Error Resilient Video Encoding Using Block-Frame Checksums

Joshua W. Wells, Jayaram Natarajan, and Abhijit Chatterjee
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332
Email: {josh.wells, jayaram, chat}@gatech.edu

***Abstract*—In this paper, a soft error resilient video encoder design is presented. The design uses the inherent architecture of modern video encoders as a basis for building a checksum based error detection mechanism. Modern video encoders use previously coded data in a video sequence as a reference to encode future data. Data prediction is efficient in terms of compression, but it enables errors to potentially affect large portions of a video sequence. The proposed Block Frame Checksum based error correction technique improves video quality by limiting error propagation. The design provides an on-line, end-to-end error detection scheme that minimizes the impact of soft-errors on video quality without reprocessing data. Additionally, scalable checksum processing provides a trade-off between power used for error detection and video quality.**

I. Introduction and Motivation

Modern video encoding techniques, such as H.264/AVC [1], rely heavily on the reuse of data. This increased data reuse has resulted in a drastic increase in both compression efficiency and computational complexity. Advancements in VLSI have complemented this by performing computations faster over generations of scaling thus satisfying bit rate requirements [2]. While advancements in the areas of compression and VLSI have enabled the continuous streaming of high definition video, error resiliency has become a major cause of concern. Using smaller, faster hardware increases the likelihood of encountering soft-errors [3]. Combined with increased compression techniques, errors occurring while encoding parts of a frame are propagated aggressively to other parts in the same frame and future frames [4]. Another major concern in modern electronics especially with the increasing demand for battery operated devices is power. In video applications, performance vs power is an important trade-off that determines video quality vs battery life. In the following, we first discuss prior concepts and the H.264 encoder architecture before presenting our soft-error detection and mitigation methodology which arrests the propagation of soft-errors and dramatically increases video quality.

II. Background and Prior concepts

Soft errors are transient faults that occur when energetic particles (alpha particles, protons and neutrons) strike on the silicon device generating an electron-hole pair. The source of such errors could be from a packaging material or cosmic rays. In the recent years, due to decreasing feature size and supply voltage, the impact of soft error on a silicon device has increased exponentially [5]. The Soft Error Rate (SER) is given by (1) where E is the intensity of the energy particle, $Area$ is the cross sectional area of the node, $Q_{critical}$ is the critical value of total charges collected due to electron-hole pair generation and Q_s is the charge collection efficiency.

$$SER \propto E \times Area \times e^{-Q_{critical}/Q_s} \tag{1}$$

When the amount of charges collected due to electron hole pair goes above the $Q_{critical}$ value, the boolean logic may invert. Since the value of $Q_{critical}$ is proportional to supply voltage and node capacitance, SER increases exponentially with scaling.

Due to device scaling, soft errors have become a major source of errors in digital logic. Researchers have proposed techniques to mitigate soft errors using a variety of methods summarized in [5]. Selective hardening has been proposed as a specific method for reducing soft errors in video and image processing [6] [7]. In [8] a checksum based probabilistic error compensation technique is proposed for detecting and correcting transient errors in linear digital systems. In [9], [10], and [11] the authors propose a technique to mitigate soft errors in specific blocks of an encoder. Extensive work has been done on error resilience in multimedia applications to combat errors occurring during transmission [4] [12]. The presented work is focused on detecting and mitigating soft errors, or transient errors, while a video frame is being encoded.

III. MPEG/H.264 Encoder

Video data consists of a sequence of frames corresponding to advancing values of time. Each frame is divided into $N \times N$ sub-arrays of pixels, called macroblocks (MB) with each MB being encoded independently. The block diagram of a typical MPEG encoder and decoder is shown in Fig. 1 and Fig. 2 respectively. The dashed components are not part of a typical encoder and will be discussed in Section VI-A. The forward path consists of a summing block and the transform / quantize block. The feedback path consists of an inverse transform / inverse quantize block, a summing block, and memory. Spatial encoding of a MB, referred to as intra frame encoding, is

This material is based upon work supported by the National Science Foundation under Grant No. CCR-0834484.

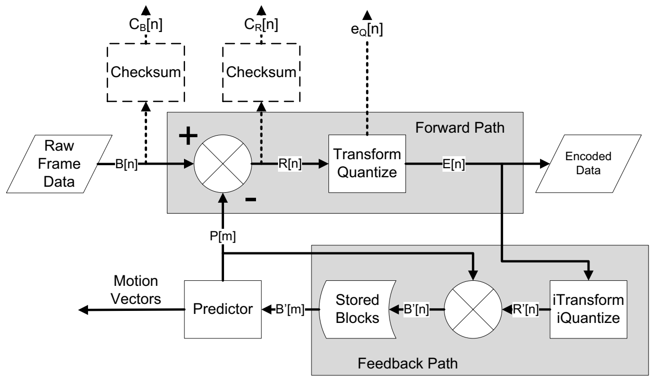


Fig. 1. Basic predictive encoder with BFC

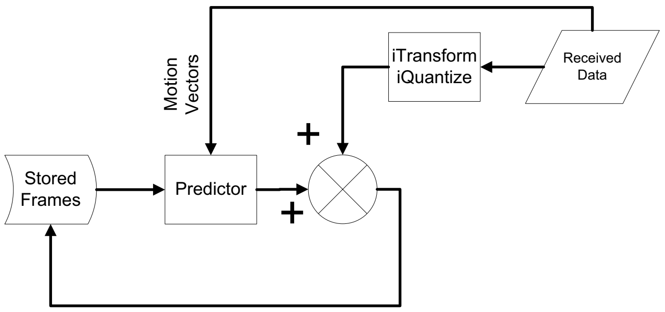
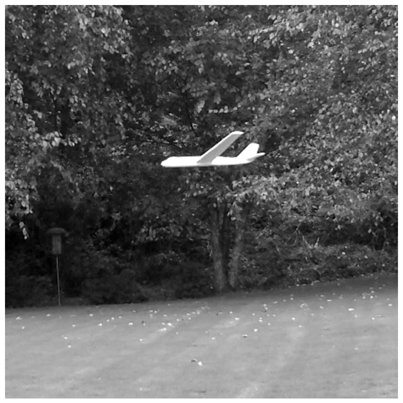


Fig. 2. Basic decoder


(a)

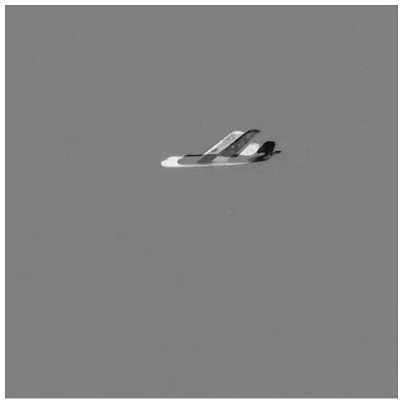

(b)

Fig. 3. Raw image (a) with its residue equivalent (b)

accomplished by using a previously encoded, adjacent MB to predict a given MB in the same frame. Temporal encoding, referred as inter frame encoding, encodes the data in a given MB using some previously encoded data from past frames. In both cases, the difference in the actual MB and the predicted MB, known as residue, is stored/transmitted. Fig. 3 shows a frame and its residue — in this case the residue in Fig. 3b is the difference between the pixel values for Fig. 3a and the pixel values of the previous encoded frame (not shown.) At the decoder side, successive MBs are reconstructed from previously decoded MBs.

IV. Impact of Macroblock Errors on Operation of Encoder and Decoder

Due to the nature of soft errors, an error can occur anywhere in hardware and affect any logical block in Fig. 1. By saying blocks, we refer to the combinational and sequential logic elements and memory associated with that block. For understanding the error propagation mechanism in detail, we consider two possible cases — soft errors occurring in the forward path and soft errors occurring in the feedback path. Both case are shown in Fig. 4.

Soft errors that occur in the forward path will affect the quality of the video differently than soft errors occurring in the feedback path. An error in the forward path produces an erroneous MB that is used as a reference for some future predictions by both the encoder and the decoder. Regardless of the validity of the MB, if the encoder and decoder agree on the the contents of the MB, any predictions that use the MB as a reference will be error-free.

An error in the feedback path produces an erroneous MB that is used as a reference for predictions by the encoder only. The decoder will perform predictions from the error-free MB. Each time the encoder uses an erroneous MB to predict a MB, the result is another erroneous MB. So once an error is introduced in the feedback path, the results of the error will propagate through multiple MBs until none of the erroneous MBs are used as references.

V. Overview of Proposed Work

In this paper, we propose a low cost error resilient video encoder using checksums of the pixel values corresponding to each MB. Error detection is performed by validating the checksum values for images obtained at minimal, specific nodes of the H.264 flow graph (architecture) of Fig. 1. The checksums calculated for the current frame are modified and reused to detect errors while encoding the next frame thus minimizing computational overhead and hence power. As a result of the checksum validation procedure, all the blocks in the encoder are made resilient to soft errors. In addition, a simple feedback error compensation mechanism is used

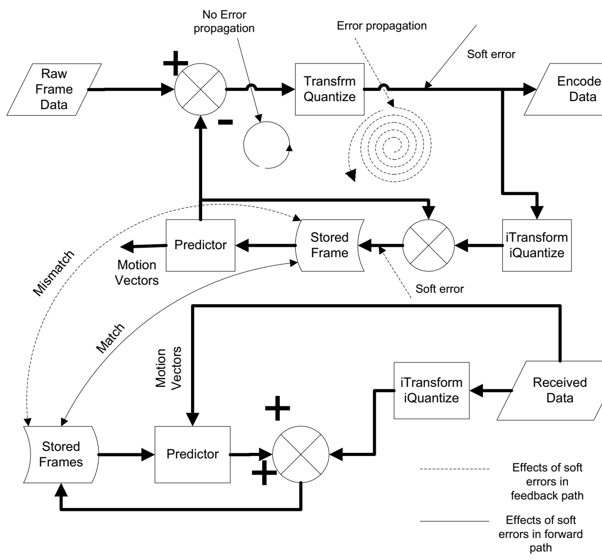


Fig. 4. Impact of soft errors on encoder and decoder

108	217	168	25	112	114
234	239	44	210	98	165
203	174	181	178	196	181
245	194	9	81	203	193
168	190	71	243	48	71
10	101	12	9	125	174

(a)

30	191	233
48	193	5
213	79	162

(b)

Fig. 5. Hypothetical video frame (a) and corresponding Checksum values (b)

to direct the encoder to skip erroneous frames or parts of frames that are used to predict successive frames (typically this prediction is used to minimize the volume of data in the transmitted residue).

The proposed work is a system level approach which uses the inherent architecture of the encoder to mitigate soft errors in the logic and memory elements of the all encoder blocks. As discussed in Section IV, errors in the feedback path of Fig. 1 can lead to unacceptable degradation in video quality and eventually complete loss of video. The aim is to mitigate such quality degradation and maintain overall video quality within acceptable levels.

The encoder block diagram shown in Fig. 1 does not include the entropy encoder. Entropy encoded data is extremely sensitive to error due to the nature of variable length coding. An error could lead to the the entropy decoder losing synchronization and decoding completely wrong data after the occurrence of error. Robust channel coding schemes are implemented to prevent errors during transmission [4]. We assume that if soft errors occur in the entropy encoder block, the decoder will identify this as a transmission error and take appropriate action.

VI. Proposed Block Frame Checksum Design

A. Block-frame Checksums

The block-frame checksum (BFC) is introduced as a an error compensation mechanism. The BFC is a certificate assigned to each MB. The BFC can be processed in parallel with its corresponding MB such that there exists an operation that can be performed on the BFC to match any operation performed to its MB. After operations have been performed to the MB and the BFC, the BFC should still be able to verify the data in the MB. The BFC is implemented by taking the modulo sum of every element in the corresponding MB. Computation of a BFC is shown by (2) where B is a MB with dimensions $N \times N$ and the number of bits used to represent the checksum is l. Pixel values in B are addressed by i and j.

$$BFC(B) = \sum_{i=1}^{N} \sum_{j=n}^{N} B_{i,j} \pmod{2^l} \tag{2}$$

Fig. 5 shows the pixel and checksum values for a hypothetical video frame consisting of an array of 3×3 macroblocks, each macroblock consisting of a 2×2 array of pixels. For comparison, a standard video frame is of size 720 by 480 pixels and the MB size is 16×16.

Using $2^l = 256$ (modulus), the BFC values for each of the 2×2 macroblocks is given by $BFC(F_{m,n})$ where $F_{m,n}$ denotes the MB of the frame with index m, n. For example, $F_{1,1} = [[108, 217]; [234, 239]]$. The checksum values are computed as follows: $BFC(F_{1,1}) = (108+217+234+239)$ (mod 256) = 30; $BFC(F_{1,2}) = 191$; $BFC(F_{1,3}) = 233$; $BFC(F_{2,1}) = 48$; $BFC(F_{2,2}) = 193$; $BFC(F_{2,3}) = 5$; $BFC(F_{3,1}) = 213$; $BFC(F_{3,2}) = 79$; $BFC(F_{3,3}) = 162$. Note that the checksum values are stored in local memory within the encoder and are used to perform error detection as described below. A key aspect of this coding scheme is that (simultaneous) multiple errors in different macroblocks can be detected and steps to perform error compensation (using data from previous frames) can be performed.

Using modular addition to compute the checksum is important because of its additive property. When the detection of soft errors is discussed in Section VI-B, it will become apparent why the checksum function used must have this property. The values used for N and l impact the probability of not detecting an error. Determining the impact of soft errors on the effectiveness of the BFC is done by first assuming a bit error rate ber. A bit flip error has equal probability of flipping a one to a zero as a zero to a one assuming an equally weighted distribution of ones and zeros in the video data. Therefore, the probability of a change in bit value, represented by the random variable A, is given by (3) where -1 indicates a change from a 1 to a 0, 1 indicates a change from a 0 to a 1, and 0 indicates no error.

$$P(A = a) = \begin{cases} ber/2 & a = \{-1, 1\} \\ 1 - ber & a = 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The magnitude of soft errors on a single pixel value is shown by considering the probability of a bit error at every bit position of a number and the value of the bit's position. The amount that an l-bit value will deviate by is given by the random variable D which is calculated as shown in (4).

$$D = \sum_{i=0}^{l-1} 2^i A \tag{4}$$

The random variable E in (5) gives the total deviation of the BFC with block dimensions $N \times N$, and word size l.

$$E = \sum^{N^2} D \pmod{2^l}$$
$$E = \sum^{N^2} \sum_{i=0}^{l-1} 2^i A \pmod{2^l} \tag{5}$$

The probability that the total deviation of a BFC is zero is shown in (6). The equation is a Fourier domain representation of the repeated convolution of the probability density function of D. This includes instances where the block the checksum is calculated for is error-free.

$$P(E=0) = \mathcal{F}^{-1}\{\mathcal{F}\{f_D(0)\}^{N^2}\} \pmod{2^l} \tag{6}$$

Subtracting the probability of an error-free block (7) yields the probability that an error will go undetected. Assuming a standard macroblock size of 16×16, an optimal word length for the BFC can be selected. Fig. 6 shows the tradeoff between BFC word length and and the bit error rate of the system. If the bit error rate for the device is known to be at most a certain value, the BFC word length can be scaled down accordingly to just meet the maximum allowable undetectable error probability. If bit error rate statistics are not known, the maximum BFC word length should be selected.

$$P(U=1) = P(E=0) - (1 - (1 - ber)^{lN^2}) \tag{7}$$

B. Soft Error Detection

The detection of soft errors can be described at high level using a simple algorithm. To check MB $B[n]$:

1) Encode and store $B[n]$ and its BFC $C[n]$
2) Compute prediction $B_P[n+1]$ for next MB $B[n+1]$ from stored $B[n]$ and BFC $C_P[n+1]$ from $C[n]$
3) Subtract $B_P[n+1]$ from $B[n+1]$ and $C_P[n+1]$ from $C[n+1]$ to get $B_R[n+1]$ and $C_R[n+1]$ (residue)
4) Compute BFC $C'_R[n+1]$ for $B_R[n+1]$ and compare to $C_R[n+1]$

If the residue BFC's do not match, an error has been detected. Since a prediction is subtracted from the input MB, the BFC of the input MB subtracted from the BFC of the prediction should equal the BFC of their residue as shown in (8).

$$C'_R[n] = C_B[n] - C_P[m] \pmod{2^l} \tag{8}$$

The additive property of the BFC allows for the establishment of the the relationship shown in (9). The BFC associated

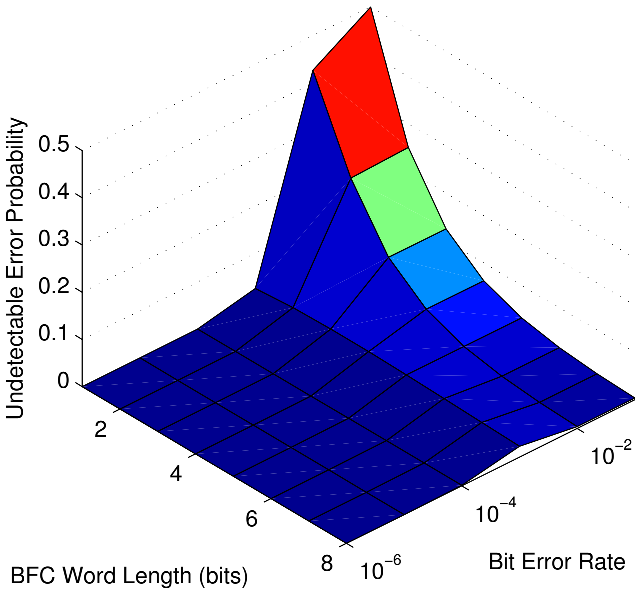


Fig. 6. Probability of an undetectable error in a 16×16 macroblock

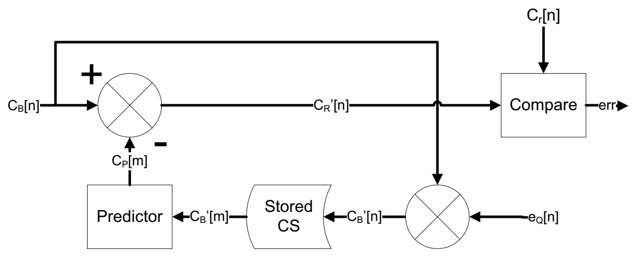


Fig. 7. Error detection architecture

with the sum of two MBs is equal to the sum of the BFC of each of the MBs. This means $C_P[m]$ can be subtracted from $C_B[n]$ to produce $C'_R[n]$ as shown in Fig. 7.

$$BFC(B_A + B_B) = BFC(B_A) + BFC(B_B) \pmod{2^l} \tag{9}$$

The three dashed components added to the typical encoder design shown in Fig. 1 are used by the error detection mechanism in Fig. 7 to indicate when some data has been computed incorrectly. $C_B[n]$ is the BFC associated with the raw input of a block which is assumed to be good. This checksum will accompany block $B[n]$ through its entire life cycle in the encoder. The BFC will be used for any future block that uses $B[n]$ as a prediction source. $C_P[m]$ represents a modified version of the BFC associated and stored with block $B[m]$. This BFC is transformed by the predictor block in an corresponding manner to the type of prediction that is being used on block $B[m]$. If the data for block $B[m]$ and the associated BFC $C'_B[m]$ have been processed correctly, the newly computed BFCs for $B[n]$ and $R[n]$ will cause (8) to evaluate as true. If the equation does not evaluate as true, an error has been detected in the processing of block $B[m]$

or $B[n]$. The compare block in Fig. 7 uses the equality to validate each block when it is used for a prediction.

The third input to the error detector is the quantization error for $e_Q[n]$. The including of quantization error is necessary due to the lossy nature of the transform and quantization followed by inverse transform and inverse quantization of the encoder shown in Fig. 1. The inverse transform and inverse quantizer are present in the encoder are part of the typical encoder design and are present to help ensure the encoder and decoder are predicting from the same data. The BFC for block $B[n]$ must be modified to reflect the information loss in the block. The error induced by quantization for a block is inherently known to the transform/quantizer. Taking the quantization error and adding it to the BFC for the quantized block, transforms $C_B[n]$ to $C'_B[n]$ so it can be properly associated with the changed block $B[n]$ to $B'[n]$.

With the modifications to the encoder, an efficient method for detecting errors has been created. Not only is it known if a frame is in error, the error is localized to a specific MB. Depending on the application of the encoder, the error can be corrected by reprocessing data or another method that will be discussed in the following section.

C. Soft Error Mitigation

Correcting errors detected using the technique described in the previous section given an unlimited amount of time is obvious — MBs not known to be correct should be reprocessed until an accurate result is achieved. The focus of error mitigation in this section is to minimize the impact of errors. Many real-time applications for video encoders do not allow unlimited time to reprocess information. In this case, the impact of the error can be minimized by preventing a detected error from propagating through multiple blocks. When a BFC compare fails as described in section VI-B, the MB used for prediction and the MB being encoded should be considered bad and removed from the list of stored blocks available to perform predictions. In addition to removing $B'[m]$ from memory, $B'[n]$ should never be committed to memory when an error is detected. Given the points in the architecture of the encoder where BFCs are calculated, it is not possible to know if a detected error is the result of a soft error encountered when computing the prediction block, the current block, or possibly the BFCs themselves. The only thing that can be said is both $B'[m]$ and $B'[n]$ contain data that is not known to be correct.

VII. Results

Encoder design for intra and inter prediction were considered independently. Simulation of errors is modeled by assuming a constant bit error rate at the input to each block in Fig. 1 and a constant frame rate. This means a constant error rate is achieved in time and the errors are roughly evenly distributed throughout the device. Frame quality was measured using peak signal-to-noise ratio (PSNR) as shown in (10) where m and n are the dimensions of reference frame F and decoded frame F'.

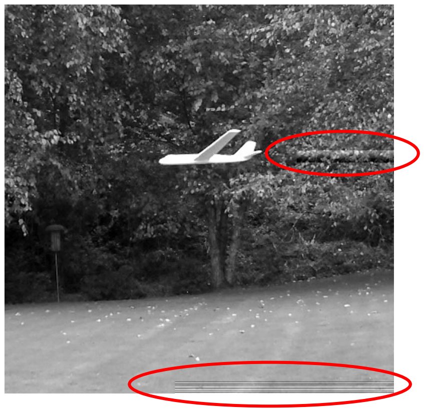


Fig. 8. Intra encoded frame with spatially propagating errors

$$PSNR = 20 \cdot \log_{10} \cdot \left(\frac{2^l - 1}{\sqrt{\frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \left[F(i,j) - F'(i,j)\right]^2}} \right) \quad (10)$$

A. Intra Prediction

For purposes of testing the proposed design, encoding a raster order intra frame is simulated. Using a block size of 16×16 and starting with the top-left most block in the frame, every block is predicted using data to its immediate left. The left-most blocks are predicted using the blocks immediately above them. Using this scheme it is possible for errors to propagate spatially down the left column of blocks and along any column leading right. Fig. 8 is a intra encoded version of Fig. 3b in the presence of soft errors. This particular frame was encoded with a BER of 10^{-7}. The circled areas show how a single bit-error's effect can extend beyond the local MB.

The use of BFCs in intra encoded frames evaluates favorably. Fig. 9 shows a comparison of a BFC and non-BFC encoder when performing intra encoding on a single frame. The higher quality achieved by the BFC encoder is possible because errors are detected and not allowed to propagate. The proposed technique shows an improvement in quality when the error rate is between 10^{-6} and 10^{-2}. When the error rate falls below 10^{-7}, errors are practically nonexistent, providing both encoder designs with optimal results. When the error rate climbs above 10^{-1}, the performance of the two encoders converges again, representing the compliment to the first case — practically all of the blocks contain errors. This represents a complete loss of video.

B. Inter Prediction

Soft errors in inter predicted frames have a larger impact on video quality than intra predicted frames. This is due to the substantially larger amount of data involved with inter prediction. Although they are not simulated here, motion vectors allow soft errors in inter predicted frames to propagate

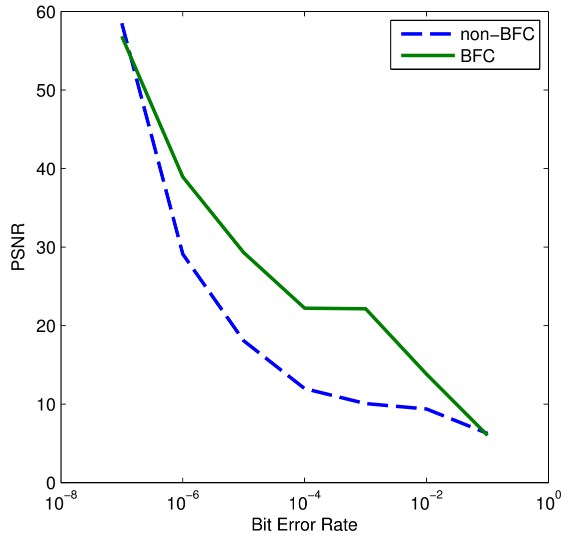


Fig. 9. Intra encoded frame quality in the presence of soft errors

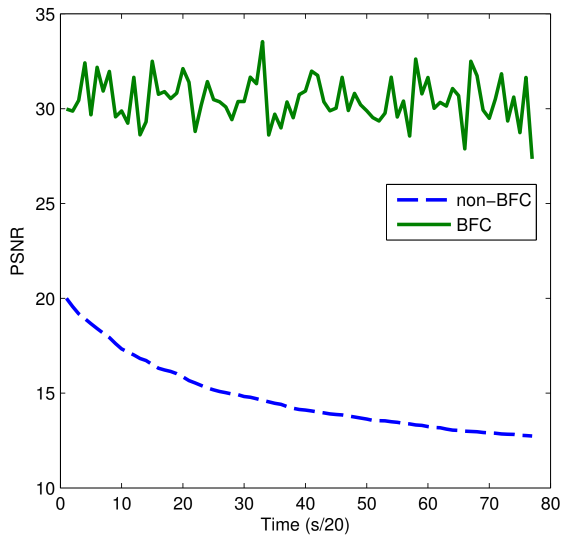


Fig. 10. Inter encoded frame quality in the presence of soft errors

spatially and temporally. For the purposes of simulating the performance of the BFC encoder using inter predicted frames, a simple prediction method was used. The predictor for each MB (excluding MBs in the first frame) is the MB in the same spatial position in the frame immediately preceding the frame being encoded.

Fig. 10 shows the performance using and not using BFC over a sequence of frames in the presence of soft errors with $ber = 10^{-6}$. Each fame is encoded using inter prediction except for the first which uses intra. The degradation of the quality of the non-BFC encoded frames can be seen over time. The BFC encoded sequence has variations in PSNR over time, but since the errors are detected, they are not used for subsequent predictions.

VIII. Conclusion

The problem of keeping the video encoder's reference frames in sync with the corresponding video decoder is a challenge that has been present since predictive encoders were first used. Using a predictive encoder in a real-time environment complicates the issue further. Through soft errors in the encoder, artifacts manifest themselves in the decoder due to differences in the reference frames stored on the encoder and the decoder. Block-frame checksums offer an error detection technique that can be implemented with minimal changes to the encoder / decoder. Through the use of a feedback loop, it is possible to detect errors anywhere along the loop's path in the encoder. BFCs can be checked in parallel with the processing of video frames with minimal additional cost. BFCs can greatly improve the quality of encoded video in the presence of soft errors.

References

[1] T. Wiegand, G. Sullivan, and A. Luthra, "Draft itu-t recommendation and final draft international standard of joint video specification (itu-t rec. h. 264—iso/iec 14496-10 avc)," *JVT-G050r1, Geneva, Switzerland*, 2003.

[2] V. De and S. Borkar, "Technology and design challenges for low power and high performance," in *Proceedings of the 1999 international symposium on Low power electronics and design*. ACM, 1999, p. 168.

[3] S. Borkar, T. Karnik, and V. De, "Design and reliability challenges in nanometer technologies," in *Proceedings of the 41st annual conference on Design automation*. ACM New York, NY, USA, 2004, pp. 75–75.

[4] Y. Wang, S. Wenger, J. Wen, and A. Katsaggelos, "Review of error resilient coding techniques for real-time video communications," *IEEE Signal Processing Magazine*, vol. 17, no. 4, pp. 61–82, 2000.

[5] R. Baumann, "Soft errors in advanced computer systems," *IEEE Design & Test of Computers*, vol. 22, no. 3, pp. 258–266, 2005.

[6] I. Polian, B. Becker, M. Nakasato, S. Ohtake, and H. Fujiwara, "Low-cost hardening of image processing applications against soft errors," in *21st IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2006. DFT'06*, 2006, pp. 274–279.

[7] D. Nowroth, I. Polian, and B. Becker, "A study of cognitive resilience in a jpeg compressor," in *IEEE International Conference on Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008*, 2008, pp. 32–41.

[8] M. Ashouei and A. Chatterjee, "Checksum-based probabilistic transient-error compensation for linear digital systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 10, pp. 1447–1460, 2009.

[9] K. Lee, A. Shrivastava, I. Issenin, N. Dutt, and N. Venkatasubramanian, "Mitigating soft error failures for multimedia applications by selective data protection," in *Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems*. ACM, 2006, p. 420.

[10] H. Cheong, I. Chong, and A. Ortega, "Computation error tolerance in motion estimation algorithms," in *IEEE International Conference on Image Processing, ICIP'06*, 2006.

[11] G. Varatkar and N. Shanbhag, "Error-resilient motion estimation architecture," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 16, no. 10, pp. 1399–1412, Oct. 2008.

[12] B. Girod and N. Farber, "Feedback-based error control for mobile video transmission," *Proceedings of the IEEE*, vol. 87, no. 10, pp. 1707–1723, Oct 1999.