest of this section, we derive the error probability distribution of ripple carry adder (RCA) and use the results to generate the error models. The same procedure can be applied to derive models of other computation units. We choose RCA since it is one of the efficient structures for 12-bit data-path [10]. Figure 3 illustrates the general block diagram of a 12-bit RCA adder where 3 of the longer paths are highlighted.

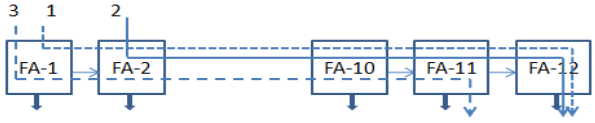


Figure 3: Block Diagram of 12-bit RCA

Assume that the delay of each full adder (FA) is the sum of nominal delay, t_{FA}, systematic variation t_{SYS}, which is typically considered same for all the FAs in a 12-bit RCA, and random variation $t_{r,-}$ which can be modeled using zero mean iid Gaussian random variable with variance σ_{FA}. Then delay of each carry chain starting from the x^{th} FA and ending at the y^{th} FA can be calculated as

$$T_{chain}(x,y) = (x-y) * (t_{FA} + t_{SYS}) + (t_{r,x} + \cdots + t_{r,y})$$

which can be simplified using the iid Gaussian properties as:

$$T_{chain}(x-y) = (x-y) * (t_{FA} + t_{SYS}) + \sqrt{(x-y)} \times t_r.$$

Here $T_{chain}(x-y)$ is a Gaussian variable with $\mu = (x-y) * (t_{FA} + t_{SYS})$ and $\sigma = \sqrt{(x-y)} \times \sigma_{FA}$. Also, the delay of any chain can be represented using only 12 different distributions $T_{chain}(1)$ to $T_{chain}(12)$.

Let's now calculate the probability of errors for each bit at the output of this 12-bit adder. Assume that the critical path delay is t_{crt}. We have 12 different paths that may lead to MSB error over the carry chain: LSB to MSB, LSB+1 to MSB, LSB+2 to MSB etc, where each has a different delay distribution. In order to calculate the probability of error for MSB, we use the Bayes' theorem and sum all the probabilities as $p(t_{MSB} > t_{crt}) = \sum_{z=1}^{12} p(T_{chain}(z) > t_{crt} | chain = z) \times p(chain = z)$, where t_{MSB} is the path delay of MSB bit and $p(chain = z) = \frac{1}{2^z}$. Similarly, we have 11 different paths that may lead to an MSB-1 error and $p(t_{MSB-1} > t_{crt}) = \sum_{z=1}^{11} p(T_{chain}(z) > t_{crt} | chain = z) \times p(chain = z)$. Thus for each output bit we can calculate its error probability for a given t_{crt}.

In 45nm PTM models [7], $t_{FA} = 35ps, t_{SYS} = 5ps$ and $\sigma_{FA} = 5ps$ at nominal voltage (1V). t_{FA} and σ_{FA} increases as we lower the voltage. For instance, at 0.6V, $t_{FA} = 70ps, t_{SYS} = 5ps$ and $\sigma_{FA} = 10ps$. The distribution of errors for different voltage levels is shown in Figure 4-a when the allowable critical path is 400ps. For instance, when voltage is set to 0.8V, probability of error for MSB is 0.0033, MSB-1 is 0.0028, MSB-2 is 0.0018 and the rest are almost zero. The trend in our results is similar to the data-path error analysis in [8].

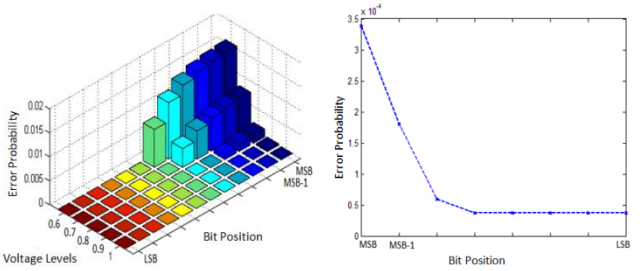


Figure 4: a) Probability of error distribution for 8-bit RCA for different voltage settings; b) Pdf of errors from MSB to LSB for BER=10^{-4}

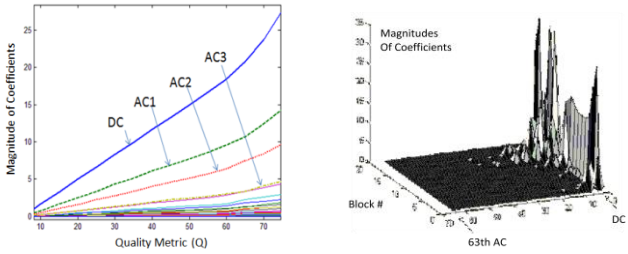
Memory: Errors in the register files and buffers are generally dominated by fully random errors due to random dopant fluctuation (RDF) which is modeled as a zero mean Gaussian random variable. Even though error levels are low at nominal voltage levels, as a result of voltage scaling, these number can be as high as 10^{-4} at around 0.85V and even 10^{-3} at 0.75V [11].

Figure 4-b shows the probability distribution function (pdf) of failures due to combination of data path delay violation and memory failures for each bit position when the overall BER is approximately 10^{-4}. We assume that each coefficient consists of only 8 bits after quantization. We also assume that the failures

generated during computation of DCT along rows, DCT along columns, quantizer and memory are independent of each other and that the errors generated in row DCT computation are propagated to the other units. The total error pdf is the sum of error probabilities of each unit.

4. PROPOSED ALGORITHM-SPECIFIC TECHNIQUE

It is well-known that in natural images neighboring pixels are highly correlated. It turns out that in frequency domain neighboring coefficients have similar values. Figure 5-a shows the average magnitude of the DC coefficient and several AC coefficients after zig-zag scan for different values of Q for Bridge image. These figures demonstrate three features of the data: a) there is a direct relation between coefficients and Q, b) there is a similarity in the magnitude between coefficients of two adjacent AC coefficients after zig-zag scan, c) coefficients corresponding to higher frequencies generally consist of smaller values. In addition, from our simulations, we find that coefficients of the same order but in consecutive blocks also have similar magnitudes. This is illustrated in Figure 5-b which shows 64 coefficient values of the first 20 blocks of Bridge image when $Q = 50$.



(a) (b)

Figure 5: Magnitude of DC and AC coefficients a) averaged over all blocks; b) first 20 blocks of Bridge image

Recall that while 11-bits are sufficient to represent any 8x8 DCT coefficient [12], quantization stage determines the number of bits used to represent each coefficient. For instance, when $Q = 50$, the 5th AC (AC5) coefficient which is originally 11 bits is quantized and rounded to $AC_q(5) = round(\frac{AC5}{10})$ which is represented with 8-bits (bold in Table-1). Table-1 specifies how many bits are sufficient to represent the coefficients after quantization step for different values of Q. In order to reduce the complexity, we partitioned the 64 coefficients into 4 groups: Group-1 consists of coefficients DC to AC-15, Group-2 consists of AC-16 to AC-31, and so on. Table-1 lists the number of bits necessary for representing each group for different ranges of Q.

Quantizer	Group-1	Group-2	Group-3	Group-4
Q=<5	5	4	3	2
5<Q=<15	6	5	4	3
15<Q=<30	7	6	5	4
30<Q=<55	**8**	7	6	5
55<Q=<70	8	7	6	6

Table 1: Number of bits necessary to represent each group of 2D DCT coefficients for natural images

Next, we describe the proposed algorithm-specific technique to mitigate the effect of data-path and memory errors. It consists of 3 steps.

Step 1: We use the information in Table-1 to detect the bit errors that occur in the sign extension bits of coefficients. When k bit representation is sufficient for Group l, then by definition, the sign extension bits k to MSB should be all zero for a positive number and all one for a negative number. We pick three bits from the sign extension bits to find the correct sign extension bits. This step is applicable to the groups that can be represented using 6 bits or less. False detection probability of this scheme is ${}^3C_2 BER_s{}^2(1 - BER_s) + BER_s{}^3$, where BER_s represents error rate probability of a single bit.

Step 2: We use the fact that coefficients that are adjacent to each other have similar magnitudes. We detect and correct an error when we find an abnormal increase in magnitude in one of the coefficients. The procedure is as follows. In order to detect an error in the j^{th} AC coefficient of the k^{th} block, we take the average of the two adjacent coefficients, namely, $(j-1)^{th}$ and $(j+1)^{th}$, and compare it with the j^{th} coefficient. If the difference is higher than a predetermined threshold, we calculate the average of the j^{th} AC coefficient of the $(k-1)^{th}$ and $(k+1)^{th}$ block and compare again with the j^{th} coefficient. If the difference is again higher than the threshold, we change the value of the j^{th} coefficient to the average of the two neighboring coefficients in the same block. The pseudo code for this step is given in Figure 6. Since each group specified in Table-1 has different bit width specifications, we assign different threshold levels for each group to reduce the false detection probability. For instance, the threshold value for Group-1 is 64 whereas it is only 8 for Group-4. These threshold values were determined by experimentation with a sample set of images.

```
BEGIN
1.   Initialize Parameters: Thresholds (THR_i) i=1,2,3,4
2.   FOR each AC coefficient LOOP
3.     IF ( AC(k,j)-( AC(k,j+1) + AC(k,j-1) )/2 > THR_i )
4.        IF ( AC(k,j)-( AC(k+1,j) + AC(k-1,j) )/2 > THR_i )
5.           AC(k,j)=( AC(k,j+1) + AC(k,j-1) )/2
END
```

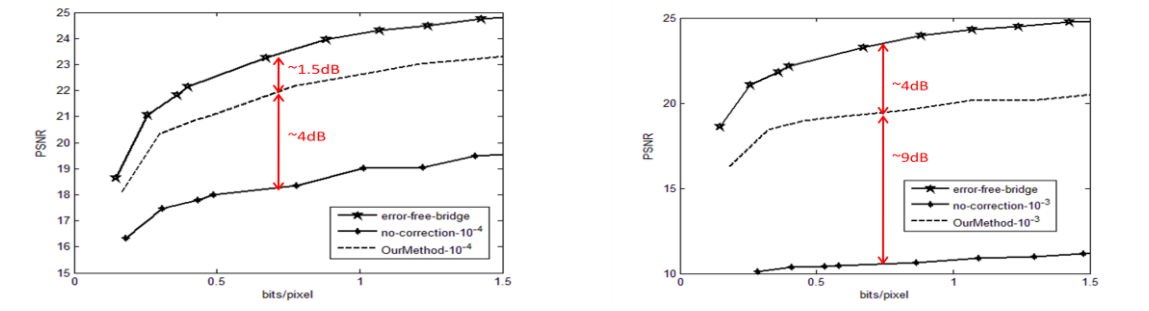
Figure 6: Pseudo code for Step 2

Step 3: For errors that are uniformly distributed such as those due to memory failures, we use another technique. Even though these errors are few in number, these bits become more important and degrade the performance greatly when they correspond to high frequency coefficients which are de-quantized with very large numbers at the decoder. The procedure to detect errors in bits which are not sign extension bits (such as bits 1-5 for Group-3 when Q=50) is as follows. Assume $bit\ (k,j,i)$ represents the bit in the i^{th} position (where i is $0 \le i \le 7$) of the j^{th} coefficient (where j is $0 \le j \le 63$) of the k^{th} block. When $bit(k,j,i)$ is 1, we check a group of bits at $bit(k-1{:}\,k+1, j, i-1{:}\,i+1)$ (bits from neighboring blocks same frequency coefficient) and $bit(k, j-3{:}\,j+1, i-1{:}\,i+1)$ (bits from the same block but consecutive frequency coefficients). If we detect another non-zero bit in this search space, we consider $bit\ (k,j,i)$ to be correct except when the non-zero bit occurs in a group of three consecutive ones $(bit(k', j', i-1{:}\,i+1)$ where k', j' are in the local search space). Based on our simulations, false detection probability of our method increases for lower bit-planes and so we constrained our search space to bit-planes 3 or higher. The area and power overhead of the additional storage is quite small.

5. SIMULATION RESULTS

5.1. PSNR Performance

The quality performance is described in terms of peak signal to noise ratio (PSNR), where the noise is defined as the difference



(a) (b)

Figure 7: PSNR vs. Compression rate performance for Bridge image when a) BER= 10^{-4} and b) BER=10^{-3}

between the original and the reconstructed image. The compression rate is measured in number of bits required to represent one pixel (bpp). Four representative images (Bridge, Baboon, Lena and Pepper) are used to generate the results. The error model (pdf) in Section 3 is used to characterize the errors in each bit position.

The performance of the proposed method when BER=10^{-4} and 10^{-3} are shown in Figure 7-a and b for the Bridge image. At $BER = 10^{-4}$, our method has 4 dB improvement over the no-correction case and a drop of approximately 1.5 dB compared to the error-free case at 0.75 bpp compression rate. At $BER = 10^{-3}$, quality degradation due to errors is very high as shown in Figure 7-b. However with the help the proposed technique, they can be partly recovered. Specifically, the improvement compared to no-correction case is approximately 9 dB at 0.75 bpp. Table 2 summarizes the performance of the proposed technique for 4 different images at compression rate of 0.75 bpp when BER=10^{-4}. Our proposed scheme improves the performance approximately 4 dB on average compared to no-correction case.

	Error-Free	No-Correction	Proposed Technique
Bridge	23.5	18.2	22.1
Baboon	23.7	18.6	22.4
Lena	34.5	27.1	32.3
Pepper	32.5	25.7	30.4

Table 2: PSNR values of proposed technique at 0.75 bpp compression rate when BER=10^{-4}

5.2. Overhead: Area, Delay, Power Consumption:

The hardware overhead of the algorithm-specific technique consists of majority voter, coefficient comparator and magnitude correction unit. These units were synthesized using Design Compiler and 45nm models from [13]. Majority voter scheme is used in the first step to detect errors in the sign extension of bits. Coefficient comparator is used to detect abnormality in magnitudes of neighboring coefficients. Magnitude correction unit is used to detect errors due to memory failures. Table 3 illustrates the power consumption, area and latency results of the three units for clock period of 1.25ns. We see that the overhead is fairly small, enabling us to operate at scaled voltage levels with small loss in image quality.

	Majority Voter	Coefficient Comparator	Magnitude Correction Unit
Area (um^2)	24	767	104
Worst-case delay (ps)	42	459	27
Active Power (uW)	3.6	96	6.5
Leakage Power (uW)	0.1	2.7	0.4

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

5. CONCLUSION

In this paper, we proposed a technique to mitigate errors in data-path (DCT along rows, columns and quantizer) and memory unit of JPEG implementations. We characterized the errors in these units and derived a pdf of errors as a function of the bit position. We presented an algorithm-specific technique that exploits the characteristics of the quantized coefficients including similarity between neighboring coefficients and the fact that high frequency coefficients have smaller values. The technique improves PSNR performance by approximately 4dB when BER=10^{-4} compared to the no-correction case with a degradation of about 1.5 dB in PSNR compared to the error-free case.

REFERENCES

[1] S. R. Sarangi, et al. "VARIUS: A Model of Process Variation and Resulting Timing Errors for Microarchitects", IEEE Transaction on Semiconductor Manufacturing, vol 21, no 1, pg. 3-13, Feb. 2008.

[2] T. Xanthopoulos, et al. "Low-Power DCT Core Using Adaptive Bitwidth and Arithmetic Activity Exploiting Signal Correlations and Quantization", IEEE Journal of Solid State Circuits, vol. 35, no. 5, pg. 740-750, May 2000.

[3] J. Park, et al., "Dynamic Bit-Width Adaptation in DCT: An Approach to Trade Off Image Quality and Computation Energy", IEEE Transaction on VLSI, vol 18, issue 5, pg. 787-793, May 2010.

[4] G. Karakonstantis, et al. "Process-Variation Resilient and Voltage-Scalable DCT Architecture for Robust Low-Power Computing", IEEE Transaction on VLSI, vol 18, issue 10, pg. 1461-1470, Oct. 2010.

[5] E.P. Kim et al., "Soft NMR: Analysis & Application to DSP Systems", ICASSP, pg. 1494 – 1497, March 2010.

[6] M. Cho, et al., "Accuracy-aware SRAM: a reconfigurable low power SRAM architecture for mobile multimedia applications", Asia and South Pacific Design Automation Conference, pg. 823-828, 2009.

[7] ptm.asu.edu.

[8] Y. Liu, et al., "Computation Error Analysis in Digital Signal Processing Systems with Overscaled Supply Voltage", IEEE Transaction on VLSI, vol 18, issue 4, pg. 517-526, April 2010.

[9] The independent JPEG Group, "The sixth public release of independent JPEG Group's Free JPEG Software", C Source code of JPEG Encoder research 6b, March 1998 (ftp://ftp.uu.net/graphics/jpeg).

[10] J. M. Rabaey, et al. "Digital Integrated Circuits: A Design Perspective", Second Edition, Prentice Hall, 2003.

[11] Y. Emre and C. Chakrabarti, "Memory Error Compensation Techniques for JPEG2000", IEEE Workshop on Signal Processing Systems, Oct 2010.

[12] Tinku Acharya, Ping-Sing Tsai, "JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI Architectures", Wiley Inter-Science, 2004.

[13] Nangate, Sunnyvale, California, "45nm Open Cell Library", http://www.nangate.com/.